Slide 1

CS540
Computer Networks II

Sandy Wang
chwang_98@yahoo.com

"Data and Computer Communications", 10/e, by William Stallings, Chapter 11 "Local Area Network Overview".

1. OVERVIEW

In this chapter, we look at the underlying technology and protocol architecture of LANs. Chapters 12 and 13 are devoted to a discussion of specific LAN systems.

Topics

1. Overview
2. LAN Switching
3. IPv4
4. IPv6
5. Routing Protocols -- RIP, RIPng, OSPF
6. Routing Protocols -- ISIS, BGP
7. MPLS
8. Midterm Exam
9. Transport Layer -- TCP/UDP
10. Congestion Control & Quality of Service (QoS)
11. Access Control List (ACL)
12. Application Layer Protocols
13. Application Layer Protocols continue
14. Others – Multicast, SDN
15. Final Exam

Reference Books

- **Routing TCP/IP Volume I, 2ne Edition** by Jeff Doyle and Jennifer Carroll

  ISBN: 1-57870-089-2
- **Routing TCP/IP Volume II** by Jeff Doyle and Jennifer DeHaven
  ISBN: 1-57870-089-2
- **Cisco CCNA Routing and Switching ICND2 200-101 Official Cert Guide, Academic Edition** by Wendel Odom -- July 10, 2013**.**
  ISBN-13: 978-1587144882
- **The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference** by Charles M. Kozierok – October 1, 2005.
  ISBN-13: 978-1593270476
- **CCNA Routing and Switching 200-120 Network Simulator.** By Wendell Odom, Sean Wilkins. Published by Pearson IT Certification.
- http://class.svuca.edu/~sandy/class/CS540/

Grading

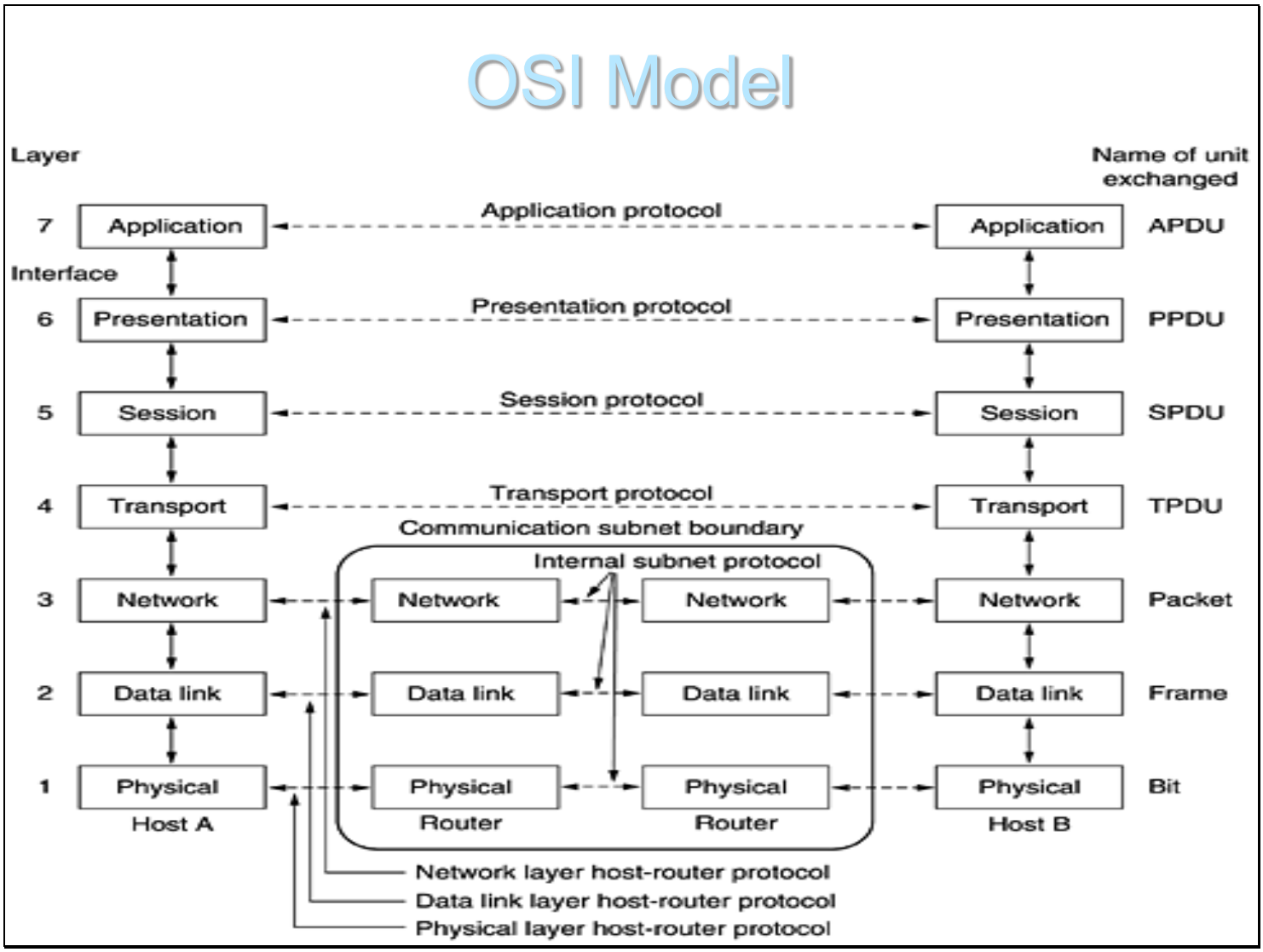
- Attendance → 10%
- Quiz/Lab → 20%
- Midterm Exam → 30%
- Final Exam → 40%

Grading Scale:

90 to 100	→ A
80 to 89	→ B
70 to 79	→ C
60 to 69	→ D
less than 60	→ F

No makeup for exams and quizs.

Slide 6


OSI Model
Layer
Name of unit exchanged
7
Application
Application protocol
Application
APDU
Interface
6
Presentation
Presentation protocol
Presentation
PPDU
5
Session
Session protocol
Session
SPDU
4
Transport
Transport protocol
Transport
TPDU
Communication subnet boundary
Internal subnet protocol
3
Network
Network
Network
Network
Packet
2
Data link
Data link
Data link
Data link
Frame
1
Physical
Physical
Physical
Physical
Bit
Host A
Router
Router
Host B
Network layer host-router protocol
Data link layer host-router protocol
Physical layer host-router protocol

Functions of Protocol Architecture

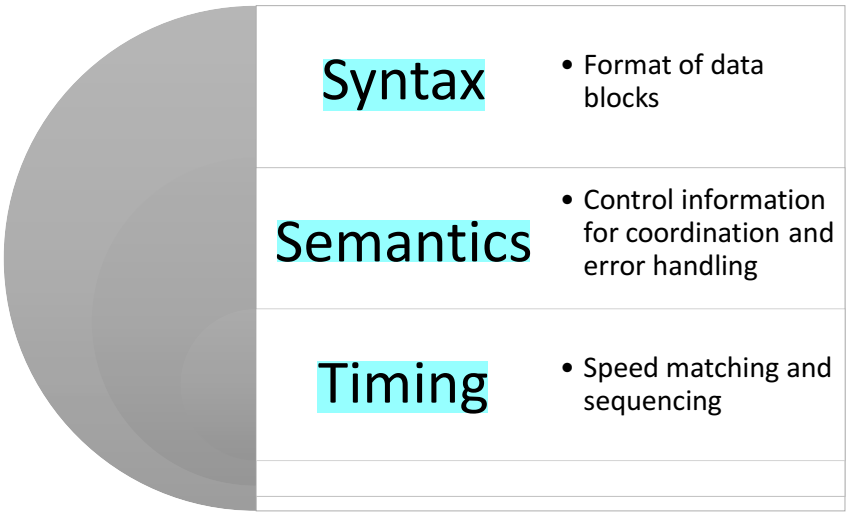
- Breaks logic into subtask modules which are implemented separately
- Modules are arranged in a vertical stack
  - Each layer in the stack performs a subset of functions
  - Relies on next lower layer for primitive functions
  - Provides services to the next higher layer
  - Changes in one layer should not require changes in other layers

It is clear that there must be a high degree of cooperation between the two computer systems. Instead of implementing the logic for this as a single module, the task is broken up into subtasks, each of which is implemented separately. In a protocol architecture, the modules are arranged in a vertical stack. Each layer in the stack performs a related subset of the functions required to communicate with another system. It relies on the next lower layer to perform more primitive functions and to conceal the details of those functions. It provides services to the next higher layer. Ideally, layers should be defined so that changes in one layer do not require changes in other layers.

Key Features of a Protocol

A protocol is a set of rules or conventions that allow peer layers to communicate

The key features of a protocol are:



Of course, it takes two to communicate, so the same set of layered functions must exist in two systems. Communication is achieved by having the corresponding, or peer , layers in two systems communicate. The peer layers communicate by means of formatted blocks of data that obey a set of rules or conventions known as a protocol .
The key features of a protocol are as follows:

• Syntax : Concerns the format of the data blocks

• Semantics : Includes control information for coordination and error handling

• Timing : Includes speed matching and sequencing

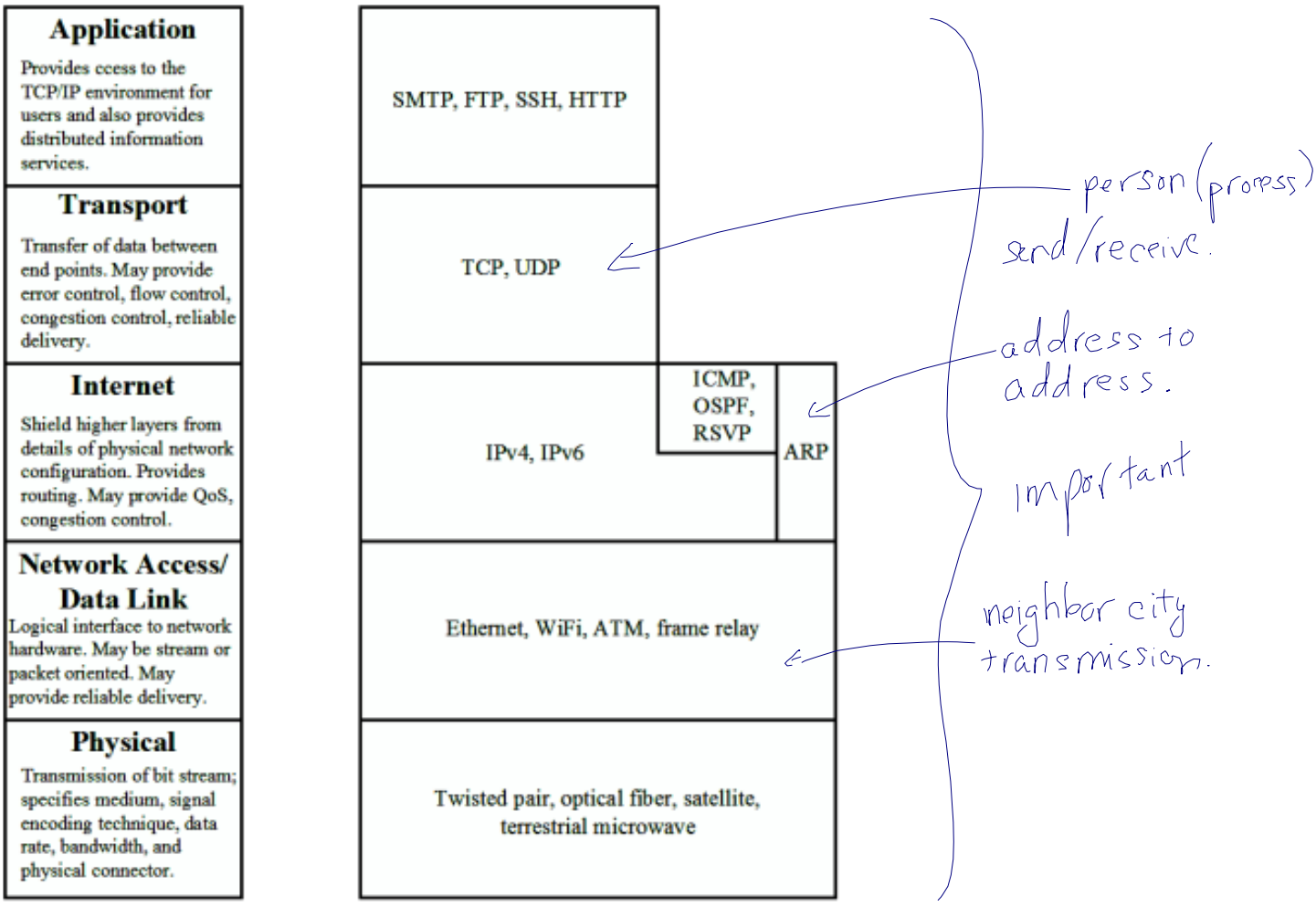


Figure 2.3 The TCP/IP Layers and Example Protocols

In general terms, computer communications can be said to involve three agents: applications, computers, and networks. Examples of applications include file transfer
and electronic mail. The applications that we are concerned with here are distributed applications that involve the exchange of data between two computer systems. These applications, and others, execute on computers that can often support multiple simultaneous applications. Computers are connected to networks, and the data to be exchanged are transferred by the network from one computer to another. Thus, the transfer of data from one application to another involves first getting the data to the computer in which the application resides and then getting

the data to the intended application within the computer. With these concepts in mind, we can organize the communication task into five relatively independent layers
(Figure 2.3):

- Physical layer
- Network access/data link layer
- Internet layer
- Host-to-host, or transport layer
- Application layer

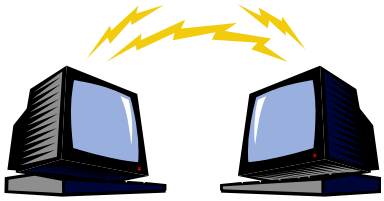
Physical Layer

- Covers the physical interface between computer and network
- Concerned with issues like:
  - Characteristics of transmission medium
  - Nature of the signals
  - Data rates

The physical layer covers the physical interface between a data transmission device (e.g., workstation, computer) and a transmission medium or network. This layer is concerned with specifying the characteristics of the transmission medium, the nature of the signals, the data rate, and related matters.

Network Access/Data Link Layer

- Covers the exchange of data between an end system and the network that it is attached to
- Concerned with:
  - Access to and routing data across a network for two end systems attached to the same network



The network access/data link layer is discussed in Section 2.2. This layer is concerned with access to and routing data across a network for two end systems attached to the same network.

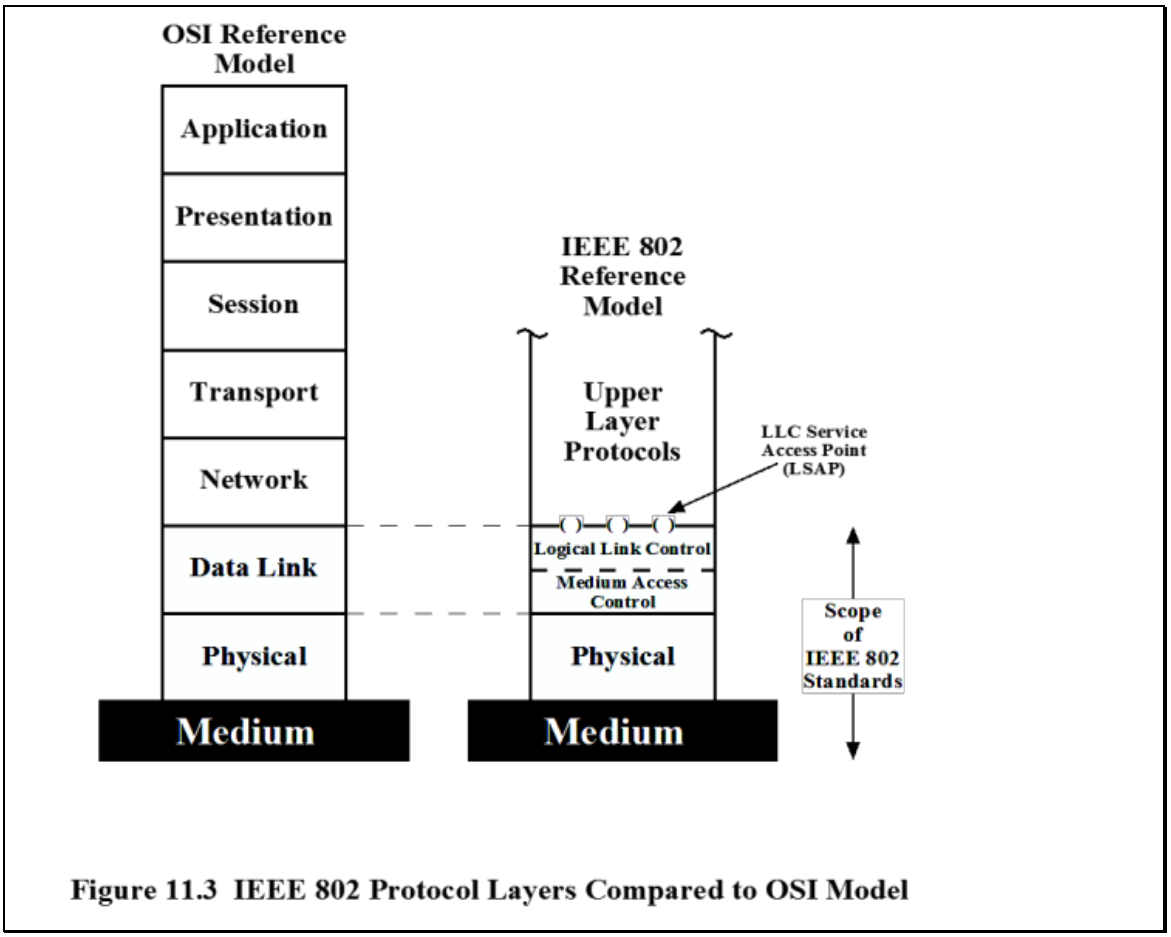
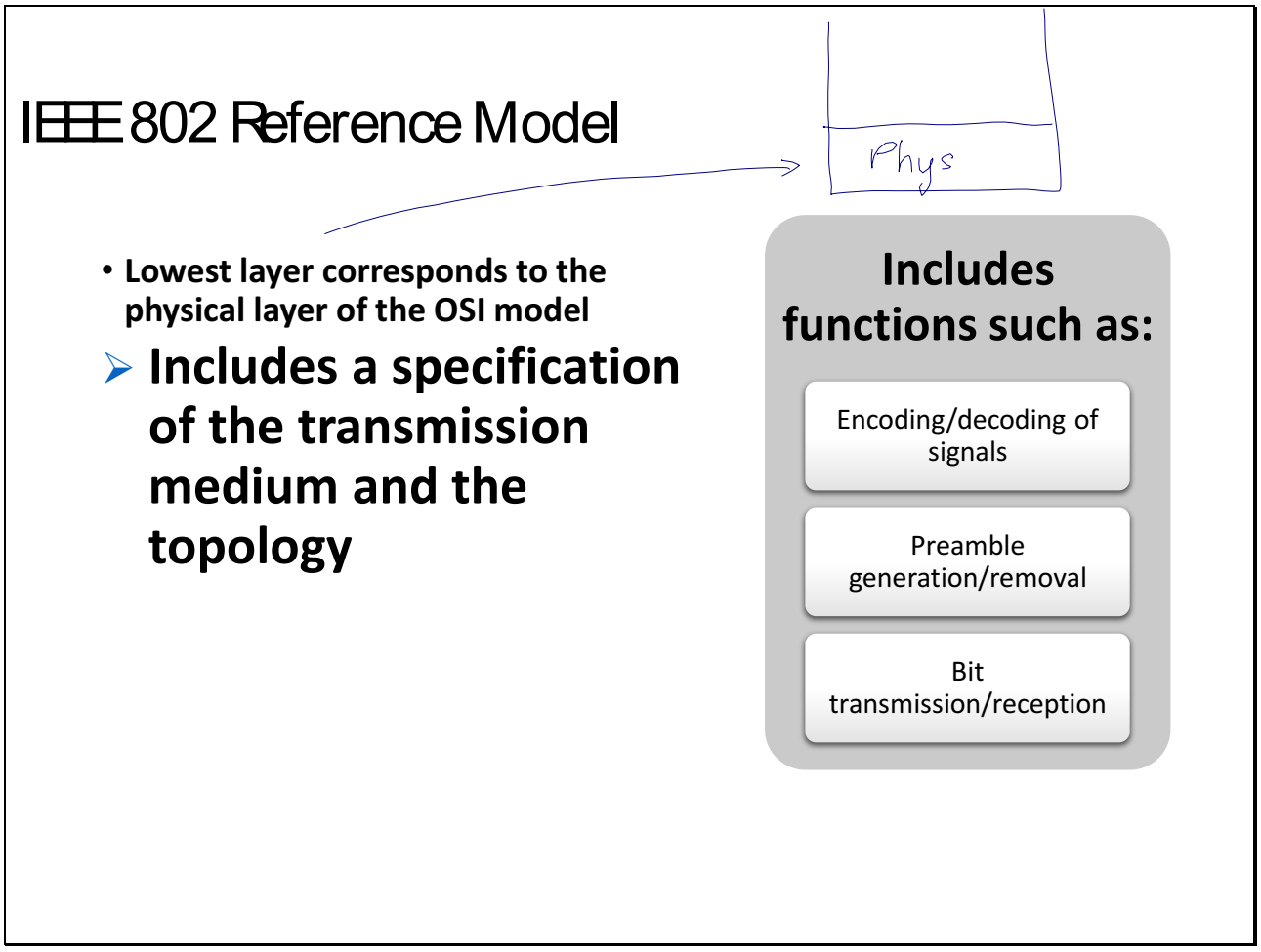


Figure 11.3 IEEE 802 Protocol Layers Compared to OSI Model

Protocols defined specifically for LAN and metropolitan area networks (MAN) transmission address issues relating to the transmission of blocks of data over the network. In OSI (open systems interconnection) terms, higher layer protocols (layer 3 or 4 and above) are independent of network architecture and are applicable to LANs, MANs, and WANs. Thus, a discussion of LAN protocols is concerned principally with lower layers of the OSI model.

Figure 11.3 relates the LAN protocols to the OSI architecture. This architecture was developed by the IEEE 802 LAN standards committee and has been adopted by all organizations working on the specification of LAN standards. It is generally referred to as the IEEE 802 reference model.



The lowest layer of the IEEE 802 reference model corresponds to the **physical layer** of the OSI model and includes such functions as

Encoding/decoding of signals

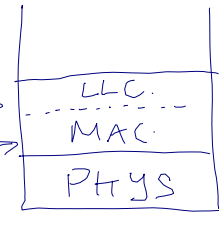
Preamble generation/removal (for synchronization)

Bit transmission/reception

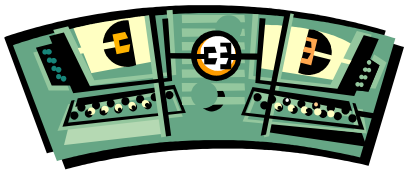
In addition, the physical layer of the 802 model includes a specification of the transmission medium and the topology. Generally, this is considered "below" the lowest layer of the OSI model. However, the choice of transmission medium and topology is critical in LAN design, and so a specification of the medium is included.

Slide 14

IEEE 802 Layers



- **Logical Link Control Layer (LLC)**
  - Provide interface to higher levels
  - Perform flow and error control
- **Media Access Control (MAC)**
  - On transmit assemble data into frame
  - On reception disassemble frame, perform address recognition and error detection
  - Govern access to LAN transmission medium



Above the physical layer are the functions associated with providing service to LAN users. These include

On transmission, assemble data into a frame with address and error-detection fields.

On reception, disassemble frame, and perform address recognition and error detection.

Govern access to the LAN transmission medium.

Provide an interface to higher layers and perform flow and error control.

These are functions typically associated with OSI layer 2. The set of functions in the last bullet item are grouped into a **logical link control (LLC)** layer. The functions in the first three bullet items are treated as a separate layer, called **medium access control (MAC)**. The separation is done for the following reasons:

The logic required to manage access to a shared-access medium is not found in traditional layer 2 data link control.

For the same LLC, several MAC options may be provided.

Slide 15

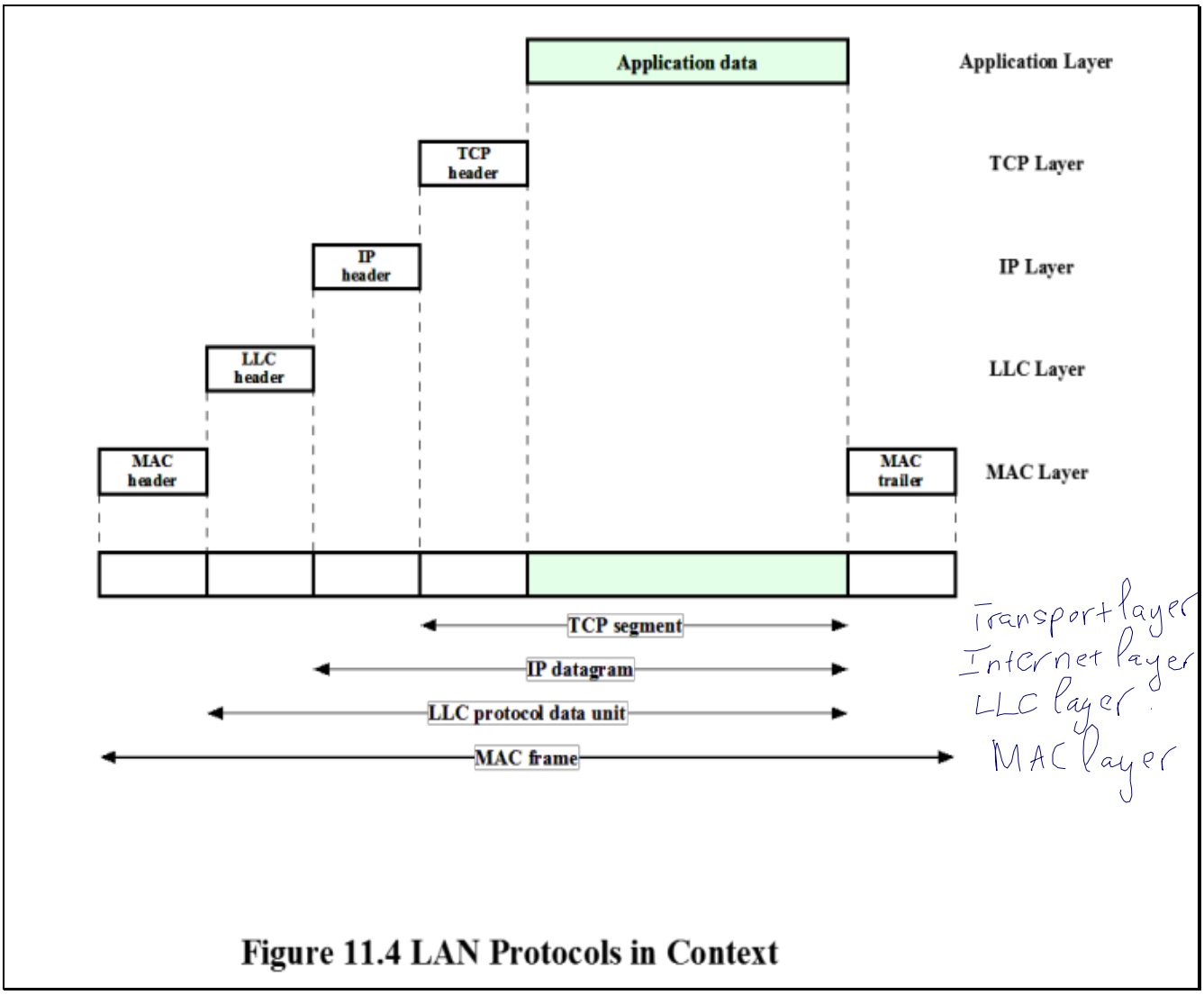
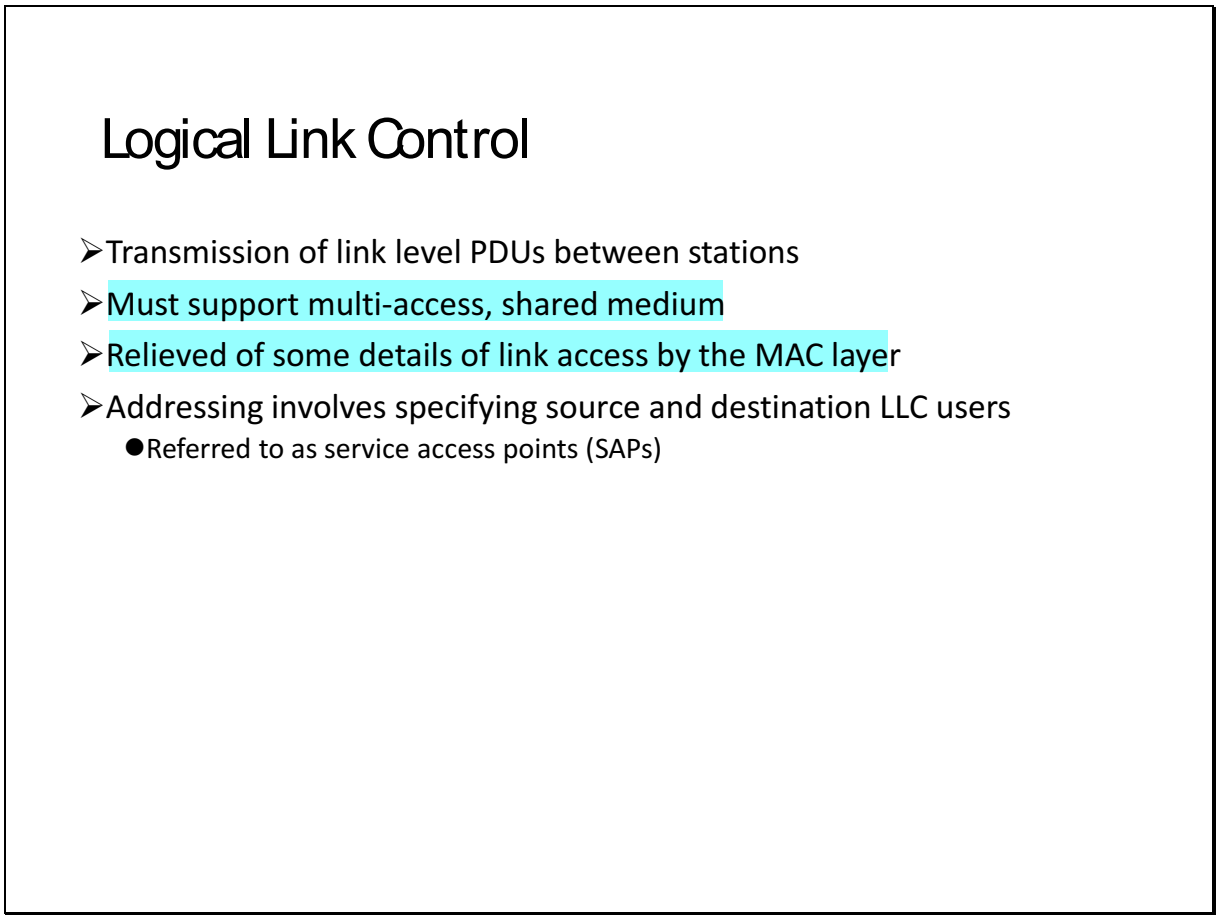


Figure 11.4 LAN Protocols in Context

Figure 11.4 illustrates the relationship between the levels of the architecture (compare Figure 2.5). Higher-level data are passed down to LLC, which appends control information as a header, creating an LLC protocol data unit (PDU). This control information is used in the operation of the LLC protocol. The entire LLC PDU is then passed down to the MAC layer, which appends control information at the front and back of the packet, forming a MAC frame . Again, the control information in the frame is needed for the operation of the MAC protocol. For context, the figure also shows the use of TCP/IP and an application layer above the LAN protocols.



The LLC layer for LANs is similar in many respects to other link layers in common use. Like all link layers, LLC is concerned with the transmission of a link-level PDU between two stations, without the necessity of an intermediate switching node. LLC has two characteristics not shared by most other link control protocols:

1. It must support the multi-access, shared-medium nature of the link (this differs from a multidrop line in that there is no primary node).

2. It is relieved of some details of link access by the MAC layer.

Addressing in LLC involves specifying the source and destination LLC users. Typically, a user is a higher-layer protocol or a network management function in the station. These LLC user addresses are referred to as service access points (SAPs), in keeping with OSI terminology for the user of a protocol layer.

We look first at the services that LLC provides to a higher-level user, and then at the LLC protocol.

Slide 17

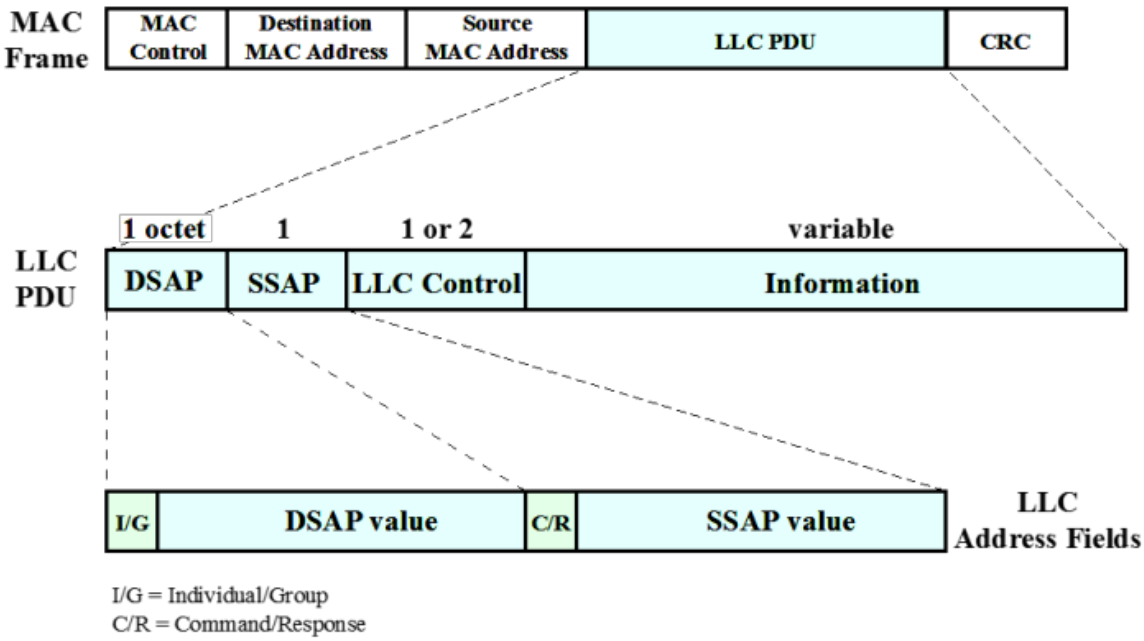
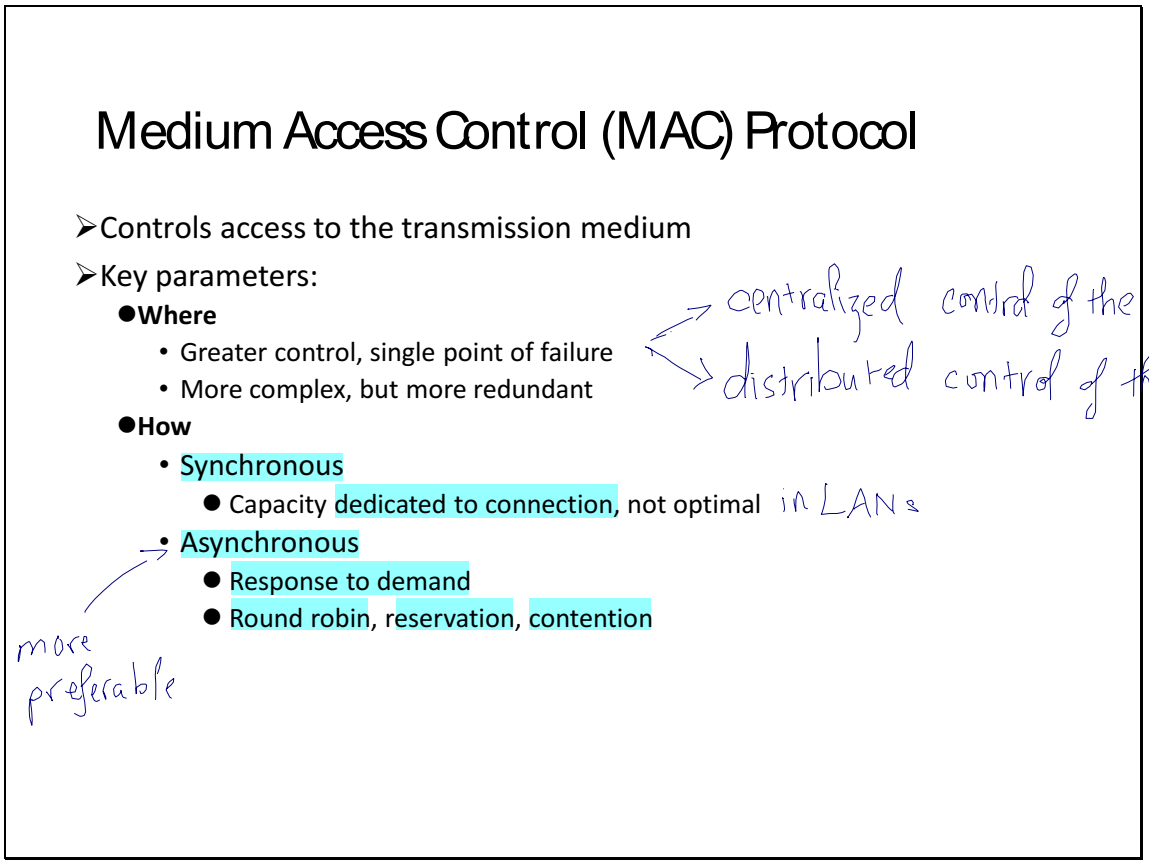


Figure 11.5 LLC PDU in a Generic MAC Frame Format

All three LLC protocols employ the same PDU format (Figure 11.5), which consists of four fields. The DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) fields each contain a 7-bit address, which specifies the destination and source users of LLC. One bit of the DSAP indicates whether the DSAP is an individual or group address. One bit of the SSAP indicates whether the PDU is a command or response PDU. The format of the LLC control field is identical to that of HDLC (Figure 7.7), using extended (7-bit) sequence numbers.

For type 1 operation , which supports the unacknowledged connectionless service, the unnumbered information (UI) PDU is used to transfer user data. There is no acknowledgment, flow control, or error control. However, there is error detection and discard at the MAC level.



All LANs and MANs consist of collections of devices that must share the network's transmission capacity. Some means of controlling access to the transmission medium is needed to provide for an orderly and efficient use of that capacity. This is the function of a medium access control (MAC) protocol.

The key parameters in any medium access control technique are where and how. *Where* refers to whether control is exercised in a centralized or distributed fashion. In a centralized scheme, a controller is designated that has the authority to grant access to the network. A station wishing to transmit must wait until it receives permission from the controller. In a decentralized network, the stations collectively perform a medium access control function to determine dynamically the order in which stations transmit. A centralized scheme has certain advantages, including

It may afford greater control over access for providing such things as priorities, overrides, and guaranteed capacity.

It enables the use of relatively simple access logic at each station.

It avoids problems of distributed coordination among peer entities

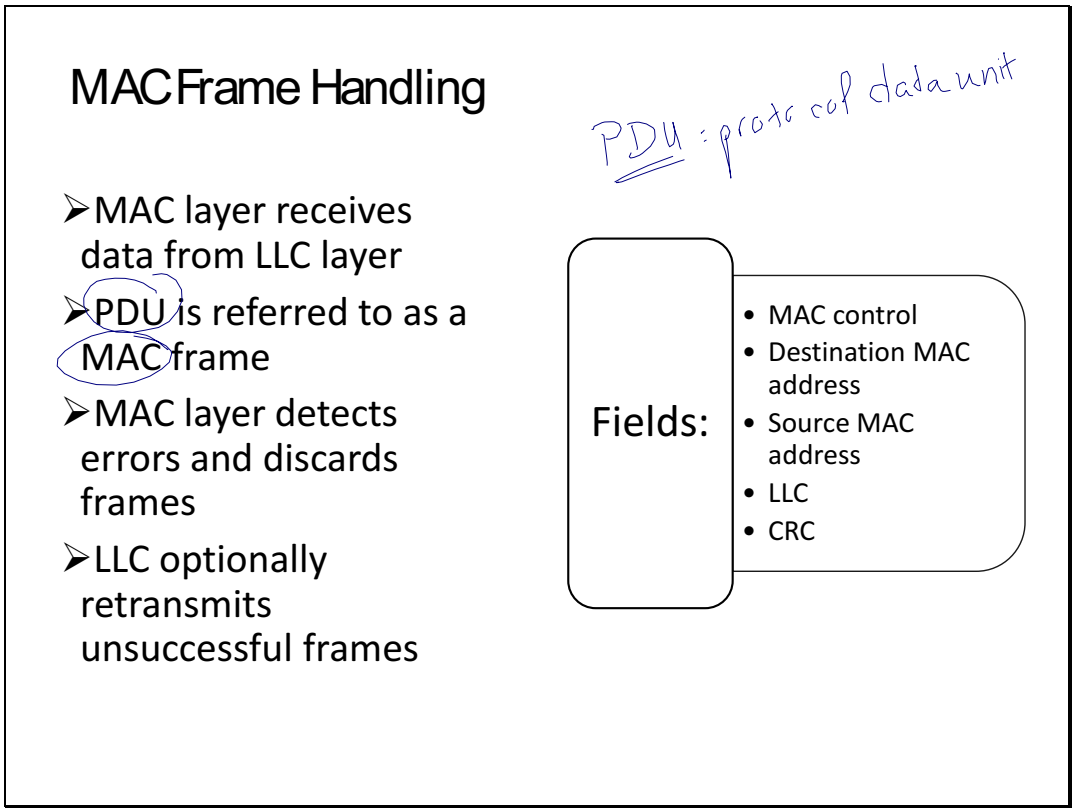
The principal disadvantages of centralized schemes are

It creates a single point of failure; that is, there is a point in the network that, if it fails, causes the entire network to fail.

It may act as a bottleneck, reducing performance.

The pros and cons of distributed schemes are mirror images of the points just made.

The second parameter, *how*, is constrained by the topology and is a tradeoff among competing factors, including cost, performance, and complexity. In general, we can categorize access control techniques as being either synchronous or asynchronous. With synchronous techniques, a specific capacity is dedicated to a connection. This is the same approach used in circuit switching, frequency division multiplexing (FDM), and synchronous time division multiplexing (TDM). Such techniques are generally not optimal in LANs and MANs because the needs of the stations are unpredictable. It is preferable to be able to allocate capacity in an asynchronous (dynamic) fashion, more or less in response to immediate demand. The asynchronous approach can be further subdivided into three categories: round robin, reservation, and contention.



The MAC layer receives a block of data from the LLC layer and is responsible for performing functions related to medium access and for transmitting the data. As with other protocol layers, MAC implements these functions making use of a protocol data unit at its layer. In this case, the PDU is referred to as a MAC frame.

The exact format of the MAC frame differs somewhat for the various MAC protocols in use. In general, all of the MAC frames have a format similar to that of Stallings DCC9e Figure 11.5. The fields of this frame are

MAC Control: This field contains any protocol control information needed for the functioning of the MAC protocol. For example, a priority level could be indicated here.

Destination MAC Address: The destination physical attachment point on the LAN for this frame.

Source MAC Address: The source physical attachment point on the LAN for this frame.

LLC: The LLC data from the next higher layer.

CRC: The Cyclic Redundancy Check field (also known as the frame check sequence, FCS, field). This is an error-detecting code, as we have seen in HDLC and other data link control protocols (Chapter 7).

In most data link control protocols, the data link protocol entity is responsible not only for detecting errors using the CRC, but for recovering from those errors by retransmitting damaged frames. In the LAN protocol architecture, these two functions are split between the MAC and LLC layers. The MAC layer is responsible for detecting errors and discarding any frames that are in error. The LLC layer optionally keeps track of which frames have been successfully received and retransmits unsuccessful frames.

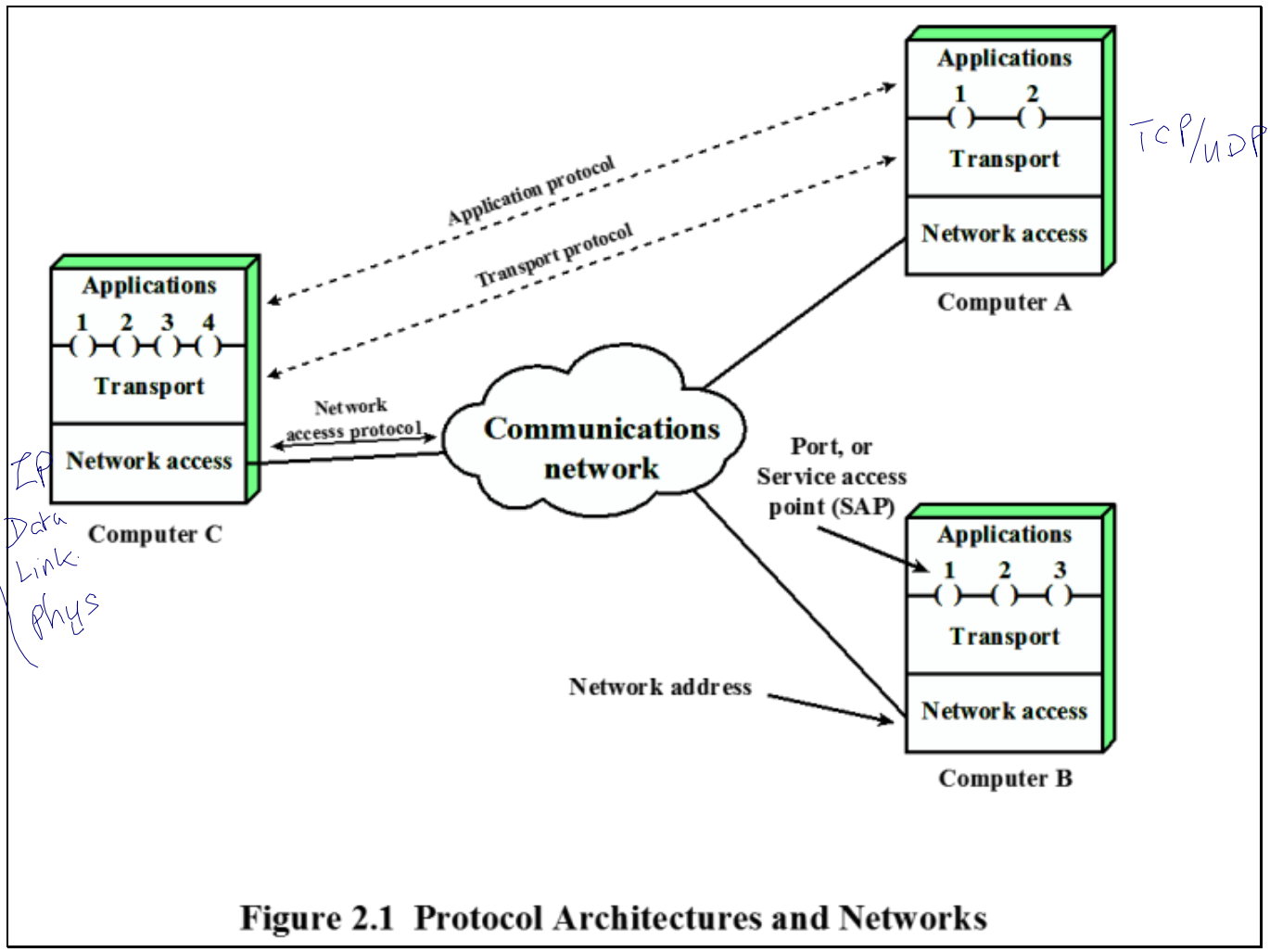


Figure 2.1 Protocol Architectures and Networks

Figures 2.1 and 2.2 illustrate this simple architecture. Figure 2.1 shows three computers connected to a network. Each computer contains software at the network access and transport layers and at the application layer for one or more applications. For successful communication, every entity in the overall system must have a unique address. In the three-layer model, two levels of addressing are needed. Each computer on the network has a unique network address; this allows the network to deliver data to the proper computer. Each application on a computer has an address that is unique within that computer; this allows the transport layer to support multiple applications at each computer. These latter addresses are known as service access points (SAPs), or ports , connoting the fact that each application is individually accessing the services of the transport layer.

Figure 2.1 indicates that modules at the same level (peers) on different computers communicate with each other by means of a protocol. An application entity (e.g., a file transfer application) in one computer communicates with an application in another computer via an application-level protocol (e.g., the File Transfer Protocol). The interchange is not direct (indicated by the dashed line) but is mediated

by a transport protocol that handles many of the details of transferring data between two computers. The transport protocol is also not direct, but relies on a network-level protocol to achieve network access and to route data through the network to the destination system. At each level, the cooperating peer entities focus on what they need to communicate to each other.

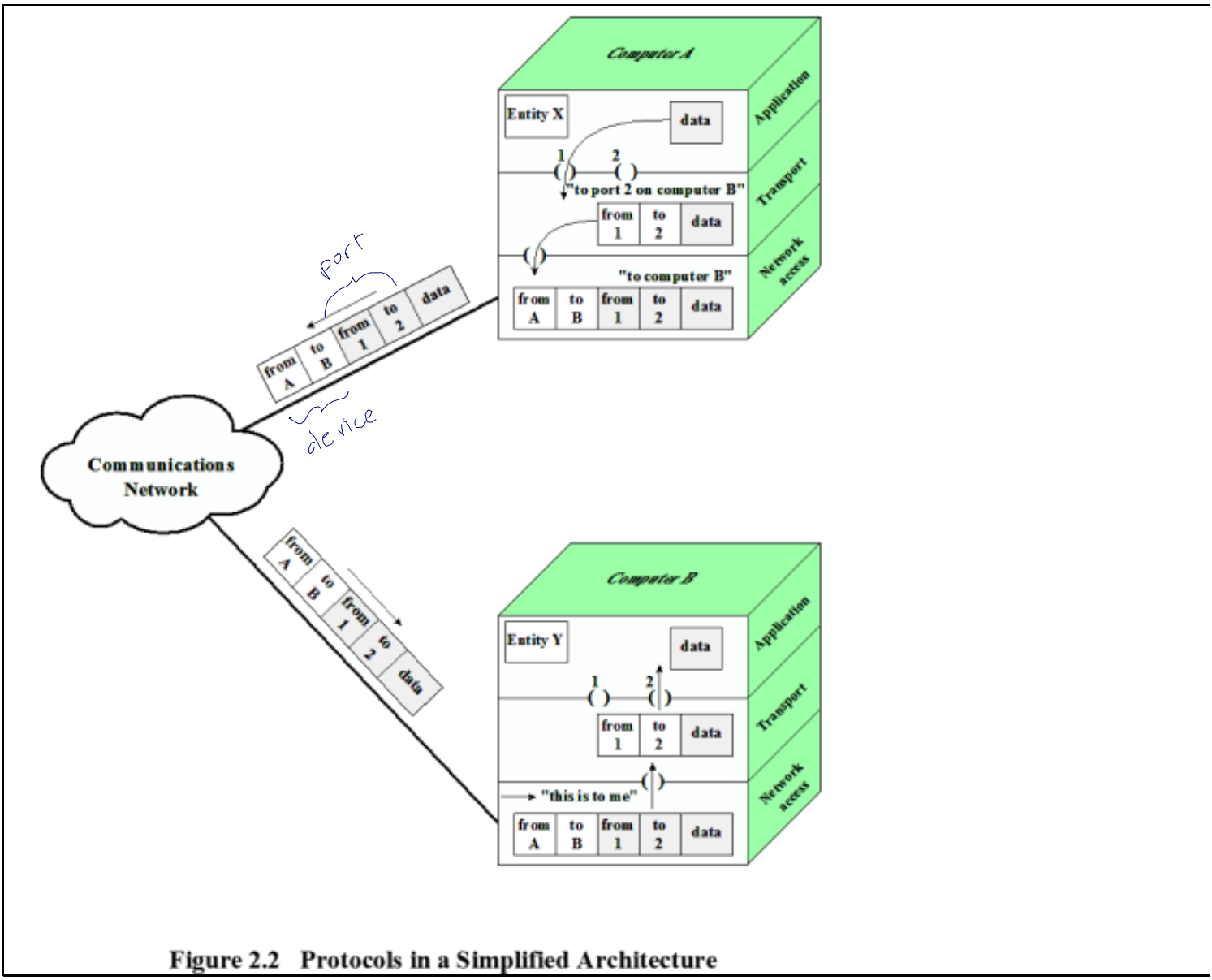


Figure 2.2 Protocols in a Simplified Architecture

Let us trace a simple operation. Suppose that an application, associated with port 1 at computer A, wishes to send a message to another application, associated with port 2 at computer B. The application at A hands the message over to its transport layer with instructions to send it to port 2 on computer B. The transport layer hands the message over to the network access layer, which instructs the network to send the message to computer B. Note that the network need not be told the identity of the destination port. All that it needs to know is that the data are intended for computer B.

To control this operation, control information, as well as user data, must be transmitted, as suggested in Figure 2.2. Let us say that the sending application generates a block of data and passes this to the transport layer. The transport layer may break this block into two smaller pieces for convenience, as discussed subsequently.

To each of these pieces the transport layer appends a transport header , containing protocol control information. The addition of control information to data is referred to as encapsulation . The combination of data from the next higher layer and control information is known as a protocol data unit (PDU) ; in this case, it is referred to as a transport PDU. Transport PDUs are typically called segments . The header in each segment contains control information to be used by the peer transport protocol at computer B. Examples of items that may be stored in this header include the following:

• Source port: This indicates the application that sent the data.

• Destination port: When the destination transport layer receives the segment, it must know to which application the data are to be delivered.

• Sequence number: Because the transport protocol is sending a sequence of segments, it numbers them sequentially so that if they arrive out of order, the destination transport entity may reorder them.

• Error-detection code: The sending transport entity may include a code that is a function of the contents of the segment. The receiving transport protocol performs the same calculation and compares the result with the incoming code. A discrepancy results if there has been some error in transmission. In that case, the receiver can discard the segment and take corrective action. This code is also referred to as a checksum or frame check sequence.

The next step is for the transport layer to hand each segment over to the network layer, with instructions to transmit it to the destination computer. To satisfy this request, the network access protocol must present the data to the network with a request for transmission. As before, this operation requires the use of control information. In this case, the network access protocol (NAP) appends a network access header to the data it receives from the transport layer, creating a network access PDU, typically called a packet . Examples of the items that may be stored in the header include the following:

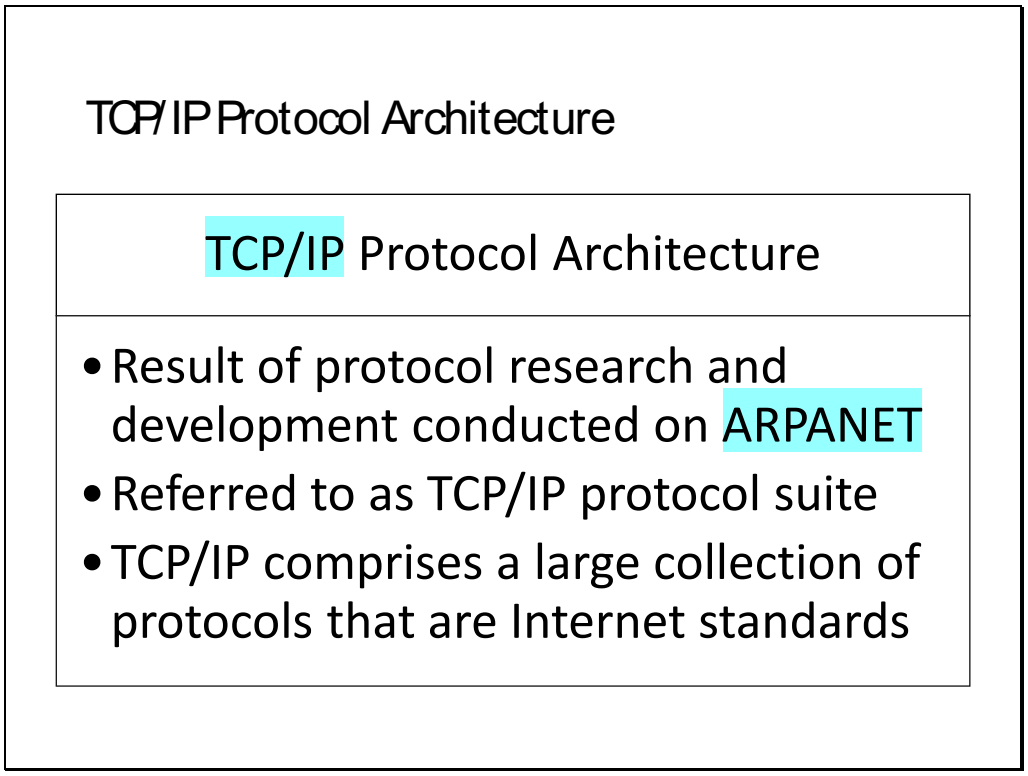
• Source computer address: Indicates the source of this packet.

• Destination computer address: The network must know to which computer on the network the data are to be delivered.

• Facilities requests: The network access protocol might want the network to make use of certain facilities, such as priority.

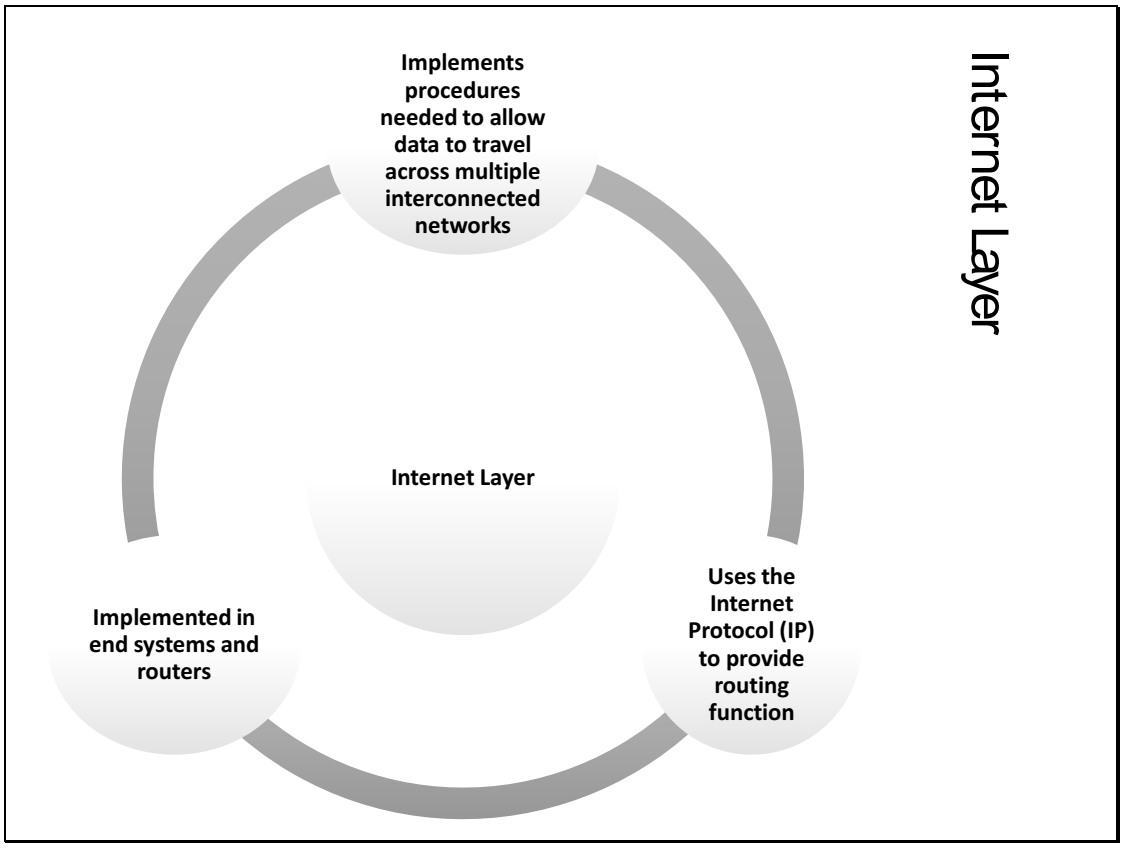
Note that the transport header is not “visible” at the network access layer; the network access layer is not concerned with the contents of the transport segment.

The network accepts the network packet from A and delivers it to B. The network access module in B receives the packet, strips off the packet header, and transfers the enclosed transport segment to B’s transport layer module. The transport layer examines the segment header and, on the basis of the port field in the header, delivers the enclosed record to the appropriate application, in this case the file transfer module in B.



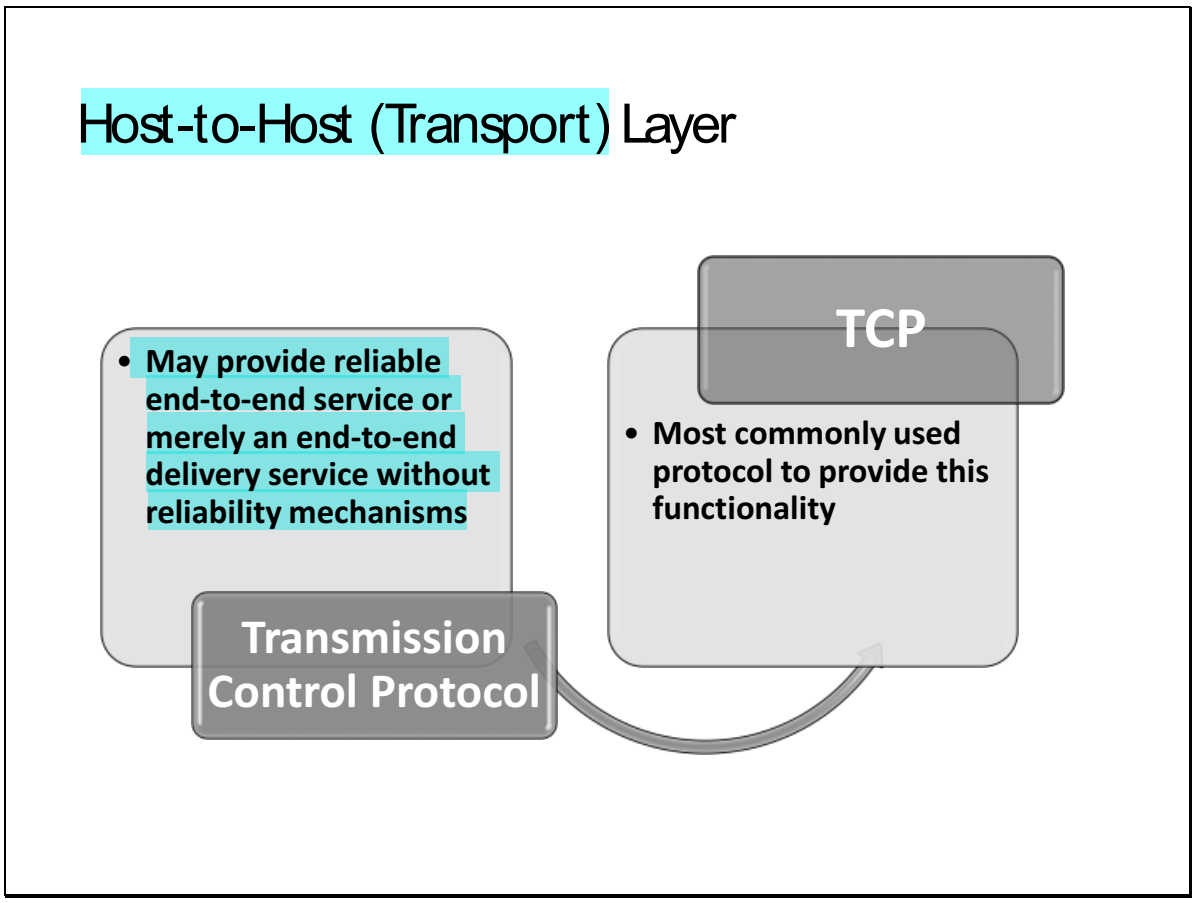
TCP/IP is a result of protocol research and development conducted on the experimental packet-switched network, ARPANET, funded by the Defense Advanced Research Projects Agency (DARPA), and is generally referred to as the TCP/IP protocol suite. This protocol suite consists of a large collection of protocols that have been issued as Internet standards by the Internet Activities Board (IAB).

Slide 23



In those cases where two devices are attached to different networks, procedures are needed to allow data to traverse multiple interconnected networks. This is the function of the internet layer . The Internet Protocol (IP) is used at this layer to provide the routing function across multiple networks. This protocol is implemented not only in the end systems but also in routers . A router is a processor that connects two networks and whose primary function is to relay data from one network to the other on its route from the source to the destination end system.

Slide 24

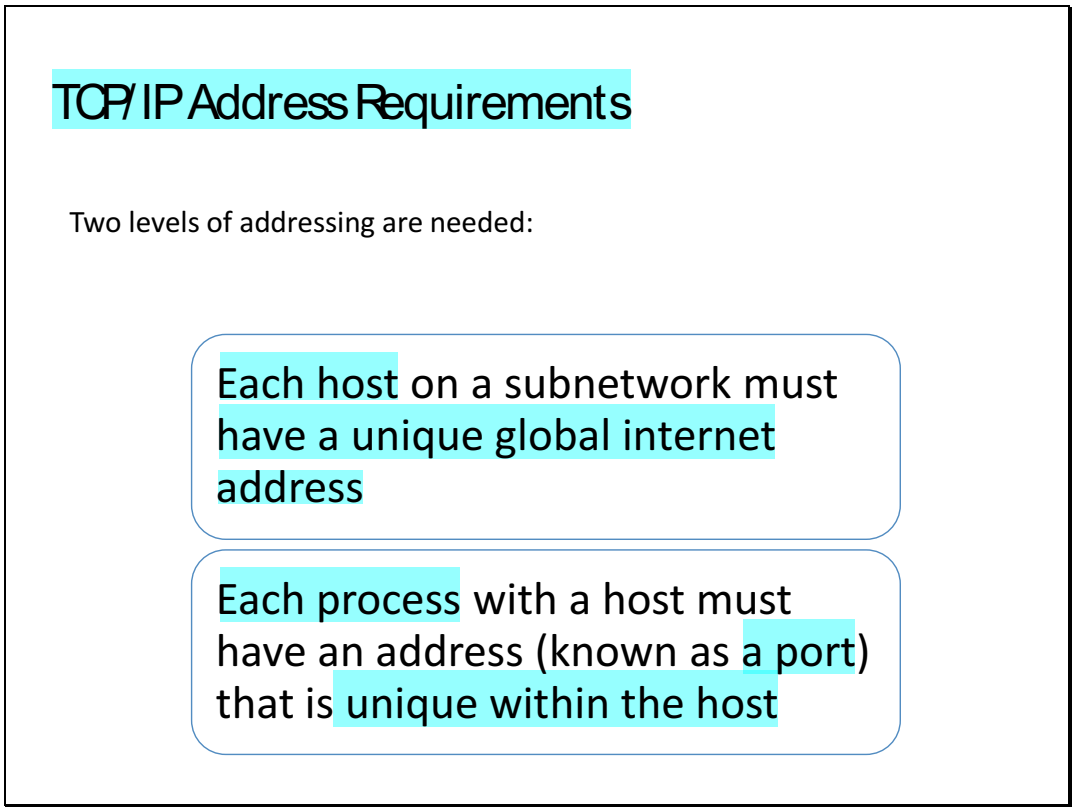


The host-to-host layer , or transport layer , may provide reliable end-to-end service, as discussed in Section 2.2, or merely an end-to-end delivery service without reliability mechanisms. The Transmission Control Protocol (TCP) is the most commonly used protocol to provide this functionality.

Application Layer

- Contains the logic needed to support the various user applications
- A separate module is needed for each different type of application that is peculiar to that application

Finally, the application layer contains the logic needed to support the various user applications. For each different type of application, such as file transfer, a separate module is needed that is peculiar to that application.



As is mentioned in Section 2.2, every entity in the overall system must have a unique address. Each host on a subnetwork must have a unique global internet address; this allows the data to be delivered to the proper host. Each process with a host must have an address that is unique within the host; this allows the host-to-host protocol (TCP) to deliver data to the proper process. These latter addresses are known as ports .

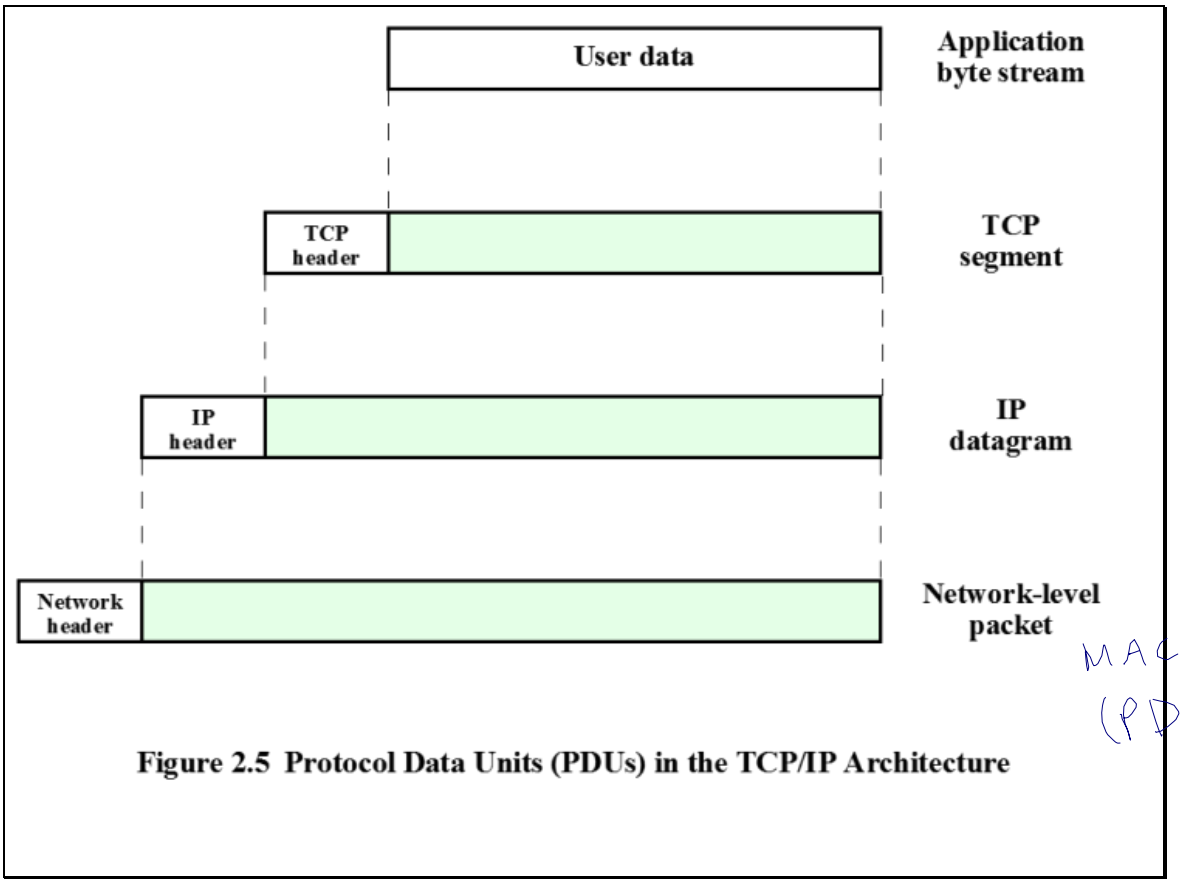


Figure 2.5 Protocol Data Units (PDUs) in the TCP/IP Architecture

Let us trace a simple operation. Suppose that a process, associated with port 3 at host A, wishes to send a message to another process, associated with port 2 at host B. The process at A hands the message down to TCP with instructions to send it to host B, port 2. TCP hands the message down to IP with instructions to send it to host B. Note that IP need not be told the identity of the destination port. All it needs to know is that the data are intended for host B. Next, IP hands the message down to the network access layer (e.g., Ethernet logic) with instructions to send it to router J (the first hop on the way to B).

To control this operation, control information, as well as user data, must be transmitted, as suggested in Figure 2.5. Let us say that the sending process generates a block of data and passes this to TCP. TCP may break this block into smaller pieces to make it more manageable. To each of these pieces, TCP appends control information known as the TCP header, forming a TCP segment . The control information is to be used by the peer TCP entity at host B. Examples of items in this header include:

• Destination port: When the TCP entity at B receives the segment, it must know to whom the data are to be delivered.

• Sequence number: TCP numbers the segments that it sends to a particular

destination port sequentially, so that if they arrive out of order, the TCP entity at B can reorder them.

• Checksum: The sending TCP includes a code that is a function of the contents of the remainder of the segment. The receiving TCP performs the same calculation and compares the result with the incoming code. A discrepancy results if there has been some error in transmission.

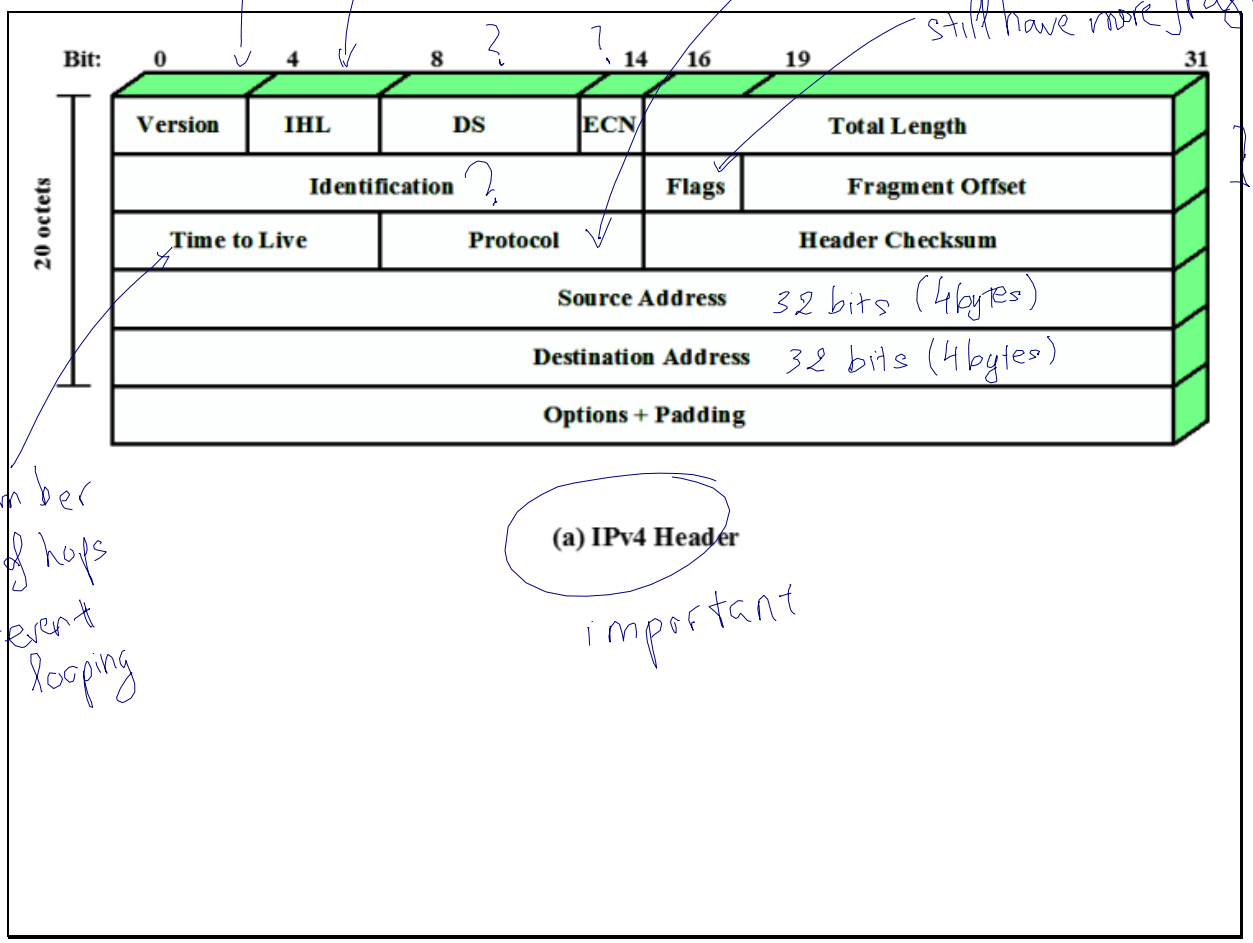
Next, TCP hands each segment over to IP, with instructions to transmit it to B. These segments must be transmitted across one or more subnetworks and relayed through one or more intermediate routers. This operation, too, requires the use of control information. Thus IP appends a header of control information to each segment to form an IP datagram . An example of an item stored in the IP header is the destination host address (in this example, B).

Finally, each IP datagram is presented to the network access layer for transmission across the first subnetwork in its journey to the destination. The network access layer appends its own header, creating a packet, or frame. The packet is transmitted across the subnetwork to router J. The packet header contains the information that the subnetwork needs to transfer the data across the subnetwork.

At router J, the packet header is stripped off and the IP header is examined. On the basis of the destination address information in the IP header, the IP module in the router directs the datagram out across subnetwork 2 to B. To do this, the datagram is again augmented with a network access header.

When the data are received at B, the reverse process occurs. At each layer, the corresponding header is removed, and the remainder is passed on to the next higher layer, until the original user data are delivered to the destination process.

Slide 28



For decades, the keystone of the TCP/IP architecture has been IPv4, generally referred to as IP. Figure 2.7a shows the IP header format, which is a minimum of 20 octets, or 160 bits. The header, together with the segment from the transport layer, forms an IP-level PDU referred to as an IP datagram or an IP packet. The header includes 32-bit source and destination addresses. The Header Checksum field is used to detect errors in the header to avoid misdelivery. The Protocol field indicates which higher-layer protocol is using IP. The ID, Flags, and Fragment Offset fields are used in the fragmentation and reassembly process. Chapter 14 provides more details.

Slide 29

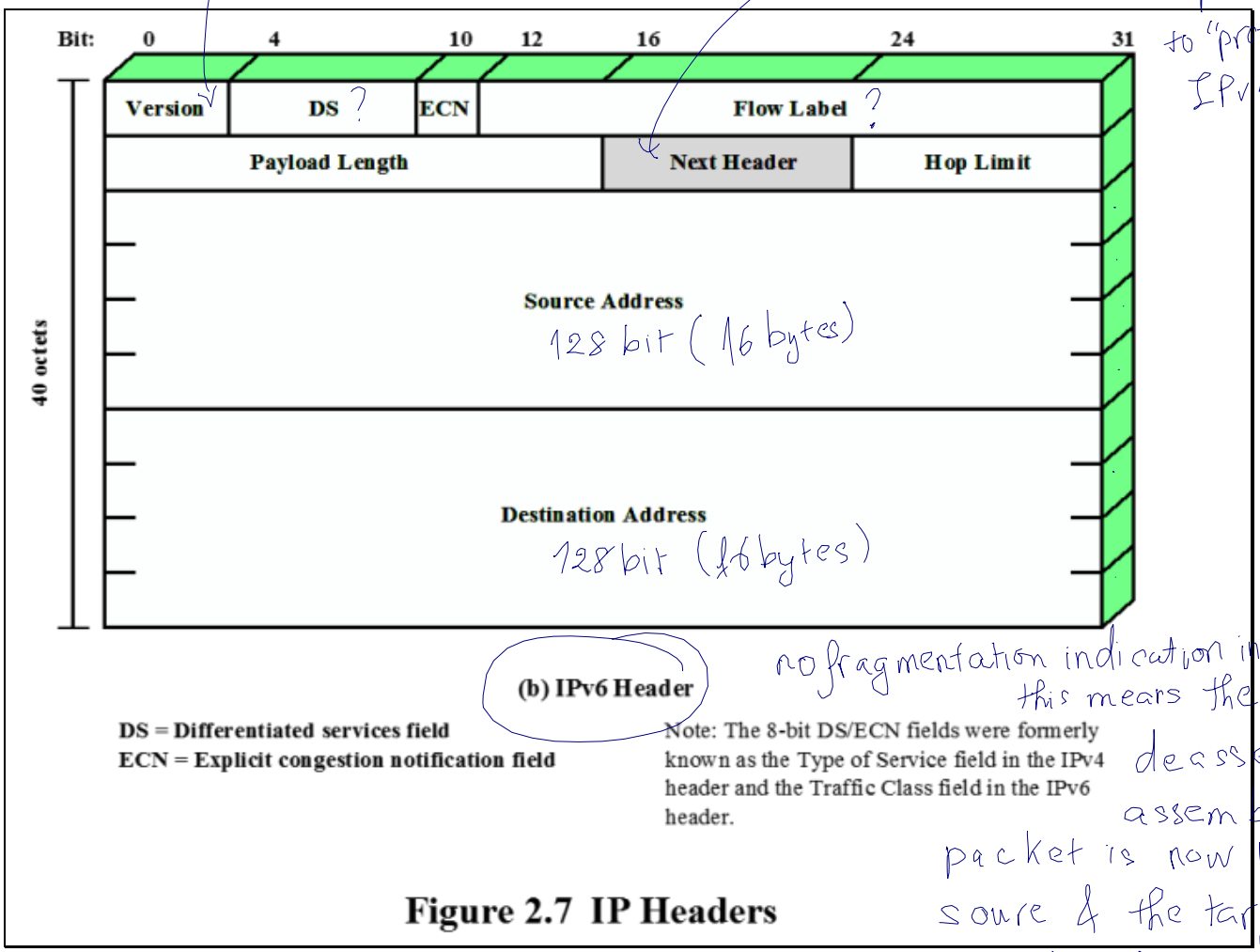


Figure 2.7 IP Headers

In 1995, the Internet Engineering Task Force (IETF), which develops protocol standards for the Internet, issued a specification for a next-generation IP, known then as IPng. This specification was turned into a standard in 1996 known as IPv6. IPv6 provides a number of functional enhancements over the existing IP, designed to accommodate the higher speeds of today's networks and the mix of data streams, including graphic and video, that are becoming more prevalent. But the driving force behind the development of the new protocol was the need for more addresses. IPv4 uses a 32-bit address to specify a source or destination. With the explosive growth of the Internet and of private networks attached to the Internet, this address length became insufficient to accommodate all systems needing addresses. As Figure 2.7b shows, IPv6 includes 128-bit source and destination address fields.

Ultimately, all installations using TCP/IP are expected to migrate from the current IP to IPv6, but this process will take many years, if not decades.

Transmission Control Protocol (TCP)

- TCP is the transport layer protocol for most applications
- TCP provides a reliable connection for transfer of data between applications
- A TCP segment is the basic protocol unit
- TCP tracks segments between entities for duration of each connection

For most applications running as part of the TCP/IP architecture, the transport layer protocol is TCP. TCP provides a reliable connection for the transfer of data between applications. A connection is simply a temporary logical association between two entities in different systems. A logical connection refers to a given pair of port values. For the duration of the connection, each entity keeps track of TCP segments coming and going to the other entity, in order to regulate the flow of segments and to recover from lost or damaged segments.

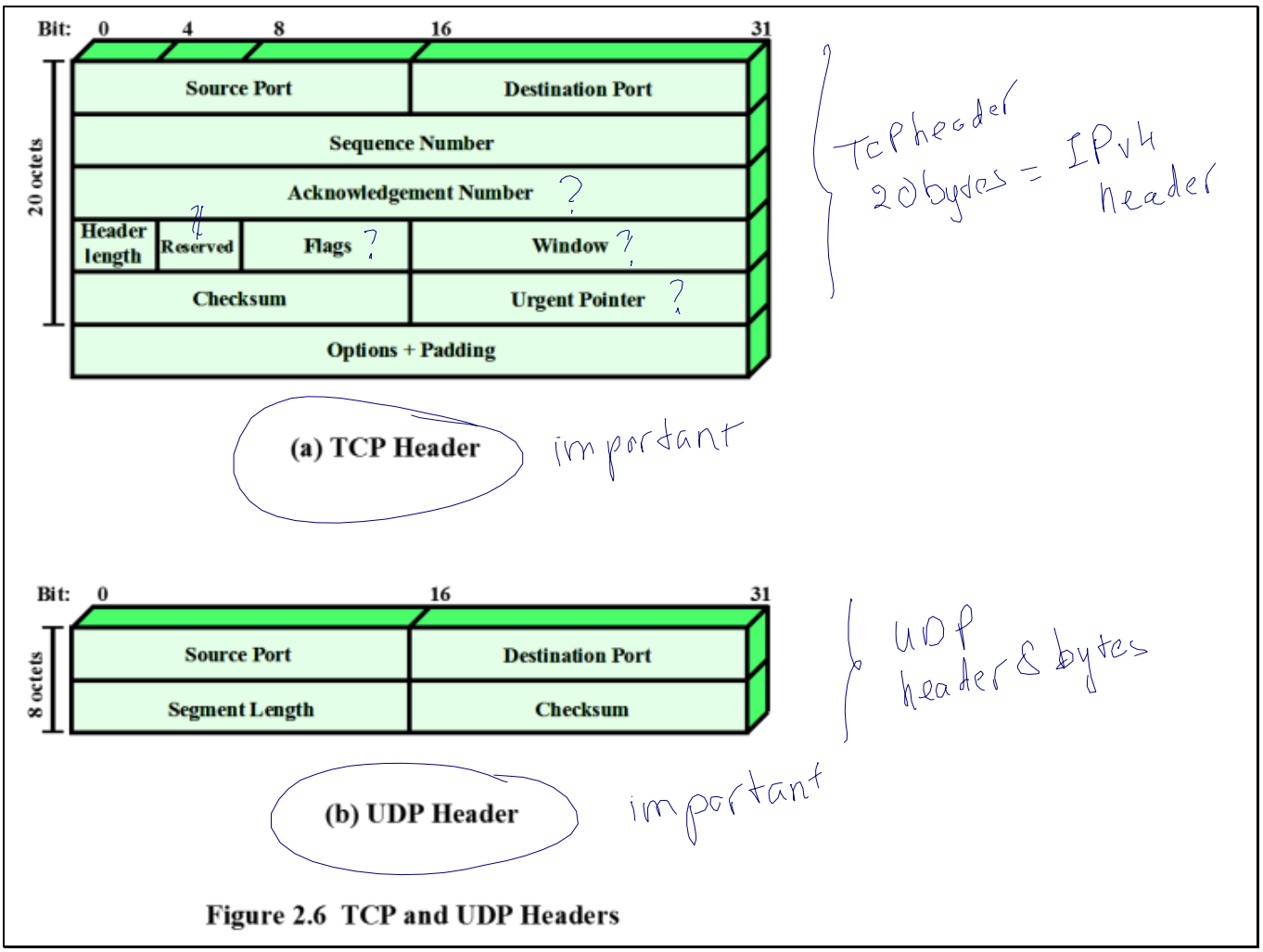


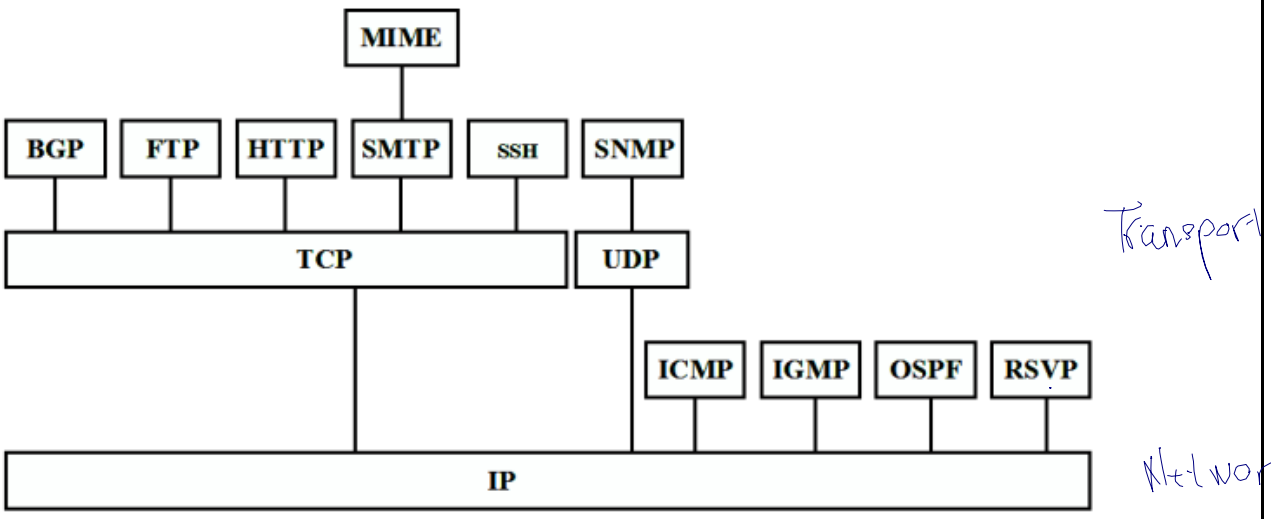
Figure 2.6 TCP and UDP Headers

Figure 2.6a shows the header format for TCP, which is a minimum of 20 octets, or 160 bits. The Source Port and Destination Port fields identify the applications at the source and destination systems that are using this connection. The Sequence Number, Acknowledgment Number, and Window fields provide flow control and error control. The checksum is a 16-bit frame check sequence used to detect errors in the TCP segment. Chapter 15 provides more details.

User Datagram Protocol (UDP)

- Alternative to TCP
- Does not guarantee delivery, preservation of sequence, or protection against duplication
- Enables a procedure to send messages to other procedures with a minimum of protocol mechanism
- Adds port addressing capability to IP
- Used with Simple Network Management Protocol (SNMP)
- Includes a checksum to verify that no error occurs in the data

In addition to TCP, there is one other transport-level protocol that is in common use as part of the TCP/IP protocol suite: the User Datagram Protocol (UDP) . UDP does not guarantee delivery, preservation of sequence, or protection against duplication. UDP enables a procedure to send messages to other procedures with a minimum of protocol mechanism. Some transaction-oriented applications make use of UDP; one example is SNMP (Simple Network Management Protocol), the standard network management protocol for TCP/IP networks. Because it is connectionless, UDP has very little to do. Essentially, it adds a port addressing capability to IP. This is best seen by examining the UDP header, shown in Figure 2.6b. UDP also includes a checksum to verify that no error occurs in the data; the use of the checksum is optional.



BGP = Border Gateway Protocol
FTP = File Transfer Protocol
HTTP = Hypertext Transfer Protocol
ICMP = Internet Control Message Protocol
IGMP = Internet Group Management Protocol
IP = Internet Protocol
MIME = Multipurpose Internet Mail Extension
OSPF = Open Shortest Path First
RSVP = Resource ReSerVation Protocol
SMTP = Simple Mail Transfer Protocol
SNMP = Simple Network Management Protocol
SSH = Secure Shell
TCP = Transmission Control Protocol
UDP = User Datagram Protocol

Figure 2.8 Some Protocols in the TCP/IP Protocol Suite

Each layer in the TCP/IP suite interacts with its immediate adjacent layers. At the source, the application layer makes use of the services of the end-to-end layer and provides data down to that layer. A similar relationship exists at the interface between the transport and internet layers and at the interface of the internet and network access layers. At the destination, each layer delivers data up to the next higher layer.

This use of each individual layer is not required by the architecture. As Figure 2.8 suggests, it is possible to develop applications that directly invoke the services of any one of the layers. Most applications require a reliable end-to-end protocol and thus make use of TCP. Some special-purpose applications do not need the services of TCP. Some of these applications, such as the Simple Network Management Protocol (SNMP), use an alternative end-to-end protocol known as the User Datagram Protocol (UDP); others may make use of IP directly. Applications that do not involve internetworking and that do not need TCP have been developed to invoke the network access layer directly.

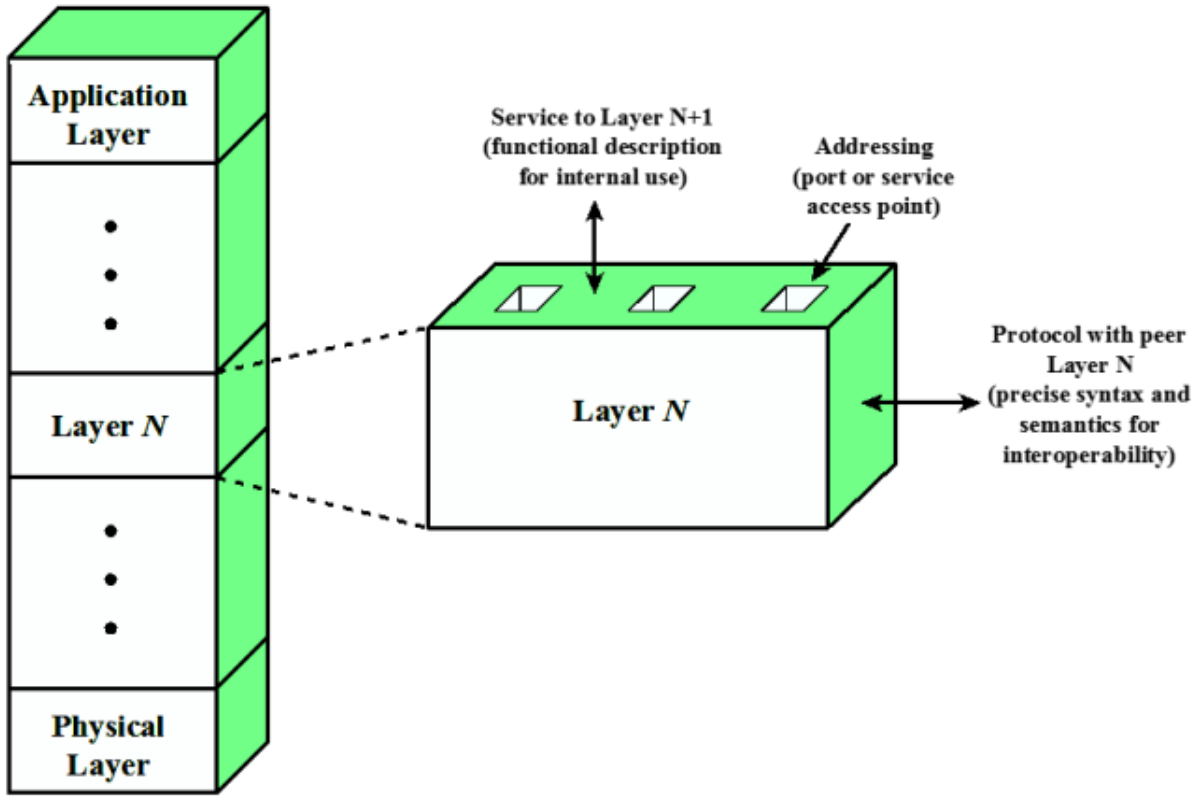


Figure 2.9 A Protocol Architecture as a Framework for Standardization

A protocol architecture, such as the TCP/IP architecture or OSI, provides a framework for standardization. Within the model, one or more protocol standards can be developed at each layer. The model defines in general terms the functions to be performed at that layer and facilitates the standards-making process in two ways:

- Because the functions of each layer are well defined, standards can be developed independently and simultaneously for each layer. This speeds up the standards-making process.

- Because the boundaries between layers are well defined, changes in standards in one layer need not affect already existing software in another layer. This makes it easier to introduce new standards.

Figure 2.9 illustrates the use of a protocol architecture as such a framework. The overall communications function is decomposed into a number of distinct layers. That is, the overall function is broken up into a number of modules,

making the interfaces between modules as simple as possible. In addition, the design principle of information hiding is used: Lower layers are concerned with greater levels of detail; upper layers are independent of these details. Each layer provides services to the next higher layer and implements a protocol to the peer layer in other systems.

Figure 2.9 also shows more specifically the nature of the standardization required at each layer. Three elements are key:

• Protocol specification: Two entities at the same layer in different systems cooperate and interact by means of a protocol. Because two different open systems are involved, the protocol must be specified precisely. This includes the format of the protocol data units exchanged, the semantics of all fields, and the allowable sequence of PDUs.

• Service definition: In addition to the protocol or protocols that operate at a given layer, standards are needed for the services that each layer provides to the next higher layer. Typically, the definition of services is equivalent to a functional description that defines what services are provided, but not how the services are to be provided.

• Addressing: Each layer provides services to entities at the next higher layer. These entities are referenced by means of a port, or service access point (SAP) . Thus, a network service access point (NSAP) indicates a transport entity that is a user of the network service.

The need to provide a precise protocol specification for open systems is self-evident. The other two items listed warrant further comment. With respect to service definitions, the motivation for providing only a functional definition is as follows. First, the interaction between two adjacent layers takes place within the confines of a single open system and is not the concern of any other open system. Thus, as long as peer layers in different systems provide the same services to their next higher layers, the details of how the services are provided may differ from one system to another without loss of interoperability. Second, it will usually be the case that adjacent layers are implemented on the same processor. In that case, we would like to leave the system programmer free to exploit the hardware and operating system to provide an interface that is as efficient as possible.

With respect to addressing, the use of an address mechanism at each layer, implemented as a service access point, allows each layer to multiplex multiple users from the next higher layer. Multiplexing may not occur at each layer, but the model allows for that possibility.

Service Primitives and Parameters

- Services between adjacent layers
- Expressed as:
  - Primitives
    - Specify the function to be performed
  - Parameters
    - Used to pass data and control information

The services between adjacent layers in a protocol architecture are expressed in terms of primitives and parameters. A primitive specifies the function to be performed, and the parameters are used to pass data and control information. The actual form of a primitive is implementation dependent. An example is a procedure call.

Table 2.1
Service Primitive Types

REQUEST	A primitive issued by a service user to invoke some service and to pass the parameters needed to specify fully the requested service
INDICATION	A primitive issued by a service provider either to 1. indicate that a procedure has been invoked by the peer service user on the connection and to provide the associated parameters, or 2. notify the service user of a provider-initiated action
RESPONSE	A primitive issued by a service user to acknowledge or complete some procedure previously invoked by an indication to that user
CONFIRM	A primitive issued by a service provider to acknowledge or complete some procedure previously invoked by a request by the service user

Four types of primitives are used in standards to define the interaction between adjacent layers in the architecture. These are defined in Table 2.1.

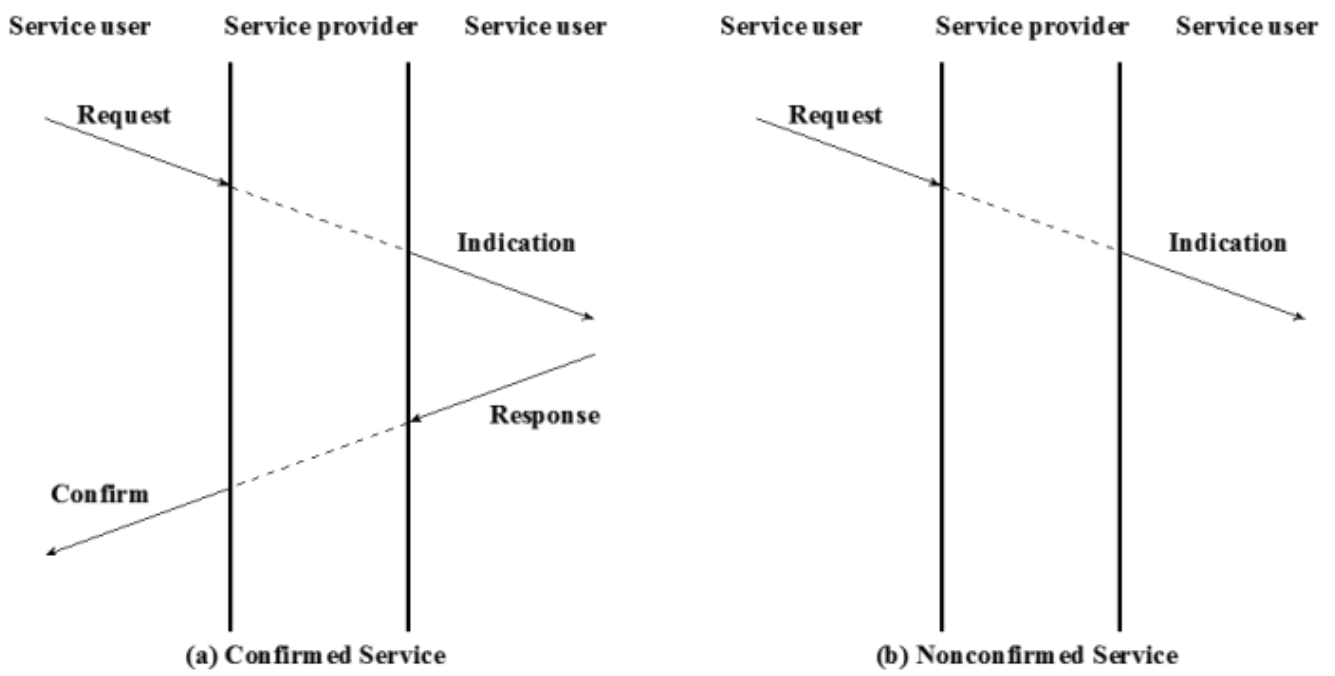
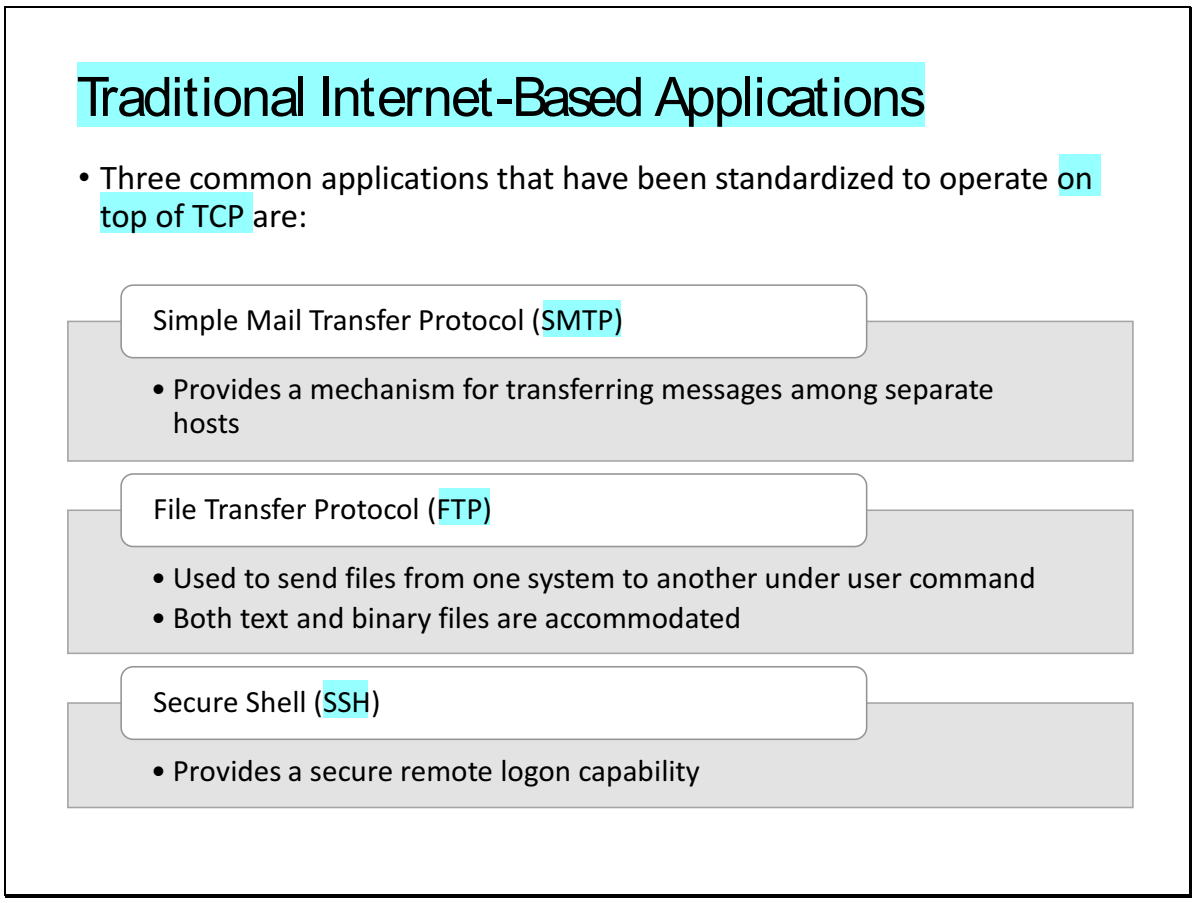


Figure 2.10 Time Sequence Diagrams for Service Primitives

The layout of Figure 2.10a suggests the time ordering of these events.

This sequence of events is referred to as a confirmed service , as the initiator receives confirmation that the requested service has had the desired effect at the other end. If only request and indication primitives are involved (corresponding to steps 1 through 3), then the service dialog is a nonconfirmed service ; the initiator receives no confirmation that the requested action has taken place (Figure 2.10b).



A number of applications have been standardized to operate on top of TCP. We mention three of the most common here.

The Simple Mail Transfer Protocol (SMTP) provides a basic electronic mail transport facility. It provides a mechanism for transferring messages among separate hosts. Features of SMTP include mailing lists, return receipts, and forwarding. SMTP does not specify the way in which messages are to be created; some local editing or native electronic mail facility is required. Once a message is created, SMTP accepts the message and makes use of TCP to send it to an SMTP module on another host. The target SMTP module will make use of a local electronic mail package to store the incoming message in a user's mailbox.

The File Transfer Protocol (FTP) is used to send files from one system to another under user command. Both text and binary files are accommodated, and the protocol provides features for controlling user access. When a user wishes to engage in file transfer, FTP sets up a TCP connection to the target system for the exchange of control messages. This connection allows user ID and password to be transmitted and allows the user to specify the file and file actions desired. Once a file transfer is approved, a second TCP connection is set up for the data transfer. The file is transferred over the data connection, without the overhead of any headers

or control information at the application level. When the transfer is complete, the control connection is used to signal the completion and to accept new file transfer commands.

SSH (Secure Shell) provides a secure remote logon capability, which enables a user at a terminal or personal computer to log on to a remote computer and function as if directly connected to that computer. SSH also supports file transfer between the local host and a remote server. SSH enables the user and the remote server to authenticate each other; it also encrypts all traffic in both directions. SSH traffic is carried on a TCP connection.

Sockets Programming

- Concept was developed in the 1980s in the UNIX environment as the Berkeley Sockets Interface
  - De facto standard application programming interface (API)
  - Basis for Window Sockets (WinSock)
- Enables communication between a client and server process
- May be connection oriented or connectionless

The concept of sockets and sockets programming was developed in the 1980s in the UNIX environment as the Berkeley Sockets Interface. In essence, a socket enables communication between a client and server process and may be either connection oriented or connectionless. A socket can be considered an end point in a communication. A client socket in one computer uses an address to call a server socket on another computer. Once the appropriate sockets are engaged, the two computers can exchange data.

Typically, computers with server sockets keep a TCP or UDP port open, ready for unscheduled incoming calls. The client typically determines the socket identification of the desired server by finding it in a Domain Name System (DNS) database. Once a connection is made, the server switches the dialogue to a different port number to free up the main port number for additional incoming calls.

Internet applications, such as TELNET and remote login (rlogin), make use of sockets, with the details hidden from the user. However, sockets can be constructed from within a program (in a language such as C, Java, or Python), enabling the programmer to easily support networking functions and applications. The sockets programming mechanism includes sufficient semantics to permit unrelated processes on different hosts to communicate.

The Berkeley Sockets Interface is the de facto standard application programming interface (API) for developing networking applications, spanning a wide range of operating systems. Windows Sockets (WinSock) is based on the Berkeley specification. The Sockets API provides generic access to interprocess communications Services. Thus, the sockets capability is ideally suited for students to learn the principles of protocols and distributed applications by hands-on program development.

The Socket

- Formed by the concatenation of a port value and an IP address
  - Unique throughout the Internet
- Used to define an API
  - Generic communication interface for writing programs that use TCP or UDP
- Stream sockets
  - All blocks of data sent between a pair of sockets are guaranteed for delivery and arrive in the order that they were sent
- Datagram sockets
  - Delivery is not guaranteed, nor is order necessarily preserved
- Raw sockets
  - Allow direct access to lower-layer protocols

Recall that each TCP and UDP header includes Source Port and Destination Port fields (Figure 2.6). These port values identify the respective users (applications) of the two TCP or UDP entities. Also, each IPv4 and IPv6 header includes Source Address and Destination Address fields (Figure 2.7); these IP addresses identify the respective host systems. The concatenation of a port value and an IP address forms a socket , which is unique throughout the Internet. Thus, in Figure 2.4, the combination of the IP address for host B and the port number for application X uniquely identifies the socket location of application X in host B. As the figure indicates, an application may have multiple socket addresses, one for each port into the application.

The socket is used to define an API, which is a generic communication interface for writing programs that use TCP or UDP. In practice, when used as an API, a socket is identified by the triple (protocol, local address, local process). The local address is an IP address and the local process is a port number. Because port numbers are unique within a system, the port number implies the protocol (TCP or UDP). However, for clarity and ease of implementation, sockets used for an API include the protocol as well as the IP address and port number in defining a unique socket.

Corresponding to the two protocols, the Sockets API recognizes two types of sockets: stream sockets and datagram sockets. Stream sockets make use of TCP, which provides a connection-oriented reliable data transfer. Therefore, with stream sockets, all blocks of data sent between a pair of sockets are guaranteed for delivery and arrive in the order that they were sent. Datagram sockets make use of UDP, which does not provide the connection-oriented features of TCP. Therefore, with datagram sockets, delivery is not guaranteed, nor is order necessarily preserved.

There is a third type of socket provided by the Sockets API: raw sockets. Raw sockets allow direct access to lower-layer protocols, such as IP.

Format	Function	Parameters
socket( )	Initialize a socket	**domain** Protocol family of the socket to be created (AF_UNIX, AF_INET, AF_INET6) **type** Type of socket to be opened (stream, datagram, raw) **protocol** Protocol to be used on socket (UDP, TCP, ICMP)
bind( )	Bind a socket to a port address	**sockfd** Socket to be bound to the port address **localaddress** Socket address to which the socket is bound **addresslength** Length of the socket address structure
listen( )	Listen on a socket for inbound connections	**sockfd** Socket on which the application is to listen **queuesize** Number of inbound requests that can be queued at any time
accept( )	Accept an inbound connection	**sockfd** Socket on which the connection is to be accepted **remoteaddress** Remote socket address from which the connection was initiated **addresslength** Length of the socket address structure
connect( )	Connect outbound to a server	**sockfd** Socket on which the connection is to be opened **remoteaddress** Remote socket address to which the connection is to be opened **addresslength** Length of the socket address structure
send( ) recv( ) read( ) write( )	Send and receive data on a stream socket (either send/recv or read/write can be used)	**sockfd** Socket across which the data will be sent or read **data** Data to be sent, or buffer into which the read data will be placed **datalength** Length of the data to be written, or amount of data to be read
sendto( ) recvfrom( )	Send and receive data on a datagram socket	**sockfd** Socket across which the data will be sent or read **data** Data to be sent, or buffer into which the read data will be placed **datalength** Length of the data to be written, or amount of data to be read
close( )	Close a socket	**sockfd** Socket which is to be closed

Table 2.4

Core Socket Functions

(Table can be found on page 54 in textbook)

This subsection summarizes the key system calls. Table 2.4 lists the core Socket functions.

The first step in using Sockets is to create a new socket using the socket() command. This command includes three parameters. The domain parameter refers to the area where the communicating processes exist.

Type specifies whether this is a stream or datagram socket, and protocol specifies either TCP or UDP. The reason that both type and protocol need to be specified is to allow additional transport-level protocols to be included in a future implementation. Thus, there might be more than one datagram-style transport protocol or more than one connection-oriented transport protocol. The socket() command returns an integer result that identifies this socket; it is similar to a UNIX file descriptor. The exact socket data structure depends on the implementation. It includes the source port and IP address and, if a connection is open or pending, the destination port and IP address and various options and parameters associated with the connection.

After a socket is created, it must have an address to listen to. The bind() function binds a socket to a socket address.

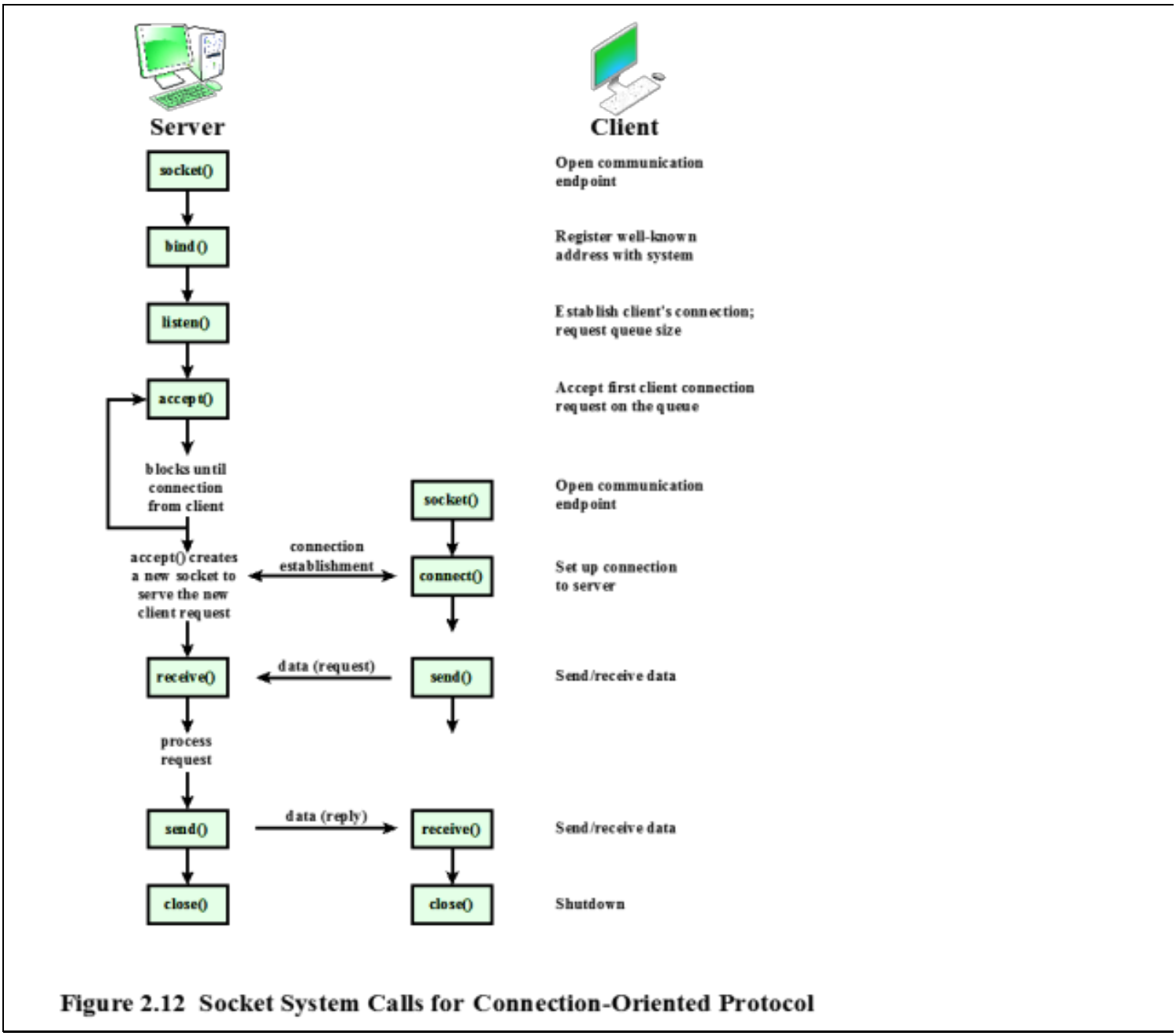


Figure 2.12 Socket System Calls for Connection-Oriented Protocol

Figure 2.12 shows the interaction of the clients and server sides in setting up, using, and terminating a connection.

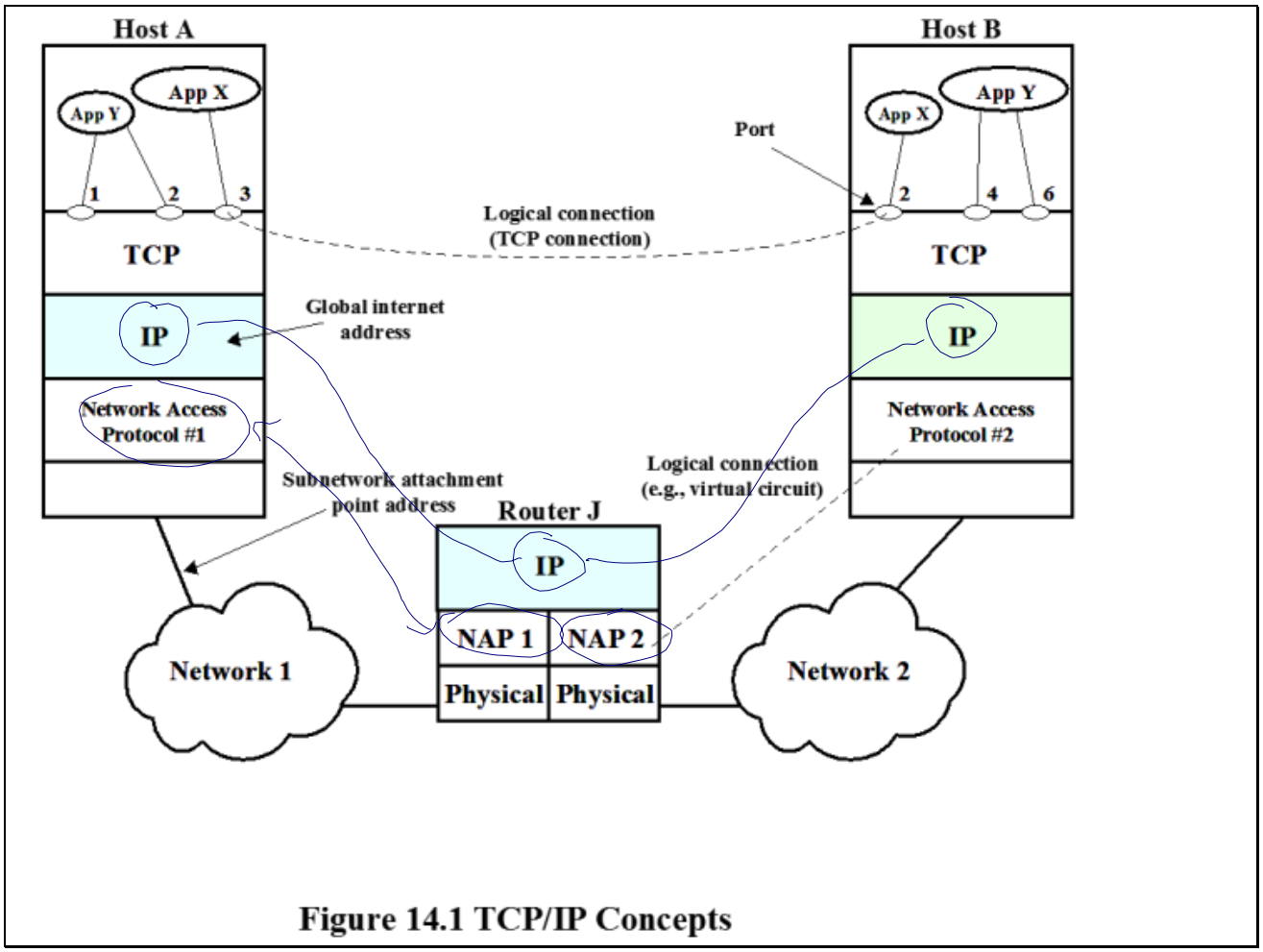


Figure 14.1 TCP/IP Concepts

The overall requirements for an internetworking facility are as follows (we refer to Figure 14.1, which repeats Figure 2.4, as an example throughout):

1. Provide a link between networks. At minimum, a physical and link control connection is needed. (Router J has physical links to N1 and N2, and on each link there is a data link protocol.)

2. Provide for the routing and delivery of data between processes on different networks. (Application X on host A exchanges data with application X on host B.)

3. Provide an accounting service that keeps track of the use of the various networks and routers and maintains status information.

4. Provide the services just listed in such a way as not to require modifications to the networking architecture of any of the constituent networks. This means that the internetworking facility must accommodate a number of differences among networks. These include:

• Different addressing schemes: The networks may use different endpoint names and addresses and directory maintenance schemes. Some form of global network addressing must be provided, as well as a directory service. (Hosts A and B and router J have globally unique IP addresses.)

• Different maximum packet size: Packets from one network may have to be broken up into smaller pieces for another. This process is referred to as fragmentation. (N1 and N2 may set different upper limits on packet sizes.)

• Different network access mechanisms: The network access mechanism between station and network may be different for stations on different networks. (e.g., N1 may be a frame relay network and N2 an Ethernet network.)

• Different timeouts: Typically, a connection-oriented transport service will await an acknowledgment until a timeout expires, at which time it will retransmit its block of data. In general, longer times are required for successful delivery across multiple networks. Internetwork timing procedures must allow successful transmission that avoids unnecessary retransmissions.

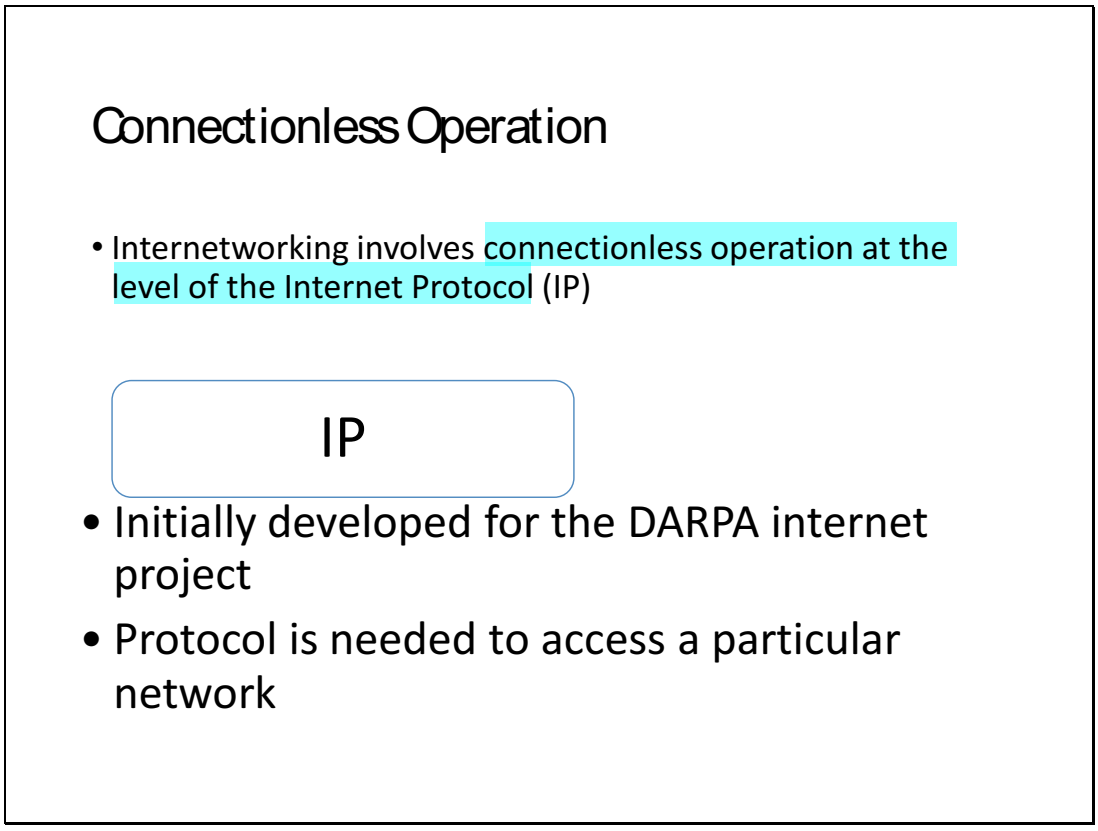
• Error recovery: Network procedures may provide anything from no error recovery up to reliable end-to-end (within the network) service. The internetwork service should not depend on nor be interfered with by the nature of the individual network's error recovery capability.

• Status reporting: Different networks report status and performance differently. Yet it must be possible for the internetworking facility to provide such information on internetworking activity to interested and authorized processes.

• Routing techniques: Intranetwork routing may depend on fault detection and congestion control techniques peculiar to each network. The internetworking facility must be able to coordinate these to route data adaptively between stations on different networks.

• User access control: Each network will have its own user access control technique (authorization for use of the network). These must be invoked by the internetwork facility as needed. Further, a separate internetwork access control technique may be required.

• Connection, connectionless: Individual networks may provide connection-oriented (e.g., virtual circuit) or connectionless (datagram) service. It may be desirable for the internetwork service not to depend on the nature of the connection service of the individual networks.



In virtually all implementations, internetworking involves connectionless operation at the level of the Internet Protocol. Whereas connection-oriented operation corresponds to the virtual circuit mechanism of a packet-switching network (Figure 9.12), connectionless-mode operation corresponds to the datagram mechanism of a packet-switching network (Figure 9.11). Each network protocol data unit is treated independently and routed from source ES to destination ES through a series of routers and networks. For each data unit transmitted by A, A makes a decision as to which router should receive the data unit. The data unit hops across the internet from one router to the next until it reaches the destination network. At each router, a routing decision is made (independently for each data unit) concerning the next hop. Thus, different data units may travel different routes between source and destination ES.

All ESs and routers share a common network-layer protocol known generically as the Internet Protocol. An Internet Protocol was initially developed for the DARPA internet project and published as RFC 791 and has become an Internet Standard. Below this Internet Protocol, a protocol is needed to access a particular network.

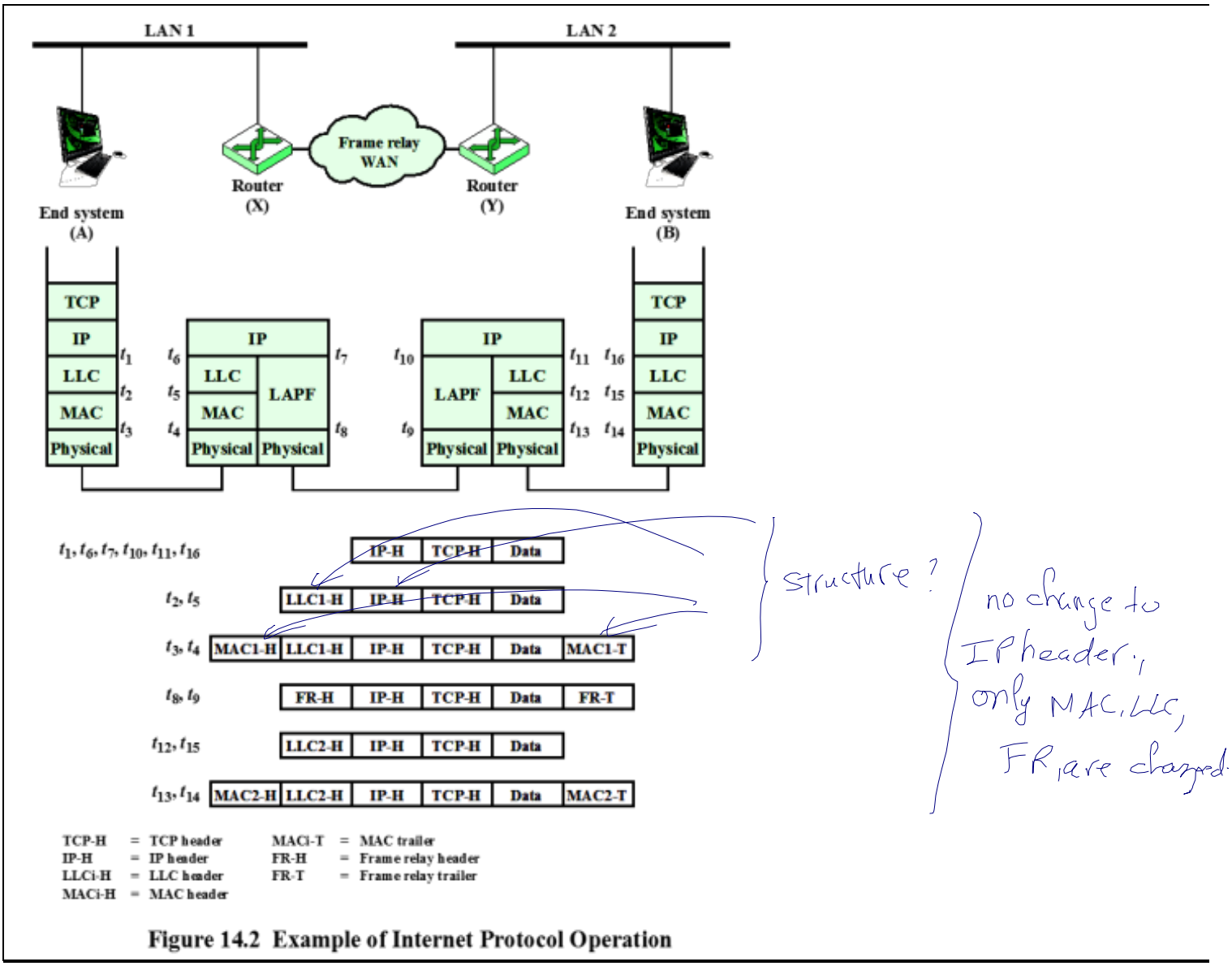


Figure 14.2 Example of Internet Protocol Operation

Thus, there are typically two protocols operating in each ES and router at the network layer: an upper sublayer that provides the internetworking function and a lower sublayer that provides network access.

Figure 14.2 depicts a typical example using IP, in which two LANs are interconnected by a frame relay WAN. The figure depicts the operation of the Internet Protocol for data exchange between host A on one LAN (network 1) and host B on another LAN (network 2) through the WAN. The figure shows the protocol architecture and format of the data unit at each stage. The end systems and routers must all share a common Internet Protocol. In addition, the end systems must share the same protocols above IP. The intermediate routers need only implement up through IP.

Connectionless Internetworking

- Connectionless internet facility is flexible
- IP provides a connectionless service between end systems
  - Advantages:
    - Is flexible
    - Can be made robust
    - Does not impose unnecessary overhead

IP provides a connectionless, or datagram, service between end systems. There are a number of advantages to this approach:

A connectionless internet facility is flexible. It can deal with a variety of networks, some of which are themselves connectionless. In essence, IP requires very little from the constituent networks.

A connectionless internet service can be made highly robust. This is basically the same argument made for a datagram network service versus a virtual circuit service.

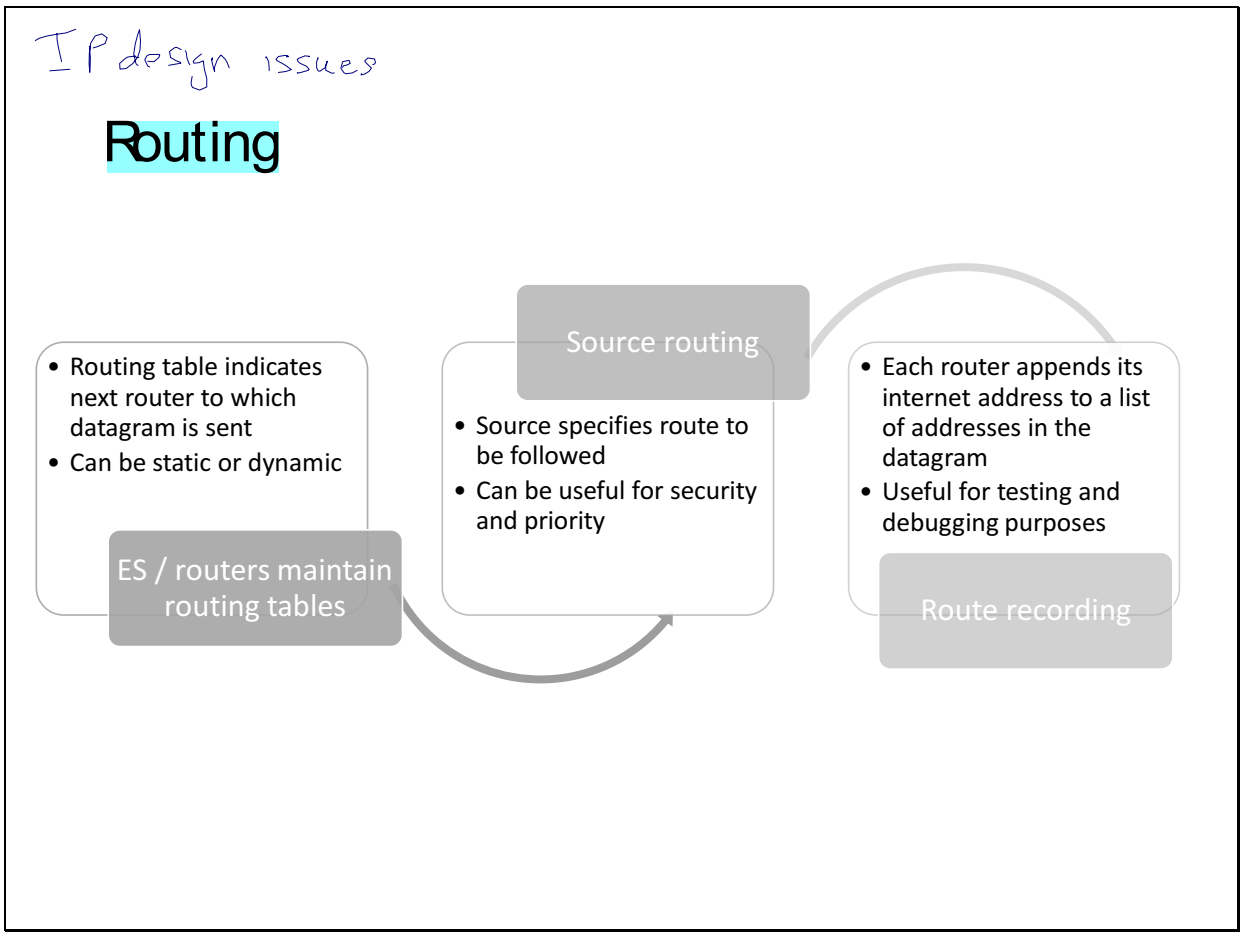
A connectionless internet service is best for connectionless transport protocols, because it does not impose unnecessary overhead.

IP Design Issues

- Routing
- Datagram lifetime
- Fragmentation and reassembly
- Error control
- Flow control

With that brief sketch of the operation of an IP-controlled internet, we now examine some design issues in greater detail:

Routing
Datagram lifetime
Fragmentation and reassembly
Error control
Flow control



For the purpose of routing, each end system and router maintains a routing table that lists, for each possible destination network, the next router to which the internet datagram should be sent.

The routing table may be static or dynamic. A static table, however, could contain alternate routes if a particular router is unavailable. A dynamic table is more flexible in responding to both error and congestion conditions. In the Internet, for example, when a router goes down, all of its neighbors will send out a status report, allowing other routers and stations to update their routing tables. A similar scheme can be used to control congestion. Congestion control is particularly important because of the mismatch in capacity between local and wide area networks. Chapter 19 discusses routing protocols.

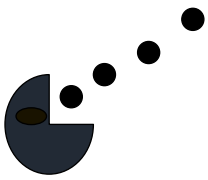
Routing tables may also be used to support other internetworking services, such as security and priority. For example, individual networks might be classified to handle data up to a given security classification. The routing mechanism must assure that data of a given security level are not allowed to pass through networks not cleared to handle such data.

Another routing technique is source routing. The source station specifies the route by including a sequential list of routers in the datagram. This, again, could be useful for security or priority requirements.

Finally, we mention a service related to routing: route recording. To record a route, each router appends its internet address to a list of addresses in the datagram. This feature is useful for testing and debugging purposes.

Datagram Lifetime

- If dynamic or alternate routing is used the potential exists for a datagram to loop indefinitely
  - Consumes resources
  - Transport protocol may need upper bound on lifetime of a datagram
    - Can mark datagram with lifetime
    - When lifetime expires, datagram is discarded



If dynamic or alternate routing is used, the potential exists for a datagram to loop indefinitely through the internet. This is undesirable for two reasons. First, an endlessly circulating datagram consumes resources. Second, we will see Chapter 15 that a transport protocol may depend on the existence of an upper bound on datagram lifetime. To avoid these problems, each datagram can be marked with a lifetime. Once the lifetime expires, the datagram is discarded.

A simple way to implement lifetime is to use a hop count. Each time that a datagram passes through a router, the count is decremented. Alternatively, the lifetime could be a true measure of time. This requires that the routers must somehow know how long it has been since the datagram or fragment last crossed a router, to know by how much to decrement the lifetime field. This would seem to require some global clocking mechanism. The advantage of using a true time measure is that it can be used in the reassembly algorithm, described next.

Fragmentation and Re-assembly

- Protocol exchanges data between two entities
- Lower-level protocols may need to break data up into smaller blocks, called fragmentation
- Reasons for fragmentation:
  - Network only accepts blocks of a certain size
  - More efficient error control and smaller retransmission units
  - Fairer access to shared facilities
  - Smaller buffers
- Disadvantages:
  - Smaller buffers
  - More interrupts and processing time

The Internet Protocol accepts a block of data from a higher-layer protocol, such as TCP or UDP, and may divide this block in to multiple blocks of some smaller bounded size to form multiple IP packets. This process is called **fragmentation**.

There are a number of motivations for fragmentation, depending on the context. Among the typical reasons for fragmentation:

The communications network may only accept blocks of data up to a certain size. For example, an ATM network is limited to blocks of 53 octets; Ethernet imposes a maximum size of 1526 octets.

Error control may be more efficient with a smaller PDU size. With smaller PDUs, fewer bits need to be retransmitted when a PDU suffers an error.

More equitable access to shared transmission facilities, with shorter delay, can be provided. For example, without a maximum block size, one station could monopolize a multipoint medium.

A smaller PDU size may mean that receiving entities can allocate smaller buffers.

An entity may require that data transfer comes to some sort of "closure" from time to time, for checkpoint and restart/recovery operations.

There are several disadvantages to fragmentation that argue for making PDUs as large as possible:

Because each PDU contains a certain amount of control information, smaller blocks have a greater percentage of overhead.

PDU arrival may generate an interrupt that must be serviced. Smaller blocks result in more interrupts.

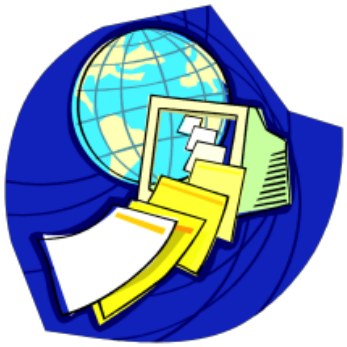
More time is spent processing smaller, more numerous PDUs.

Error and Flow Control

- Error control
  - Discarded datagram identification is needed
  - Reasons for discarded datagrams include:
    - Lifetime expiration
    - Congestion
    - FCS error

➢Flow control

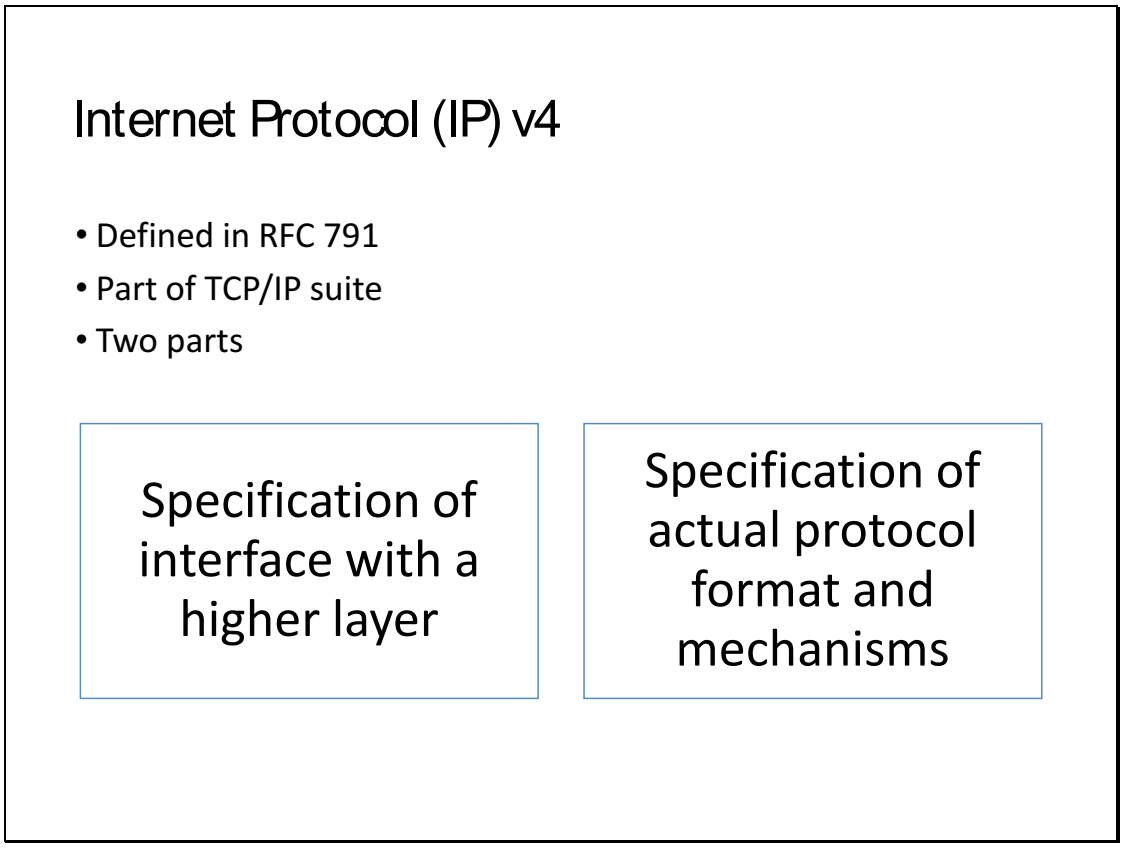
- Allows routers to limit the rate they receive data
- Send flow control packets requesting reduced data flow



The internetwork facility does not guarantee successful delivery of every datagram. When a datagram is discarded by a router, the router should attempt to return some information to the source, if possible. The source Internet Protocol entity may use this information to modify its transmission strategy and may notify higher layers. To report that a specific datagram has been discarded, some means of datagram identification is needed. Such identification is discussed in the next section.

Datagrams may be discarded for a number of reasons, including lifetime expiration, congestion, and FCS error. In the latter case, notification is not possible because the source address field may have been damaged.

Internet flow control allows routers and/or receiving stations to limit the rate at which they receive data. For the connectionless type of service we are describing, flow control mechanisms are limited. The best approach would seem to be to send flow control packets, requesting reduced data flow, to other routers and source stations. We will see one example of this with Internet Control Message Protocol, discussed in the next section.



In this section, we look at version 4 of IP, officially defined in RFC 791. Although it is intended that IPv4 will ultimately be replaced by IPv6, IPv4 is currently the dominant standard IP used in TCP/IP networks.

IP is part of the TCP/IP suite and is the most widely used internetworking protocol. As with any protocol standard, IP is specified in two parts (see Figure 2.9):

- The interface with a higher layer (e.g., TCP), specifying the services that IP provides

- The actual protocol format and mechanisms

In this section, we examine first IP services and then the protocol. This is followed by a discussion of IP address formats. Finally, the ICMP, which is an integral part of IP, is described.

IP Services

- Primitives
  - Specifies functions to be performed
  - Form of primitive implementation dependent
  - Send-request transmission of data unit
  - Deliver-notify user of arrival of data unit
- Parameters
  - Used to pass data and control information

The services to be provided across adjacent protocol layers (e.g., between IP and TCP) are expressed in terms of primitives and parameters. A primitive specifies the function to be performed, and the parameters are used to pass data and control information. The actual form of a primitive is implementation dependent. An example is a procedure call.

IP provides two service primitives at the interface to the next higher layer. The Send primitive is used to request transmission of a data unit. The Deliver primitive is used by IP to notify a user of the arrival of a data unit.

IP Parameters

- Source and destination addresses
- Protocol
- Type of Service
- Identification
- Don't fragment indicator
- Time to live
- Data length
- Option data
- User data

The parameters associated with the two primitives are :

Source address: Internetwork address of sending IP entity.

Destination address: Internetwork address of destination IP entity.

Protocol: Recipient protocol entity (an IP user, such as TCP).

Type-of-service indicators: Used to specify the treatment of the data unit in its transmission through component networks.

Identification: Used in combination with the source and destination addresses and user protocol to identify the data unit uniquely. This parameter is needed for reassembly and error reporting.

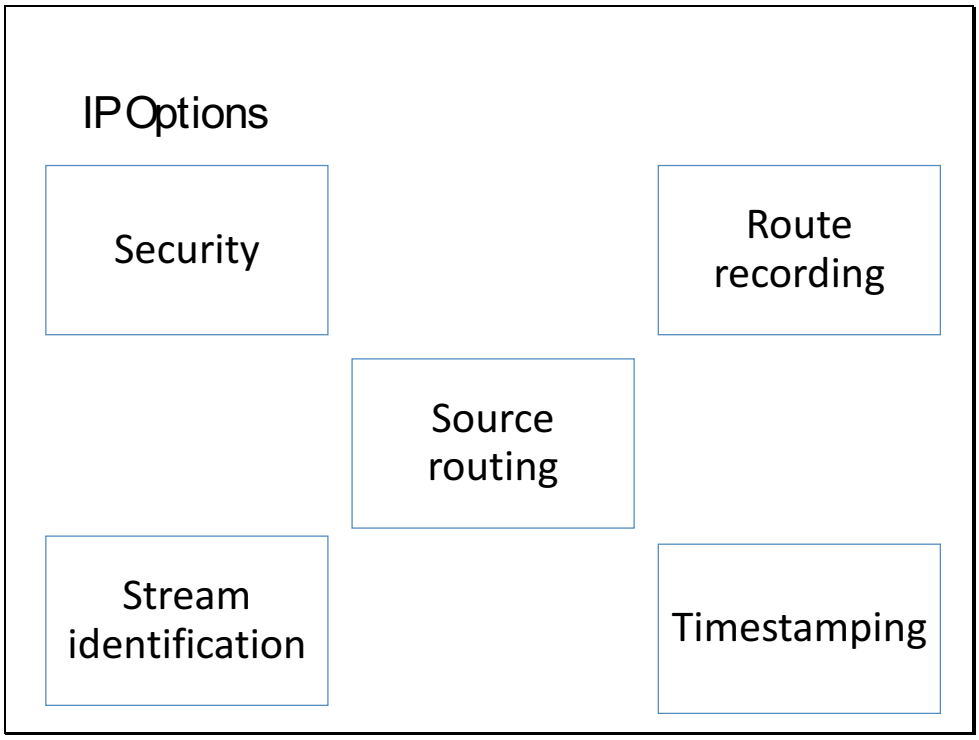
Don't fragment identifier: Indicates whether IP can fragment data to accomplish delivery.

Time to live: Measured in seconds.

Data length: Length of data being transmitted.

Option data: Options requested by the IP user.

Data: User data to be transmitted.
The *identification*, *don't fragment identifier*, and *time to live* parameters are present in the Send primitive but not in the Deliver primitive. These three parameters provide instructions to IP that are not of concern to the recipient IP user.



The options parameter allows for future extensibility and for inclusion of parameters that are usually not invoked. The currently defined options are

Security: Allows a security label to be attached to a datagram.

Source routing: A sequenced list of router addresses that specifies the route to be followed. Routing may be strict (only identified routers may be visited) or loose (other intermediate routers may be visited).

Route recording: A field is allocated to record the sequence of routers visited by the datagram.

Stream identification: Names reserved resources used for stream service. This service provides special handling for volatile periodic traffic (e.g., voice).

Timestamping: The source IP entity and some or all intermediate routers add a timestamp (precision to milliseconds) to the data unit as it goes by.

Slide 56

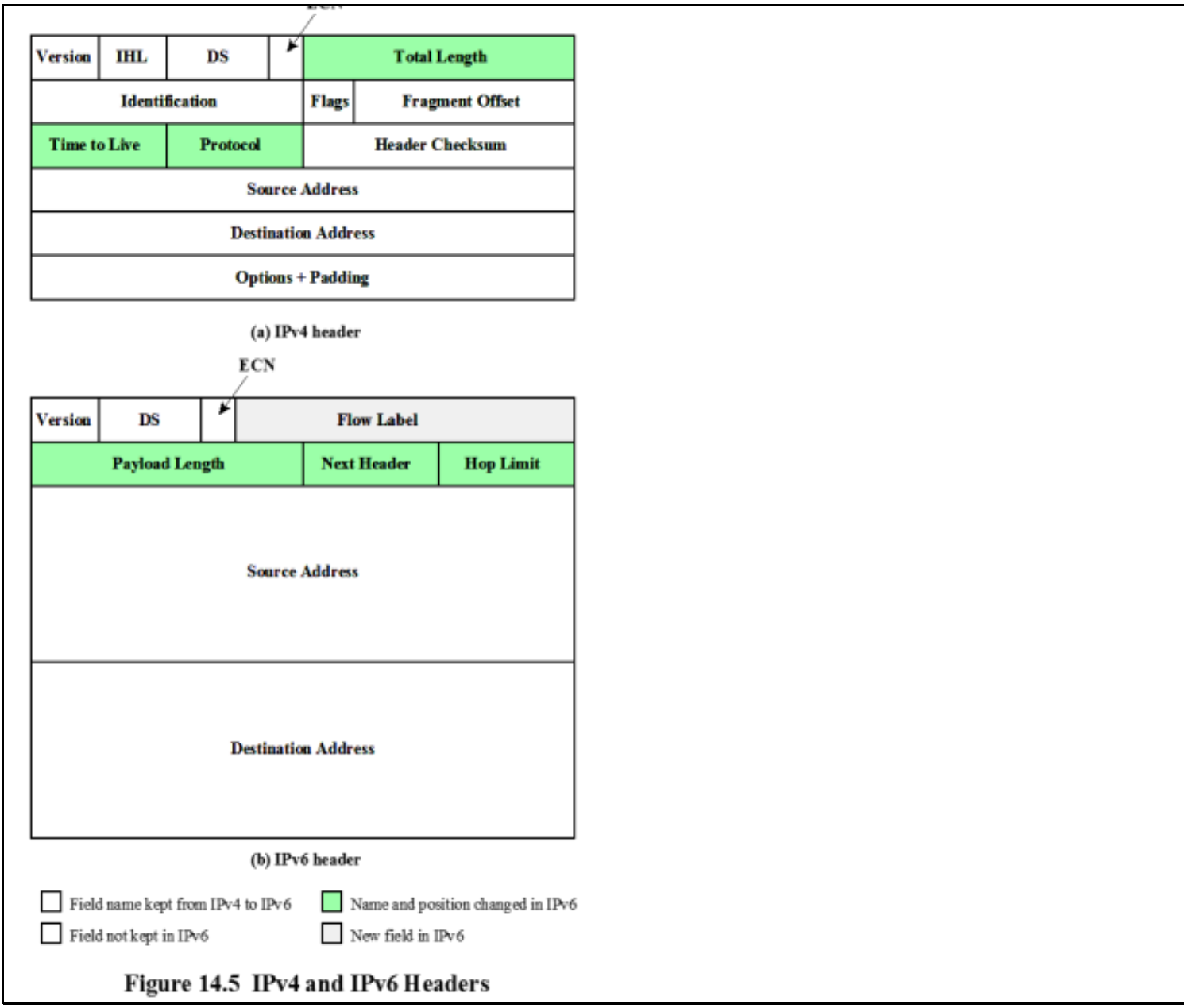


Figure 14.5 IPv4 and IPv6 Headers

The protocol between IP entities is best described with reference to the IP datagram format, shown in Figure 14.5a. The fields are

Version (4 bits): Indicates version number, to allow evolution of the protocol; the value is 4.

Internet Header Length (IHL) (4 bits): Length of header in 32-bit words. The minimum value is five, for a minimum header length of 20 octets.

DS (8 bits): This field supports the Differentiated Service function, described in Chapter 22.

ECN (2 bits): The Explicit Congestion Notification field, defined in RFC 3168, enables routers to indicate to end nodes packets that are experiencing congestion, without the necessity of immediately dropping such packets. A value of 00 indicates a packet

that is not using ECN. A value of 01 or 10 is set by the data sender to indicate that the end-points of the transport protocol are ECN-capable. A value of 11 is set by a router to indicate congestion has been encountered.

Total Length (16 bits): Total datagram length, including header plus data, in octets.

Identification (16 bits): A sequence number that, together with the source address, destination address, and user protocol, is intended to identify a datagram uniquely. Thus, this number should be unique for the datagram's source address, destination address, and user protocol for the time during which the datagram will remain in the internet.

Flags (3 bits): Only two of the bits are currently defined. The More bit is used for fragmentation and reassembly, as previously explained. The Don't Fragment bit prohibits fragmentation when set. This bit may be useful if it is known that the destination does not have the capability to reassemble fragments. However, if this bit is set, the datagram will be discarded if it exceeds the maximum size of an en route network. Therefore, if the bit is set, it may be advisable to use source routing to avoid networks with small maximum packet size.

Fragment Offset (13 bits): Indicates where in the original datagram this fragment belongs, measured in 64-bit units. This implies that fragments other than the last fragment must contain a data field that is a multiple of 64 bits in length.

Time to Live (8 bits): Specifies how long, in seconds, a datagram is allowed to remain in the internet. Every router that processes a datagram must decrease the TTL by at least one, so the TTL is similar to a hop count.

Protocol (8 bits): Indicates the next higher level protocol that is to receive the data field at the destination; thus, this field identifies the type of the next header in the packet after the IP header. Example values are TCP = 6; UDP = 17; ICMP = 1.

Header Checksum (16 bits): An error-detecting code applied to the header only. Because some header fields may change during transit (e.g., Time to Live, fragmentation-related fields), this is re-verified and recomputed at each router. The checksum is formed by taking the ones complement of the 16-bit ones complement addition of all 16-bit words in the header. For purposes of computation, the checksum field is itself initialized to a value of zero.

Source Address (32 bits): Coded to allow a variable allocation of bits to specify the network and the end system attached to the specified network, as discussed subsequently.

Destination Address (32 bits): Same characteristics as source address.

Options (variable): Encodes the options requested by the sending user.

Padding (variable): Used to ensure that the datagram header is a multiple of 32 bits in length.

Data (variable): The data field must be an integer multiple of 8 bits in length. The maximum length of the datagram (data field plus header) is 65,535 octets.

It should be clear how the IP services specified in the Send and Deliver primitives map into the fields of the IP datagram.

Internet Control Message Protocol (ICMP)

- RFC 792
- Provides a means for transferring messages from routers and other hosts to a host
- Provides feedback about problems
  - Datagram cannot reach its destination
  - Router does not have buffer capacity to forward
  - Router can send traffic on a shorter route
- Encapsulated in IP datagram
  - Hence not reliable

The IP standard specifies that a compliant implementation must also implement ICMP (RFC 792). ICMP provides a means for transferring messages from routers and other hosts to a host. In essence, ICMP provides feedback about problems in the communication environment. Examples of its use are when a datagram cannot reach its destination, when the router does not have the buffering capacity to forward a datagram, and when the router can direct the station to send traffic on a shorter route. In most cases, an ICMP message is sent in response to a datagram, either by a router along the datagram's path or by the intended destination host.

Although ICMP is, in effect, at the same level as IP in the TCP/IP architecture, it is a user of IP. An ICMP message is constructed and then passed down to IP, which encapsulates the message with an IP header and then transmits the resulting datagram in the usual fashion. Because ICMP messages are transmitted in IP datagrams, their delivery is not guaranteed and their use cannot be considered reliable.

Common ICMP Messages

- Destination unreachable
- Time exceeded
- Parameter problem
- Source quench
- Redirect
- Echo and echo reply
- Timestamp and timestamp reply
- Address mask request and reply

In those cases in which the ICMP message refers to a prior datagram, the information field includes the entire IP header plus the first 64 bits of the data field of the original datagram. This enables the source host to match the incoming ICMP message with the prior datagram. The reason for including the first 64 bits of the data field is that this will enable the IP module in the host to determine which upper-level protocol or protocols were involved. In particular, the first 64 bits would include a portion of the TCP header or other transport-level header.

The **destination unreachable** message covers a number of contingencies. A router may return this message if it does not know how to reach the destination network. In some networks, an attached router may be able to determine if a particular host is unreachable and returns the message. The destination host itself may return this message if the user protocol or some higher-level service access point is unreachable. This could happen if the corresponding field in the IP header was set incorrectly. If the datagram specifies a source route that is unusable, a message is returned. Finally, if a router must fragment a datagram but the Don't Fragment flag is set, the datagram is discarded and a message is returned.

A router will return a **time exceeded** message if the lifetime of the datagram expires. A host will send this message if it cannot complete reassembly within a time limit.

A syntactic or semantic error in an IP header will cause a **parameter problem** message to be returned by a router or host. For example, an incorrect argument may be provided with an option. The Parameter field contains a pointer to the octet in the original header where the error was detected.

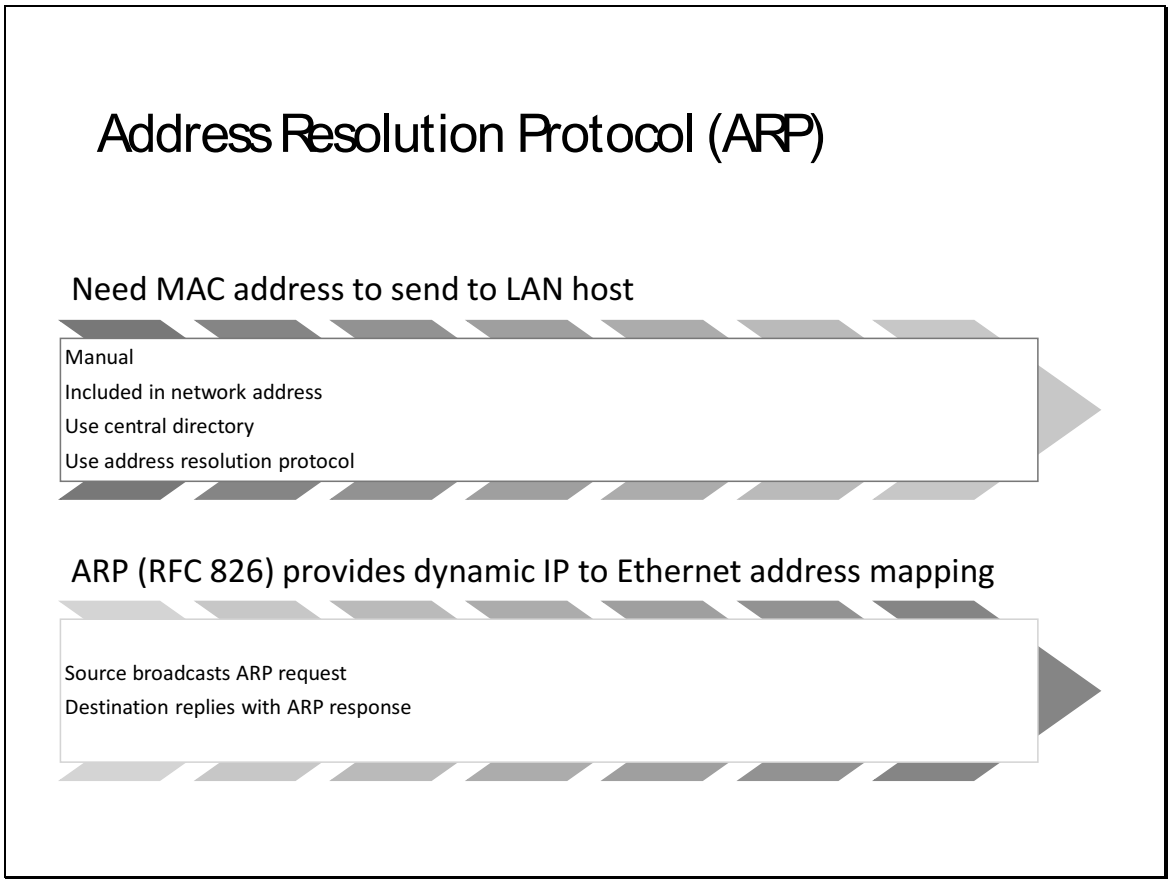
The **source quench** message provides a rudimentary form of flow control. Either a router or a destination host may send this message to a source host, requesting that it reduce the rate at which it is sending traffic to the internet destination. On receipt of a source quench message, the source host should cut back the rate at which it is sending traffic to the specified destination until it no longer receives source quench messages. The source quench message can be used by a router or host that must discard datagrams because of a full buffer. In that case, the router or host will issue a source quench message for every datagram that it discards. In addition, a system may anticipate congestion and issue source quench messages when its buffers approach capacity. In that case, the datagram referred to in the source quench message may well be delivered. Thus, receipt of a source quench message does not imply delivery or nondelivery of the corresponding datagram.

A router sends a **redirect** message to a host on a directly connected router to advise the host of a better route to a particular destination. The following is an example, using Figure 14.7. Router R1 receives a datagram from host C on network Y, to which R1 is attached. R1 checks its routing table and obtains the address for the next router, R2, on the route to the datagram's internet destination network, Z. Because R2 and the host identified by the internet source address of the datagram are on the same network, R1 sends a redirect message to C. The redirect message advises the host to send its traffic for network Z directly to router R2, because this is a shorter path to the destination. The router forwards the original datagram to its internet destination (via R2). The address of R2 is contained in the parameter field of the redirect message.

The **echo** and **echo reply** messages provide a mechanism for testing that communication is possible between entities. The recipient of an echo message is obligated to return the message in an echo reply message. An identifier and sequence number are associated with the echo message to be matched in the echo reply message. The identifier might be used like a service access point to identify a particular session, and the sequence number might be incremented on each echo request sent.

The **timestamp** and **timestamp reply** messages provide a mechanism for sampling the delay characteristics of the internet. The sender of a timestamp message may include an identifier and sequence number in the parameters field and include the time that the message is sent (originate timestamp). The receiver records the time it received the message and the time that it transmits the reply message in the timestamp reply message. If the timestamp message is sent using strict source routing, then the delay characteristics of a particular route can be measured.

The **address mask request** and **address mask reply** messages are useful in an environment that includes subnets. The address mask request and reply messages allow a host to learn the address mask for the LAN to which it connects. The host broadcasts an address mask request message on the LAN. The router on the LAN responds with an address mask reply message that contains the address mask.



Earlier in this chapter, we referred to the concepts of a global address (IP address) and an address that conforms to the addressing scheme of the network to which a host is attached (subnetwork address). For a local area network, the latter address is a MAC address, which provides a physical address for a host port attached to the LAN. Clearly, to deliver an IP datagram to a destination host, a mapping must be made from the IP address to the subnetwork address for that last hop. If a datagram traverses one or more routers between source and destination hosts, then the mapping must be done in the final router, which is attached to the same subnetwork as the destination host. If a datagram is sent from one host to another on the same subnetwork, then the source host must do the mapping. In the following discussion, we use the term *system* to refer to the entity that does the mapping.

For mapping from an IP address to a subnetwork address, a number of approaches are possible:

Each system can maintain a local table of IP addresses and matching subnetwork addresses for possible correspondents. This approach does not accommodate easy and automatic additions of new hosts to the subnetwork.

The subnetwork address can be a subset of the network portion of the IP address. However, the entire internet address is 32 bits long and for most subnetwork types (e.g., Ethernet) the host address field is longer than 32 bits.

A centralized directory can be maintained on each subnetwork that contains the IP-subnet address mappings. This is a reasonable solution for many networks.
An address resolution protocol can be used. This is a simpler approach than the use of a centralized directory and is well suited to LANs.

RFC 826 defines an Address Resolution Protocol (ARP), which allows dynamic distribution of the information needed to build tables to translate an IP address A into a 48-bit Ethernet address; the protocol can be used for any broadcast network. ARP exploits the broadcast property of a LAN; namely, that a transmission from any device on the network is received by all other devices on the network. ARP works as follows:

- Each system on the LAN maintains a table of known IP-subnetwork address mappings.

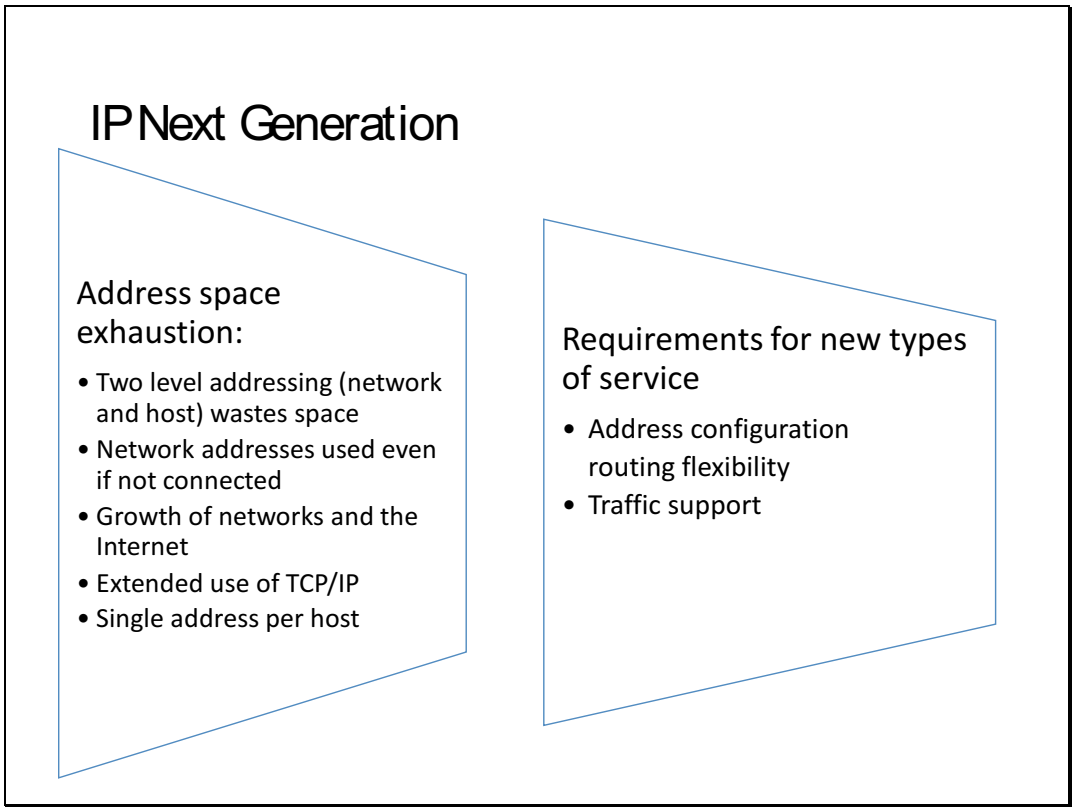
- When a subnetwork address is needed for an IP address, and the mapping is not found in the system's table, the system uses ARP directly on top of the LAN protocol (e.g., IEEE 802) to broadcast a request. The broadcast message contains the IP address for which a subnetwork address is needed.

- Other hosts on the subnetwork listen for ARP messages and reply when a match occurs. The reply includes both the IP and subnetwork addresses of the
  replying host.

- The original request includes the requesting host's IP address and subnetwork address. Any interested host can copy this information into its local table,
  avoiding the need for later ARP messages.

5. The ARP message can also be used simply to broadcast a host's IP address and subnetwork address, for the benefit of others on the subnetwork.

Slide 60



The driving motivation for the adoption of a new version of IP was the limitation imposed by the 32-bit address field in IPv4. With a 32-bit address field, it is possible in principle to assign 232 different addresses, which is over 4 billion possible addresses. One might think that this number of addresses was more than adequate to meet addressing needs on the Internet. However, in the late 1980s it was perceived that there would be a problem, and this problem began to manifest itself in the early 1990s. Reasons for the inadequacy of 32-bit addresses include the following:

The two-level structure of the IP address (network number, host number) is convenient but wasteful of the address space. Once a network number is assigned to a network, all of the host-number addresses for that network number are assigned to that network. The address space for that network may be sparsely used, but as far as the effective IP address space is concerned, if a network number is used, then all addresses within the network are used.

The IP addressing model generally requires that a unique network number be assigned to each IP network whether or not it is actually connected to the Internet.

Networks are proliferating rapidly. Most organizations boast multiple LANs, not just a single LAN system. Wireless networks have rapidly assumed a major role. The Internet itself has grown explosively for years.

Growth of TCP/IP usage into new areas will result in a rapid growth in the demand for unique IP addresses. Examples include using TCP/IP to interconnect electronic point-of-sale terminals and for cable television receivers.

Typically, a single IP address is assigned to each host. A more flexible arrangement is to allow multiple IP addresses per host. This, of course, increases the demand for IP addresses.

So the need for an increased address space dictated that a new version of IP was needed. In addition, IP is a very old protocol, and new requirements in the areas of address configuration, routing flexibility, and traffic support had been defined.

IPv6 RFCs

- RFC 1752 - Recommendations for the IP Next Generation Protocol
  - Requirements
  - PDU formats
  - Addressing, routing security issues
- RFC 2460 - overall specification
- RFC 4291 - addressing structure

In response to these needs, the Internet Engineering Task Force (IETF) issued a call for proposals for a next generation IP (IPng) in July of 1992. A number of proposals were received, and by 1994 the final design for IPng emerged. A major milestone was reached with the publication of RFC 1752, "The Recommendation for the IP Next Generation Protocol," issued in January 1995. RFC 1752 outlines the requirements for IPng, specifies the PDU formats, and highlights the IPng approach in the areas of addressing, routing, and security. A number of other Internet documents defined details of the protocol, now officially called IPv6; these include an overall specification of IPv6 (RFC 2460), an RFC dealing with addressing structure of IPv6 (RFC 4291), and numerous others.

IPv6 Enhancements

- Expanded 128 bit address space
- Improved option mechanism
  - Most not examined by intermediate routes
- Dynamic address assignment
- Increased addressing flexibility
  - Anycast and multicast
- Support for resource allocation
  - Labeled packet flows

IPv6 includes the following enhancements over IPv4:

Expanded address space: IPv6 uses 128-bit addresses instead of the 32-bit addresses of IPv4. This is an increase of address space by a factor of 2^{96}. It has been pointed out [HIND95] that this allows on the order of 6×10^{23} unique addresses per square meter of the surface of the earth. Even if addresses are very inefficiently allocated, this address space seems inexhaustible.

Improved option mechanism: IPv6 options are placed in separate optional headers that are located between the IPv6 header and the transport-layer header. Most of these optional headers are not examined or processed by any router on the packet's path. This simplifies and speeds up router processing of IPv6 packets compared to IPv4 datagrams. It also makes it easier to add additional options.

Address auto configuration: This capability provides for dynamic assignment of IPv6 addresses.

Increased addressing flexibility: IPv6 includes the concept of an anycast address, for which a packet is delivered to just one of a set of nodes. The scalability of multicast routing is improved by adding a scope field to multicast addresses.

Support for resource allocation: IPv6 enables the labeling of packets belonging to a particular traffic flow for which the sender requests special handling. This aids in the support of specialized traffic such as real-time video.

Slide 63

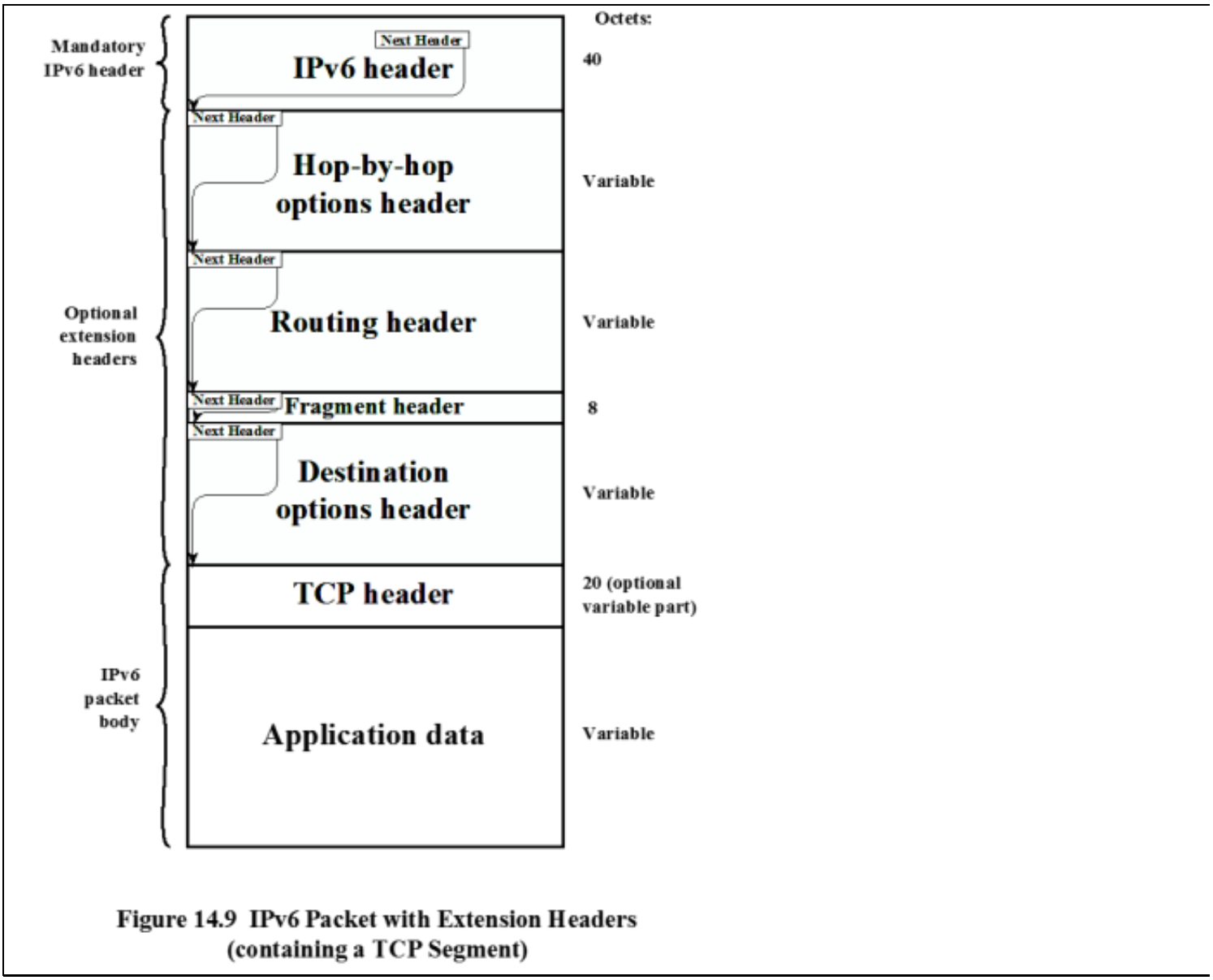


Figure 14.9 IPv6 Packet with Extension Headers (containing a TCP Segment)

The only header that is required is referred to simply as the IPv6 header. This is of fixed size with a length of 40 octets, compared to 20 octets for the mandatory portion of the IPv4 header (Figure 14.5a). The following extension headers have been defined:

Hop-by-Hop Options header: Defines special options that require hop-by-hop processing

Routing header: Provides extended routing, similar to IPv4 source routing

Fragment header: Contains fragmentation and reassembly information

Authentication header: Provides packet integrity and authentication

Encapsulating Security Payload header: Provides privacy

Destination Options header: Contains optional information to be examined by the destination node

The IPv6 standard recommends that, when multiple extension headers are used, the IPv6 headers appear in the following order:

1. IPv6 header: Mandatory, must always appear first

2. Hop-by-Hop Options header

3. Destination Options header: For options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header

4. Routing header

5. Fragment header

6. Authentication header

- Encapsulating Security Payload header

- Destination Options header: For options to be processed only by the final destination of the packet

Figure 14.9 shows an example of an IPv6 packet that includes an instance of each header, except those related to security. Note that the IPv6 header and each extension header include a Next Header field. This field identifies the type of the immediately following header. If the next header is an extension header, then this field contains the type identifier of that header. Otherwise, this field contains the protocol identifier of the upper-layer protocol using IPv6 (typically a transport-level protocol), using the same values as the IPv4 Protocol field. In Figure 14.9, the upper-layer protocol is TCP; thus, the upper-layer data carried by the IPv6 packet consist of a TCP header followed by a block of application data.

We first look at the main IPv6 header and then examine each of the extensions in turn.

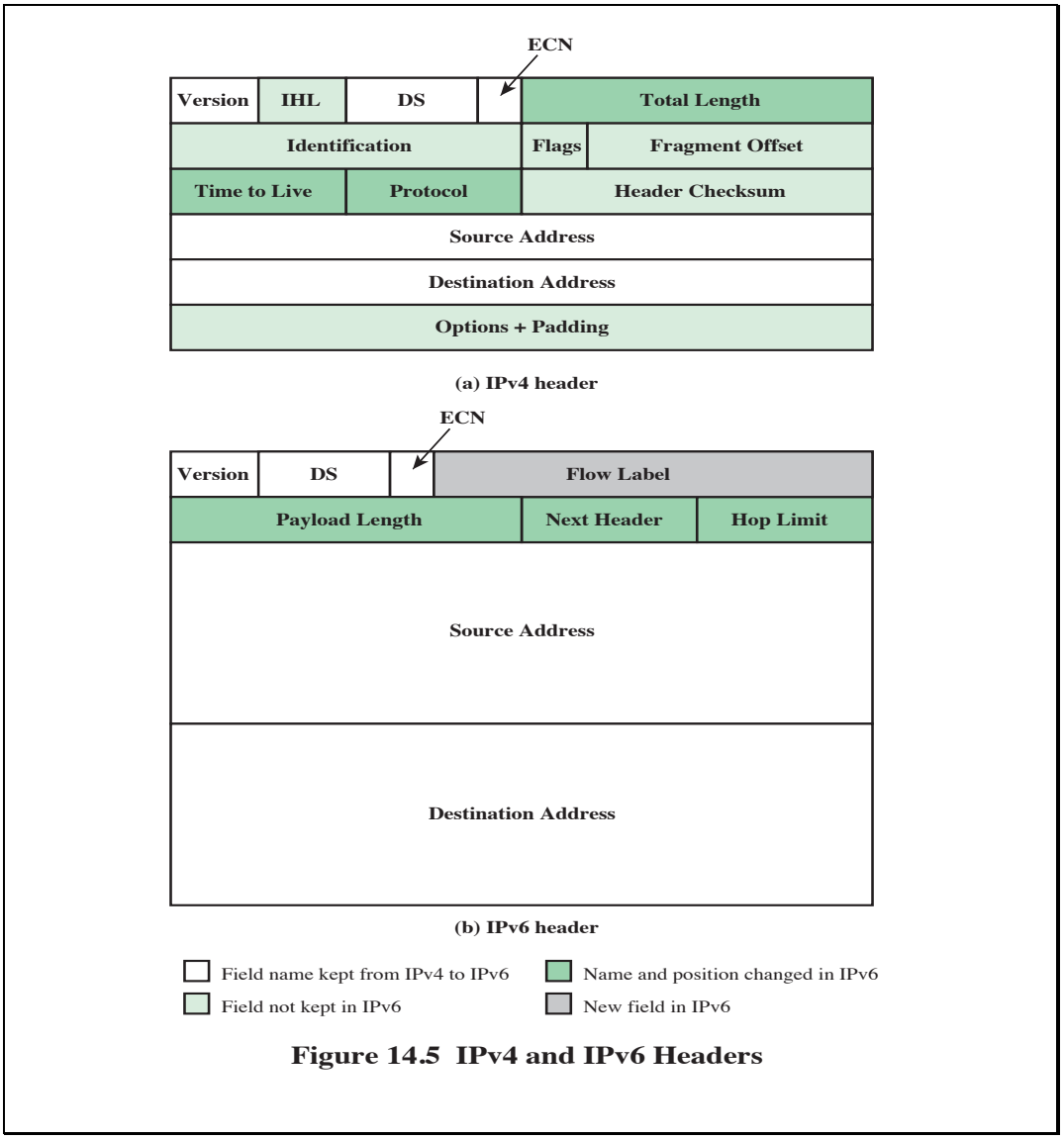


Figure 14.5 IPv4 and IPv6 Headers

The IPv6 header has a fixed length of 40 octets, consisting of the following fields (Figure 14.5b):

Version (4 bits): Internet protocol version number; the value is 6.

DS/ECN (8 bits): Available for use by originating nodes and/or forwarding routers for differentiated services and congestion functions, as described for the IPv4 DS/ECN field.

Flow Label (20 bits): May be used by a host to label those packets for which it is requesting special handling by routers within a network; discussed subsequently.

Payload Length (16 bits): Length of the remainder of the IPv6 packet following the header, in octets. In other words, this is the total length of all of the extension headers plus the transport-level PDU.

Next Header (8 bits): Identifies the type of header immediately following the IPv6 header; this will either be an IPv6 extension header or a higher-layer header, such as TCP or UDP.

Hop Limit (8 bits): The remaining number of allowable hops for this packet. The hop limit is set to some desired maximum value by the source and decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero. This is a simplification over the processing required for the Time to Live field of IPv4. The consensus was that the extra effort in accounting for time intervals in IPv4 added no significant value to the protocol. In fact, IPv4 routers, as a general rule, treat the Time to Live field as a hop limit field.

Source Address (128 bits): The address of the originator of the packet.

Destination Address (128 bits): The address of the intended recipient of the packet. This may not in fact be the intended ultimate destination if a Routing header is present, as explained subsequently.

Although the IPv6 header is longer than the mandatory portion of the IPv4 header (40 octets versus 20 octets), it contains fewer fields (8 versus 12). Thus, routers have less processing to do per header, which should speed up routing.

IPv6 Flow Label

- Related sequence of packets
- Special handling
- Identified by source and destination address plus flow label
- Router treats flow as sharing attributes
- May treat flows differently
- Alternative to including all information in every header
- Have requirements on flow label processing

RFC 3697 defines a flow as a sequence of packets sent from a particular source to a particular (unicast, anycast, or multicast) destination for which the source desires special handling by the intervening routers. A flow is uniquely identified by the combination of a source address, destination address, and a nonzero 20-bit flow label. Thus, all packets that are to be part of the same flow are assigned the same flow label by the source.

From the source's point of view, a flow typically will be a sequence of packets that are generated from a single application instance at the source and that have the same transfer service requirements. A flow may comprise a single TCP connection or even multiple TCP connections; an example of the latter is a file transfer application, which could have one control connection and multiple data connections. A single application may generate a single flow or multiple flows. An example of the latter is multimedia conferencing, which might have one flow for audio and one for graphic windows, each with different transfer requirements in terms of data rate, delay, and delay variation.

From the router's point of view, a flow is a sequence of packets that share attributes that affect how these packets are handled by the router. These include path, resource allocation, discard requirements, accounting, and security attributes. The router may treat packets from different flows differently in a number of ways, including allocating different buffer sizes, giving different precedence in terms of forwarding, and requesting different quality of service from networks.

There is no special significance to any particular flow label. Instead the special handling to be provided for a packet flow must be declared in some other way. For example, a source might negotiate or request special handling ahead of time from routers by means of a control protocol, or at transmission time by information in one of the extension headers in the packet, such as the Hop-by-Hop Options header. Examples of special

handling that might be requested include some sort of non-default quality of service and some form of real-time service.

In principle, all of a user's requirements for a particular flow could be defined in an extension header and included with each packet. If we wish to leave the concept of flow open to include a wide variety of requirements, this design approach could result in very large packet headers. The alternative, adopted for IPv6, is the flow label, in which the flow requirements are defined prior to flow commencement and a unique flow label is assigned to the flow. In this case, the router must save flow requirement information about each flow.

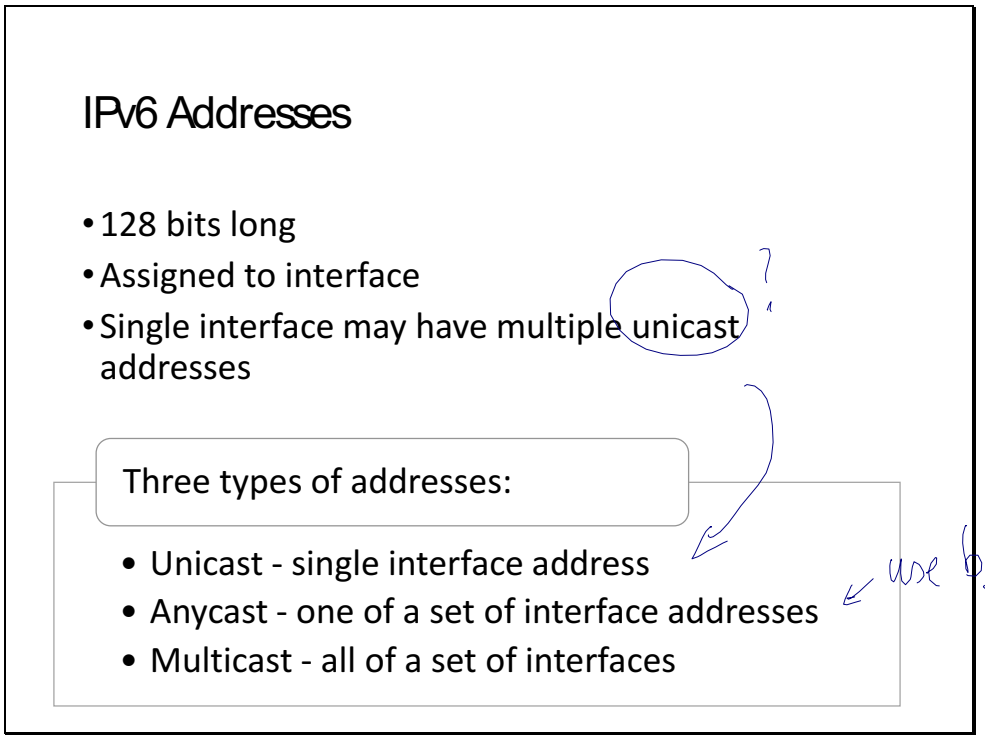
The following rules apply to the flow label:

- Hosts or routers that do not support the Flow Label field must set the field to zero when originating a packet, pass the field unchanged when forwarding a packet, and ignore the field when receiving a packet.

- All packets originating from a given source with the same nonzero Flow Label must have the same Destination Address, Source Address, Hop-by-Hop Options header contents (if this header is present), and Routing header contents (if this header is present). The intent is that a router can decide how to route and process the packet by simply looking up the flow label in a table and without examining the rest of the header.

- The source assigns a flow label to a flow. New flow labels must be chosen (pseudo-) randomly and uniformly in the range 1 to $2^{20} – 1$, subject to the restriction
  that a source must not reuse a flow label for a new flow within the lifetime of the existing flow. The zero flow label is reserved to indicate that no flow label is being used.

This last point requires some elaboration. The router must maintain information about the characteristics of each active flow that may pass through it, presumably in some sort of table. To forward packets efficiently and rapidly, table lookup must be efficient. One alternative is to have a table with 2^{20} (about 1 million) entries, one for each possible flow label; this imposes an unnecessary memory burden on the router. Another alternative is to have one entry in the table per active flow, include the flow label with each entry, and require the router to search the entire table each time a packet is encountered. This imposes an unnecessary processing burden on the router. Instead, most router designs are likely to use some sort of hash table approach. With this approach a moderate-sized table is used, and each flow entry is mapped into the table using a hashing function on the flow label. The hashing function might simply be the low-order few bits (say 8 or 10) of the flow label or some simple calculation on the 20 bits of the flow label. In any case, the efficiency of the hash approach typically depends on the flow labels being uniformly distributed over their possible range. Hence requirement number 3 in the preceding list.



IPv6 addresses are 128 bits in length. Addresses are assigned to individual interfaces on nodes, not to the nodes themselves. A single interface may have multiple unique unicast addresses. Any of the unicast addresses associated with a node's interface may be used to uniquely identify that node.

The combination of long addresses and multiple addresses per interface enables improved routing efficiency over IPv4. In IPv4, addresses generally do not have a structure that assists routing, and therefore a router may need to maintain huge table of routing paths. Longer internet addresses allow for aggregating addresses by hierarchies of network, access provider, geography, corporation, and so on. Such aggregation should make for smaller routing tables and faster table lookups. The allowance for multiple addresses per interface would allow a subscriber that uses multiple access providers across the same interface to have separate addresses aggregated under each provider's address space.

IPv6 allows three types of addresses:

Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).

Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

Connection-Oriented Transport Mechanisms

- Two basic types of transport service:

Connection-oriented

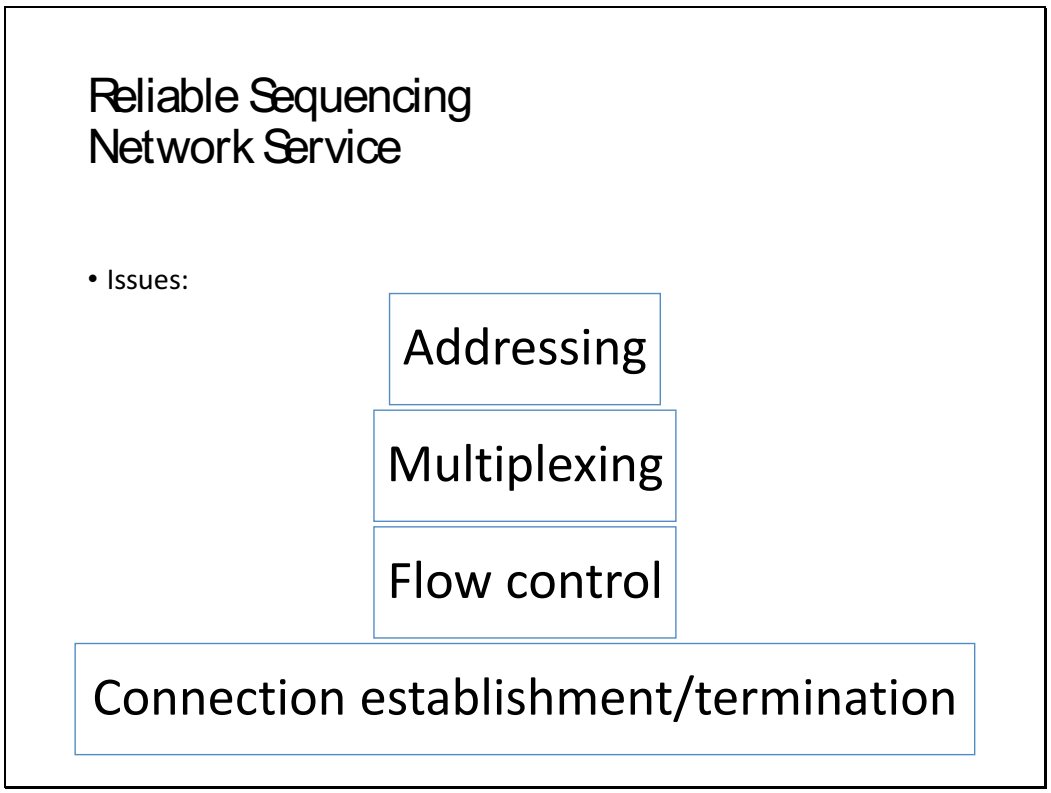
- Establishment, maintenance and termination of a logical connection between TS users
- Has a wide variety of applications
- Most common
- Implies service is reliable

Connectionless or datagram service

Two basic types of transport service are possible: connection oriented and connectionless or datagram service. A connection-oriented service provides for the establishment, maintenance, and termination of a logical connection between TS users. This has, so far, been the most common type of protocol service available and has a wide variety of applications. The connection-oriented service generally implies that the service is reliable. This section looks at the transport protocol mechanisms needed to support the connection-oriented service.

A full-feature connection-oriented transport protocol, such as TCP, is very complex. For purposes of clarity we present the transport protocol mechanisms in an evolutionary fashion. We begin with a network service that makes life easy for the transport protocol, by guaranteeing the delivery of all transport data units in order and defining the required mechanisms. Then we will look at the transport protocol mechanisms required to cope with an unreliable network service. All of this discussion applies in general to transport-level protocols. In Section 15.2, we apply the concepts developed in this section to describe TCP.

Slide 68



Let us assume that the network service accepts messages of arbitrary length and, with virtually 100% reliability, delivers them in sequence to the destination. Examples of such networks:

A highly reliable packet-switching network with an X.25 interface

A frame relay network using the LAPF control protocol

An IEEE 802.3 LAN using the connection-oriented LLC service

In all of these cases, the transport protocol is used as an end-to-end protocol between two systems attached to the same network, rather than across an internet.

The assumption of a reliable sequencing networking service allows the use of a quite simple transport protocol. Four issues need to be addressed:

Addressing

Multiplexing

Flow control

Connection establishment/termination

Flow Control

- Complex at the transport layer:
  - Considerable delay in the communication of flow control information
  - Amount of the transmission delay may be highly variable, making it difficult to effectively use a timeout mechanism for retransmission of lost data

Reasons for control:	
User of the receiving transport entity cannot keep up with the flow	Receiving transport entity itself cannot keep up with the flow of segments

Whereas flow control is a relatively simple mechanism at the link layer, it is a rather complex mechanism at the transport layer, for two main reasons:

The transmission delay between transport entities is generally long compared to actual transmission time. This means that there is a considerable delay in the communication of flow control information.

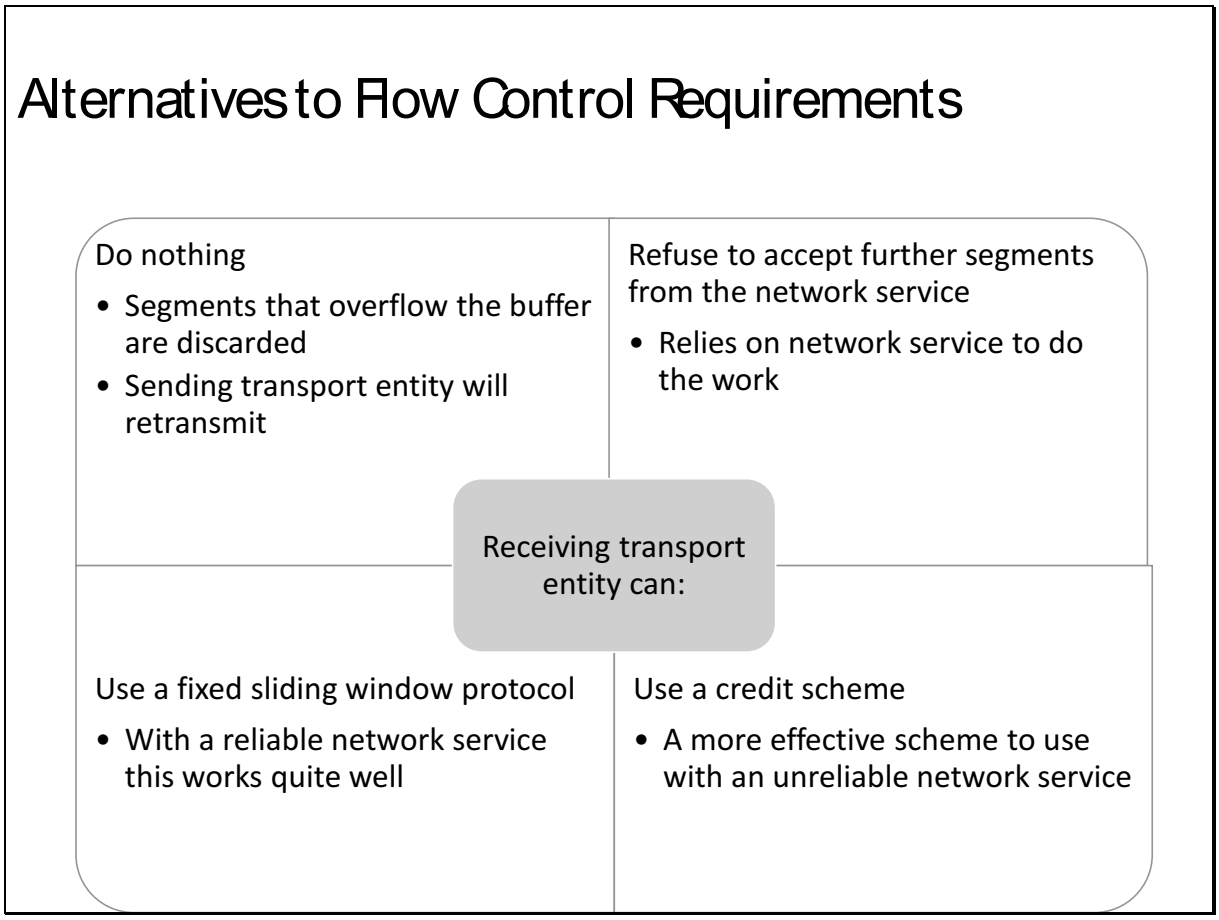
Because the transport layer operates over a network or internet, the amount of the transmission delay may be highly variable. This makes it difficult to effectively use a timeout mechanism for retransmission of lost data.

In general, there are two reasons why one transport entity would want to restrain the rate of segment transmission over a connection from another transport entity:

The user of the receiving transport entity cannot keep up with the flow of data.

The receiving transport entity itself cannot keep up with the flow of segments.

How do such problems manifest themselves? Presumably a transport entity has a certain amount of buffer space. Incoming segments are added to the buffer. Each buffered segment is processed (i.e., the transport header is examined) and the data are sent to the TS user. Either of the two problems just mentioned will cause the buffer to fill up. Thus, the transport entity needs to take steps to stop or slow the flow of segments to prevent buffer overflow. This requirement is difficult to fulfill because of the annoying time gap between sender and receiver. We return to this point subsequently.



First, we present four ways of coping with the flow control requirement. The receiving transport entity can

1. Do nothing.
2. Refuse to accept further segments from the network service.
3. Use a fixed sliding-window protocol.
4. Use a credit scheme.

Alternative 1 means that the segments that overflow the buffer are discarded. The sending transport entity, failing to get an acknowledgment, will retransmit. This is a shame, because the advantage of a reliable network is that one never has to retransmit. Furthermore, the effect of this maneuver is to exacerbate the problem. The sender has increased its output to include new segments plus retransmitted old segments.

The second alternative is a backpressure mechanism that relies on the network service to do the work. When a buffer of a transport entity is full, it refuses additional data from the network service. This triggers flow control procedures within the network that throttle the network service at the sending end. This service, in turn, refuses additional segments from its transport entity. It should be clear that this mechanism is clumsy and coarse grained. For example, if multiple transport connections are multiplexed on a single network connection (virtual circuit), flow control is exercised only on the aggregate of all transport connections.

The third alternative is already familiar to you from our discussions of link layer protocols in Chapter 7. The key ingredients, recall, are

The use of sequence numbers on data units

The use of a window of fixed size

The use of acknowledgments to advance the window

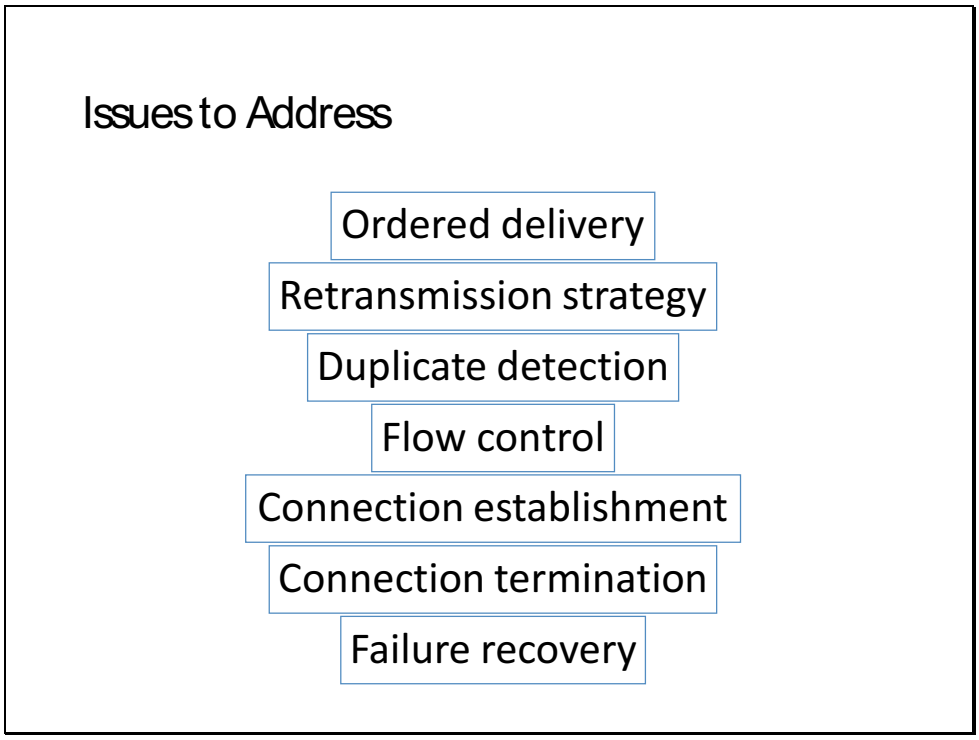
With a reliable network service, the sliding-window technique would work quite well. For example, consider a protocol with a window size of 7. When the sender receives an acknowledgment to a particular segment, it is automatically authorized to send the succeeding seven segments (of course, some may already have been sent). When the receiver's buffer capacity gets down to seven segments, it can withhold acknowledgment of incoming segments to avoid overflow. The sending transport entity can send at most seven additional segments and then must stop. Because the underlying network service is reliable, the sender will not time out and retransmit. Thus, at some point, a sending transport entity may have a number of segments outstanding for which no acknowledgment has been received. Because we are dealing with a reliable network, the sending transport entity can assume that the segments will get through and that the lack of acknowledgment is a flow control tactic. This tactic would not work well in an unreliable network, because the sending transport entity would not know whether the lack of acknowledgment is due to flow control or a lost segment.

The fourth alternative, a credit scheme, provides the receiver with a greater degree of control over data flow. Although it is not strictly necessary with a reliable network service, a credit scheme should result in a smoother traffic flow. Further, it is a more effective scheme with an unreliable network service, as we shall see.

The credit scheme decouples acknowledgment from flow control. In fixed sliding-window protocols, such as X.25 and HDLC, the two are synonymous. In a credit scheme, a segment may be acknowledged without granting new credit, and vice versa. For the credit scheme, each individual octet of data that is transmitted is considered to have a unique sequence number. In addition to data, each transmitted segment includes in its header three fields related to flow control: **sequence number** (*SN*), **acknowledgment number** (*AN*), and **window** (*W*). When a transport entity sends a segment, it includes the sequence number of the first octet in the segment data field. Implicitly, the remaining data octets are numbered sequentially following the first data octet. A transport entity acknowledges an incoming segment with a return segment that includes ($AN = i$, $W = j$), with the following interpretation:

All octets through sequence number $SN = i - 1$ are acknowledged; the next expected octet has sequence number i.

Permission is granted to send an additional window of $W = j$ octets of data; that is, the j octets corresponding to sequence numbers i through $i + j - 1$.



In the remainder of this section, unless otherwise noted, the mechanisms discussed are those used by TCP. Seven issues need to be addressed:

Ordered delivery
Retransmission strategy
Duplicate detection
Flow control
Connection establishment
Connection termination
Failure recovery

User Datagram Protocol (UDP)

- Transport-level protocol that is commonly used as part of the TCP/IP protocol suite
- RFC 768
- Provides a connectionless service for application-level procedures
- Unreliable service; delivery and duplicate protection are not guaranteed
- Reduces overhead and may be adequate in many cases

In addition to TCP, there is one other transport-level protocol that is in common use as part of the TCP/IP suite: the User Datagram Protocol (UDP), specified in RFC 768. UDP provides a connectionless service for application-level procedures. Thus, UDP is basically an unreliable service; delivery and duplicate protection are not guaranteed. However, this does reduce the overhead of the protocol and may be adequate in many cases.

The strengths of the connection-oriented approach are clear. It allows connection-related features such as flow control, error control, and sequenced delivery. Connectionless service, however, is more appropriate in some contexts. At lower layers (internet, network), a connectionless service is more robust (e.g., see discussion in Section 9.5). In addition, it represents a "least common denominator" of service to be expected at higher layers. Further, even at transport and above there is justification for a connectionless service. There are instances in which the overhead of connection establishment and termination is unjustified or even counterproductive. Examples include the following:

• Inward data collection: Involves the periodic active or passive sampling of data sources, such as sensors, and automatic self-test reports from security equipment or network components. In a real-time monitoring situation, the loss of an occasional data unit would not cause distress, because the next report should arrive shortly.

• Outward data dissemination: Includes broadcast messages to network users, the announcement of a new node or the change of address of a service, and the distribution of real-time clock values.

• Request-response: Applications in which a transaction service is provided by a common server to a number of distributed TS users, and for which a single request-response sequence is typical. Use of the service is regulated at the application level, and lower-level connections are often unnecessary and cumbersome.

• Real-time applications: Applications, such as voice and telemetry, that involve a degree of redundancy and/or a real-time transmission requirement. These must not have connection-oriented functions such as retransmission.

Thus, there is a place at the transport level for both a connection-oriented and a connectionless type of service.

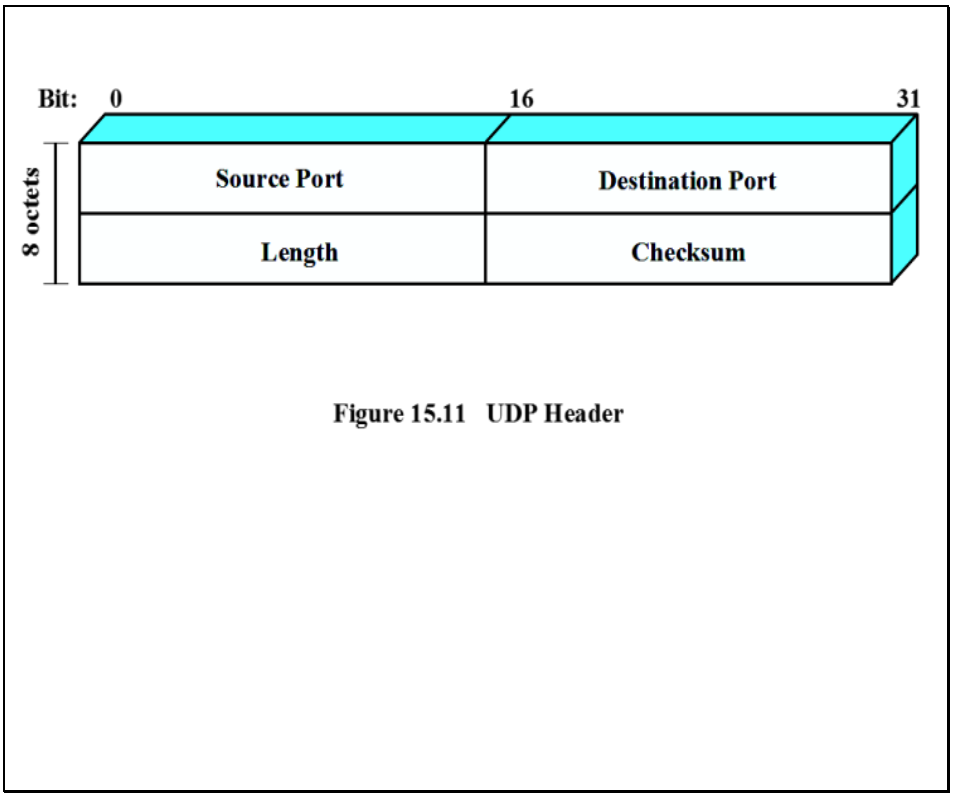


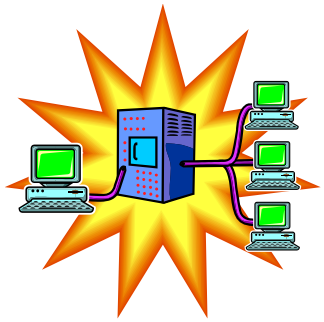
Figure 15.11 UDP Header

UDP sits on top of IP. Because it is connectionless, UDP has very little to do. Essentially, it adds a port addressing capability to IP. This is best seen by examining the UDP header, shown in Figure 15.11. The header includes a source port and destination port. The Length field contains the length of the entire UDP segment, including header and data. The checksum is the same algorithm used for TCP and IP. For UDP, the checksum applies to the entire UDP segment plus a pseudoheader prefixed to the UDP header at the time of calculation and which is the same pseudoheader used for TCP. If an error is detected, the segment is discarded and no further action is taken.

The checksum field in UDP is optional. If it is not used, it is set to zero. However, it should be pointed out that the IP checksum applies only to the IP header and not to the data field, which in this case consists of the UDP header and the user data. Thus, if no checksum calculation is performed by UDP, then no check is made on the user data at either the transport or internet protocol layers.

Routing in Packet Switching Networks

- Key design issue for (packet) switched networks
- Select route across network between end nodes
- Characteristics required:
  - Correctness
  - Simplicity
  - Robustness
  - Stability
  - Fairness
  - Optimality
  - Efficiency



The primary function of a packet-switching network is to accept packets from a source station and deliver them to a destination station. To accomplish this, a path or route through the network must be determined; generally, more than one route is possible. Thus, a routing function must be performed. The requirements for this function include the following:

- Correctness
- Simplicity
- Robustness
- Stability
- Fairness
- Optimality
- Efficiency

The first two items on the list, correctness and simplicity, are self-explanatory. Robustness has to do with the ability of the network to deliver packets via some route in the face of localized failures and overloads. Ideally, the network can react to such contingencies without the loss of packets or the breaking of virtual circuits. The designer who seeks robustness must cope with the competing requirement for

stability. Techniques that react to changing conditions have an unfortunate tendency to either react too slowly to events or to experience unstable swings from one extreme to another. For example, the network may react to congestion in one area by shifting most of the load to a second area. Now the second area is overloaded and the first is underutilized, causing a second shift. During these shifts, packets may travel in loops through the network.

A trade-off also exists between fairness and optimality. Some performance criteria may give higher priority to the exchange of packets between nearby stations compared to an exchange between distant stations. This policy may maximize average throughput but will appear unfair to the station that primarily needs to communicate with distant stations.

Finally, any routing technique involves some processing overhead at each node and often a transmission overhead as well, both of which impair network efficiency. The penalty of such overhead needs to be less than the benefit accrued based on some reasonable metric, such as increased robustness or fairness.

Elements of Routing Techniques for Packet-Switching Networks

Performance Criteria	**Network Information Source**
Number of hops	None
Cost	Local
Delay	Adjacent node
Throughput	Nodes along route
	All nodes
Decision Time	
Packet (datagram)	**Network Information Update Timing**
Session (virtual circuit)	Continuous
	Periodic
Decision Place	Major load change
Each node (distributed)	Topology change
Central node (centralized)	
Originating node (source)	

With these requirements in mind, we are in a position to assess the various design elements that contribute to a routing strategy. Table 19.1 lists these elements. Some of these categories overlap or are dependent on one another. Nevertheless, an examination of this list serves to clarify and organize routing concepts.

Autonomous Systems (AS)

- Exhibits the following characteristics:
  - Is a set of routers and networks managed by a single organization
  - Consists of a group of routers exchanging information via a common routing protocol
  - Except in times of failure, is connected (in a graph-theoretic sense); there is a path between any pair of nodes

To proceed with our discussion of routing protocols, we need to introduce the concept of an autonomous system (AS) . An AS exhibits the following characteristics:

1. An AS is a set of routers and networks managed by a single organization.

2. An AS consists of a group of routers exchanging information via a common routing protocol.

3. Except in times of failure, an AS is connected (in a graph-theoretic sense); that is, there is a path between any pair of nodes.

Interior Router Protocol (IRP)

- A shared routing protocol which passes routing information between routers within an AS
- Custom tailored to specific applications and requirements

A shared routing protocol, which we shall refer to as an interior router protocol (IRP) , passes routing information between routers within an AS. The protocol used within the AS does not need to be implemented outside of the system. This flexibility allows IRPs to be custom tailored to specific applications and requirements.

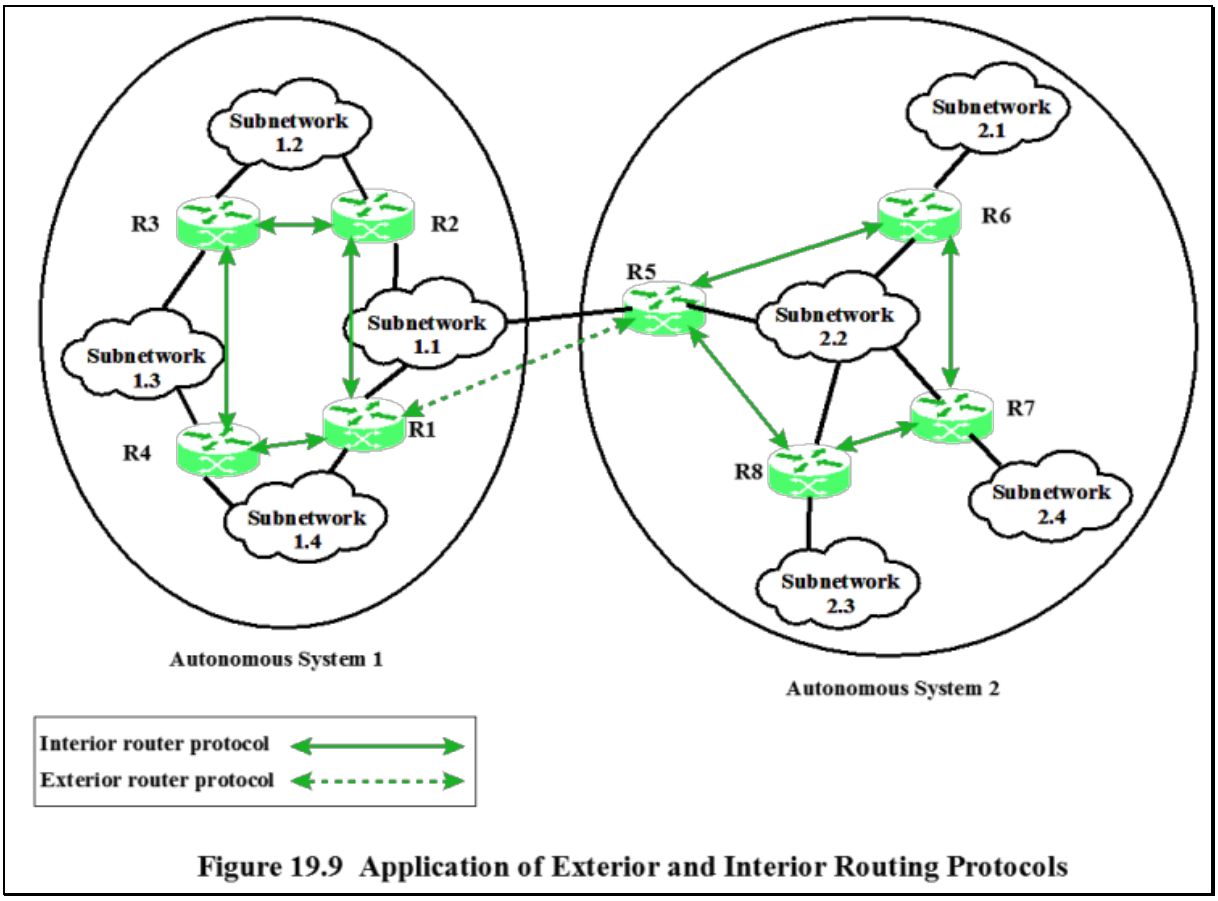


Figure 19.9 Application of Exterior and Interior Routing Protocols

It may happen, however, that an internet will be constructed of more than one AS. For example, all of the LANs at a site, such as an office complex or campus, could be linked by routers to form an AS. This system might be linked through a wide area network to other ASs. The situation is illustrated in Figure 19.9. In this case, the routing algorithms and information in routing tables used by routers in different ASs may differ. Nevertheless, the routers in one AS need at least a minimal level of information concerning networks outside the system that can be reached.

Exterior Router Protocol (ERP)

- Protocol used to pass routing information between routers in different ASs
- Will need to pass less information than an IRP for the following reason:
  - If a datagram is to be transferred from a host in one AS to a host in another AS, a router in the first system need only determine the target AS and devise a route to get into that target system
  - Once the datagram enters the target AS, the routers within that system can cooperate to deliver the datagram
  - The ERP is not concerned with, and does not know about, the details of the route

Examples

- Border Gateway Protocol (BGP)
- Open Shortest Path First (OSPF)

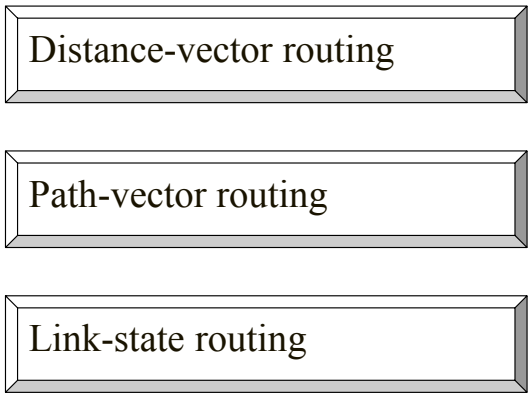
We refer to the protocol used to pass routing information between routers in different ASs as an exterior router protocol (ERP) .

We can expect that an ERP will need to pass less information than an IRP for the following reason. If a datagram is to be transferred from a host in one AS to a host in another AS, a router in the first system need only determine the target AS and devise a route to get into that target system. Once the datagram enters the target AS, the routers within that system can cooperate to deliver the datagram; the ERP is not concerned with, and does not know about, the details of the route followed within the target AS.

In the remainder of this section, we look at what are perhaps the most important examples of these two types of routing protocols: Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF) Protocol. But first, it is useful to look at a different way of characterizing routing protocols.

Approaches to Routing

- Internet routing protocols employ one of three approaches to gathering and using routing information:



Internet routing protocols employ one of three approaches to gathering and using routing information: distance-vector routing, link-state routing, and path-vector routing.

Distance-Vector Routing

- Requires that each node exchange information with its neighboring nodes
  - Two nodes are said to be neighbors if they are both directly connected to the same network
- Used in the first-generation routing algorithm for ARPANET
- Each node maintains a vector of link costs for each directly attached network and distance and next-hop vectors for each destination
- Routing Information Protocol (RIP) uses this approach

Distance-vector routing requires that each node (router or host that implements the routing protocol) exchange information with its neighboring nodes. Two nodes are said to be neighbors if they are both directly connected to the same network. This approach is that used in the first-generation routing algorithm for ARPANET, as described in Section 19.2. For this purpose, each node maintains a vector of link costs for each directly attached network and distance and next-hop vectors for each destination. The relatively simple Routing Information Protocol (RIP) uses this approach.

Distance-vector routing requires the transmission of a considerable amount of information by each router. Each router must send a distance vector to all of its neighbors, and that vector contains the estimated path cost to all networks in the configuration. Furthermore, when there is a significant change in a link cost or when a link is unavailable, it may take a considerable amount of time for this information to propagate through the internet.

Link-State Routing

- Designed to overcome the drawbacks of distance-vector routing
- When a router is initialized, it determines the link cost on each of its network interfaces
- The router then advertises this set of link costs to all other routers in the internet topology, not just neighboring routers
- From then on, the router monitors its link costs
- Whenever there is a significant change the router again advertises its set of link costs to all other routers in the configuration
- The OSPF protocol is an example
- The second-generation routing algorithm for ARPANET also uses this approach

Link-state routing is designed to overcome the drawbacks of distance-vector routing. When a router is initialized, it determines the link cost on each of its network interfaces. The router then advertises this set of link costs to all other routers in the internet topology, not just neighboring routers. From then on, the router monitors its link costs. Whenever there is a significant change (e.g., a link cost increases or decreases substantially, a new link is created, or an existing link becomes unavailable), the router again advertises its set of link costs to all other routers in the configuration.

Because each router receives the link costs of all routers in the configuration, each router can construct the topology of the entire configuration and then calculate the shortest path to each destination network. Having done this, the router can construct its routing table, listing the first hop to each destination. Because the router has a representation of the entire network, it does not use a distributed version of a routing algorithm, as is done in distance-vector routing. Rather, the router can use any routing algorithm to determine the shortest paths. In practice, Dijkstra's algorithm is used. The OSPF protocol is an example of a routing protocol that uses link-state routing. The second-generation routing algorithm for ARPANET also uses this approach.

Both link-state and distance-vector approaches have been used for interior router protocols. Neither approach is effective for an exterior router protocol.

Path-Vector Routing

- Alternative to dispense with routing metrics and simply provide information about which networks can be reached by a given router and the ASs visited in order to reach the destination network by this route
- Differs from a distance-vector algorithm in two respects:
  - The path-vector approach does not include a distance or cost estimate
  - Each block of routing information lists all of the ASs visited in order to reach the destination network by this route

An alternative, known as path-vector routing , is to dispense with routing metrics and simply provide information about which networks can be reached by a given router and the ASs that must be crossed to get there. The approach differs from a distance-vector algorithm in two respects: First, the path-vector approach does not include a distance or cost estimate. Second, each block of routing information lists all of the ASs visited in order to reach the destination network by this route.

Because a path vector lists the ASs that a datagram must traverse if it follows this route, the path information enables a router to perform policy routing. That is, a router may decide to avoid a particular path in order to avoid transiting a particular AS. For example, information that is confidential may be limited to certain kinds of ASs. Or a router may have information about the performance or quality of the portion of the internet that is included in an AS that leads the router to avoid that AS. Examples of performance or quality metrics include link speed, capacity, tendency to become congested, and overall quality of operation. Another criterion that could be used is minimizing the number of transit ASs.

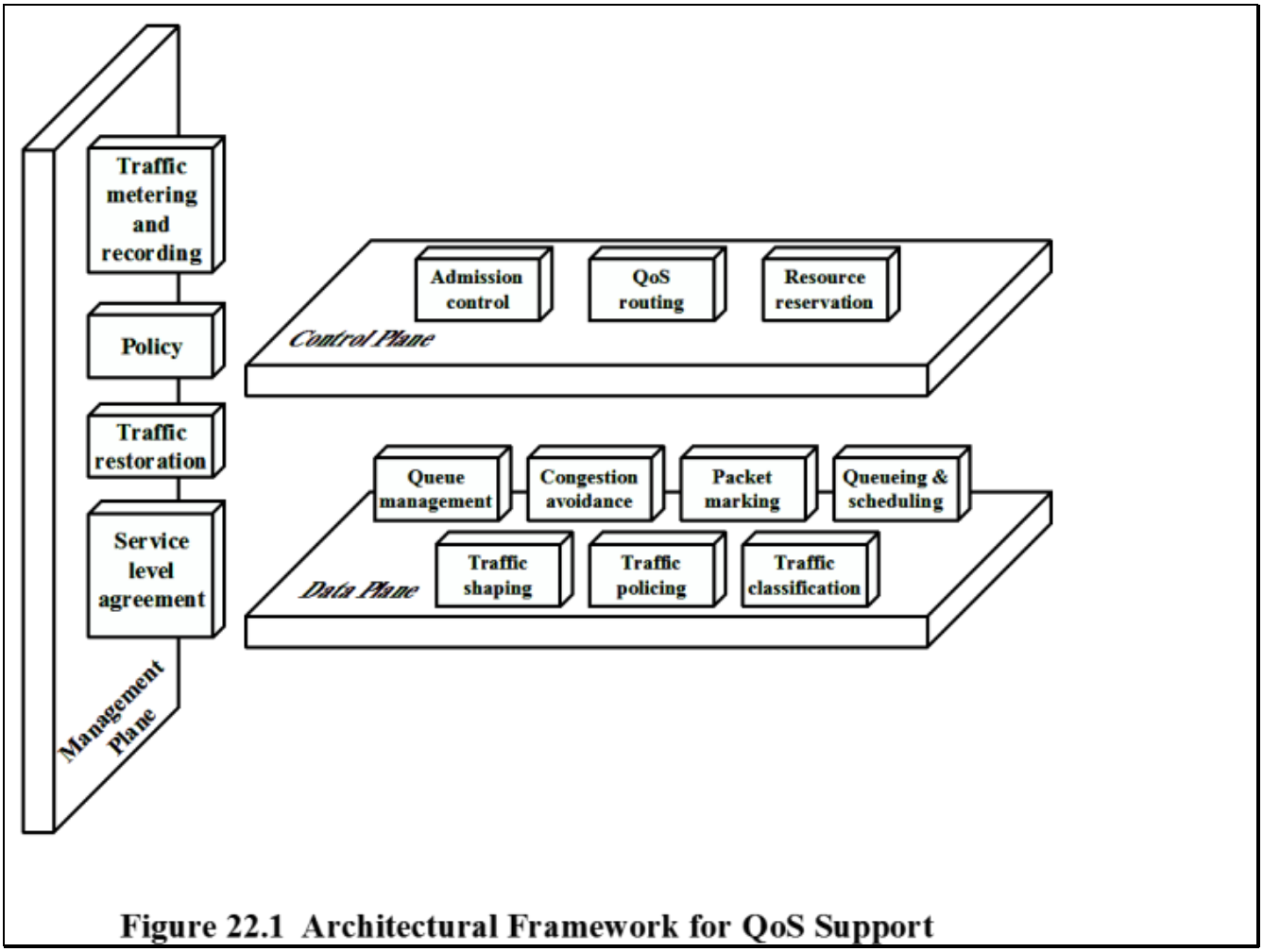


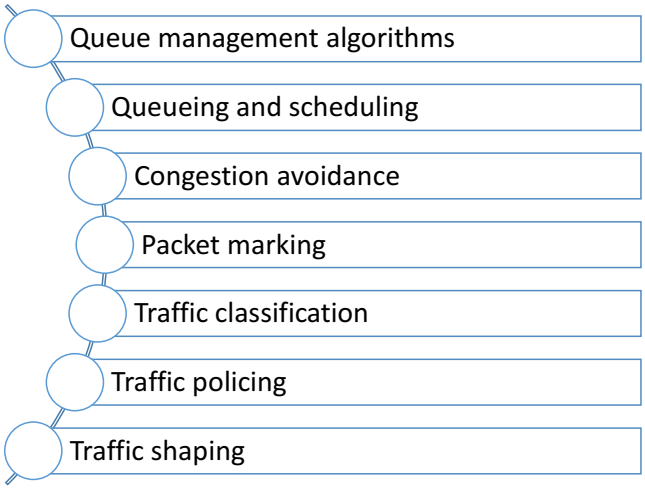
Figure 22.1 Architectural Framework for QoS Support

Before looking at the Internet standards that deal with provision of quality of service (QoS) in the Internet and private internetworks, it is useful to consider an overall architectural framework that relates the various elements that go into QoS provision. Such a framework has been developed by the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) as part of its Y series of Recommendations. The Recommendation, Y.1291 (An Architectural Framework for Support of Quality of Service in Packet Networks), gives a "big picture" overview of the mechanisms and services that comprise a QoS facility.

The Y.1291 framework comprises a set of generic network mechanisms for controlling the network service response to a service request, which can be specific to a network element, or for signaling between network elements, or for controlling and administering traffic across a network. Figure 22.1 shows the relationship among these elements, which are organized into three planes: data, control, and management.

Data Plane

- Includes those mechanisms that operate directly on flows of data



The data plane includes those mechanisms that operate directly on flows of data. We briefly comment on each.

Queue management algorithms manage the length of packet queues by dropping packets when necessary or appropriate. Active management of queues is concerned
primarily with congestion avoidance. In the early days of the Internet, the queue management discipline was to drop any incoming packets when the queue was full, referred to as the tail drop technique. There are a number of drawbacks to tail drop, including [BRAD98]:

1. There is no reaction to congestion until it is necessary to drop packets, whereas a more aggressive congestion avoidance technique would likely improve overall network performance.

2. Queues tend to be close to full, which causes an increase in packet delay through a network and which can result in a large batch of drop packets for bursty traffic, necessitating many packet retransmissions.

3. Tail drop may allow a single connection or a few flows to monopolize queue space, preventing other connections from getting room in the queue.

One noteworthy example of queue management is random early detection (RED). RED drops incoming packets probabilistically based on an estimated average queue size. The probability for dropping increases as the estimated average queue size grows. RED is described in Appendix P.

Queueing and scheduling algorithms, also referred to as queueing discipline algorithms, determine which packet to send next and are used primarily to manage the allocation of transmission capacity among flows. Queueing discipline is discussed in Section 22.2

Congestion avoidance deals with means for keeping the load of the network under its capacity such that it can operate at an acceptable performance level, not experiencing congestion collapse. Congestion avoidance is discussed in detail in Chapter 20.

Packet marking encompasses two distinct functions. First, packets may be marked by edge nodes of a network to indicate some form of QoS that the packet should receive. An example is the Differentiated Services (DS) field in the IPv4 and IPv6 packets (Figure 14.5) and the Traffic Class field in MPLS labels, discussed in Chapter 23. Such markings may be used by intermediate nodes to provide differential treatment to incoming packets. Packet marking can also be used to mark packets as nonconformant, which may be dropped later if congestion is experienced.

Traffic classification can be done on a packet or flow basis. All traffic assigned to a particular flow or other aggregate can then be treated similarly. The flow label in the IPv6 header (Figure 14.5b) can be used for traffic classification.

Traffic policing , discussed in Chapter 20, deals with the determination of whether the traffic being presented is on a hop-by-hop basis compliant with prenegotiated policies or contracts. Nonconformant packets may be dropped, delayed, or labeled as nonconformant. ITU-T Recommendation Y.1221 (Traffic Control and Congestion Control in IP-based Networks) recommends the use of token bucket to characterize traffic for purposes of traffic policing.

Traffic shaping , also discussed in Chapter 20, deals with controlling the rate and volume of traffic entering and transiting the network on a per-flow basis. The entity responsible for traffic shaping buffers nonconformant packets until it brings the respective aggregate in compliance with the traffic. The resulted traffic thus is not as bursty as the original and is more predictable. Y.1221 recommends the use of leaky bucket and/or token bucket for traffic shaping.

Control Plane

- Concerned with creating and managing the pathways through which user data flows
- It includes:
  - Admission control
  - QoS routing
  - Resource reservation

The control plane is concerned with creating and managing the pathways through which user data flows. It includes admission control, QoS routing, and resource reservation.

Admission control determines what user traffic may enter the network. This may be in part determined by the QoS requirements of a data flow compared to the current resource commitment within the network. RSVP, described in Section 22.3, implements this form of admission control. But beyond balancing QoS requests with available capacity to determine whether to accept a request, there are other considerations in admission control. Network managers and service providers must be able to monitor, control, and enforce use of network resources and services based on policies derived from criteria such as the identity of users and applications, traffic/bandwidth requirements, security considerations, and time-of-day/ week. RFC 2753 (A Framework for Policy-based Admission Control) discusses such policy-related issues.

QoS routing is a routing technique that determines a network path that is likely to accommodate the requested QoS of a flow. This contrasts with the philosophy of the routing protocols described in Chapter 19, which generally are looking for a least-cost path through the network. RFC 2386 (A Framework for QoS-based Routing in the Internet) provides an overview of the issues involved in QoS routing, which is an area of ongoing study.

Resource reservation is a mechanism that reserves network resources on demand for delivering desired network performance to a requesting flow. The resource reservation mechanism that has been implemented for the Internet is RSVP, described in Section 22.3.

Management Plane

- Contains mechanisms that affect both control plane and data plane mechanisms
- Includes:
  - Service level agreement (SLA)
  - Traffic metering and recording
  - Traffic restoration
  - Policy

The management plane contains mechanisms that affect both control plane and data plane mechanisms. The control plane deals with the operation, administration, and management aspects of the network. It includes service level agreement (SLA), traffic restoration, traffic metering and recording, and policy.

A service level agreement (SLA) typically represents the agreement between a customer and a provider of a service that specifies the level of availability, serviceability, performance, operation, or other attributes of the service. SLAs are discussed in Section 22.5.

Traffic metering and recording concerns monitoring the dynamic properties of a traffic stream using performance metrics such as data rate and packet loss rate. It involves observing traffic characteristics at a given network point and collecting and storing the traffic information for analysis and further action. Depending on the conformance level, a meter can invoke necessary treatment (e.g., dropping or shaping) for the packet stream. Section 22.6 discusses the types of metrics that are used in this function.

Traffic restoration refers to the network response to failures. This encompasses a number of protocol layers and techniques.

Policy is a category that refers to a set of rules for administering, managing, and controlling access to network resources. They can be specific to the needs of the service provider or reflect the agreement between the customer and service provider, which may include reliability and availability requirements over a period of time and other QoS requirements.

Slide 88

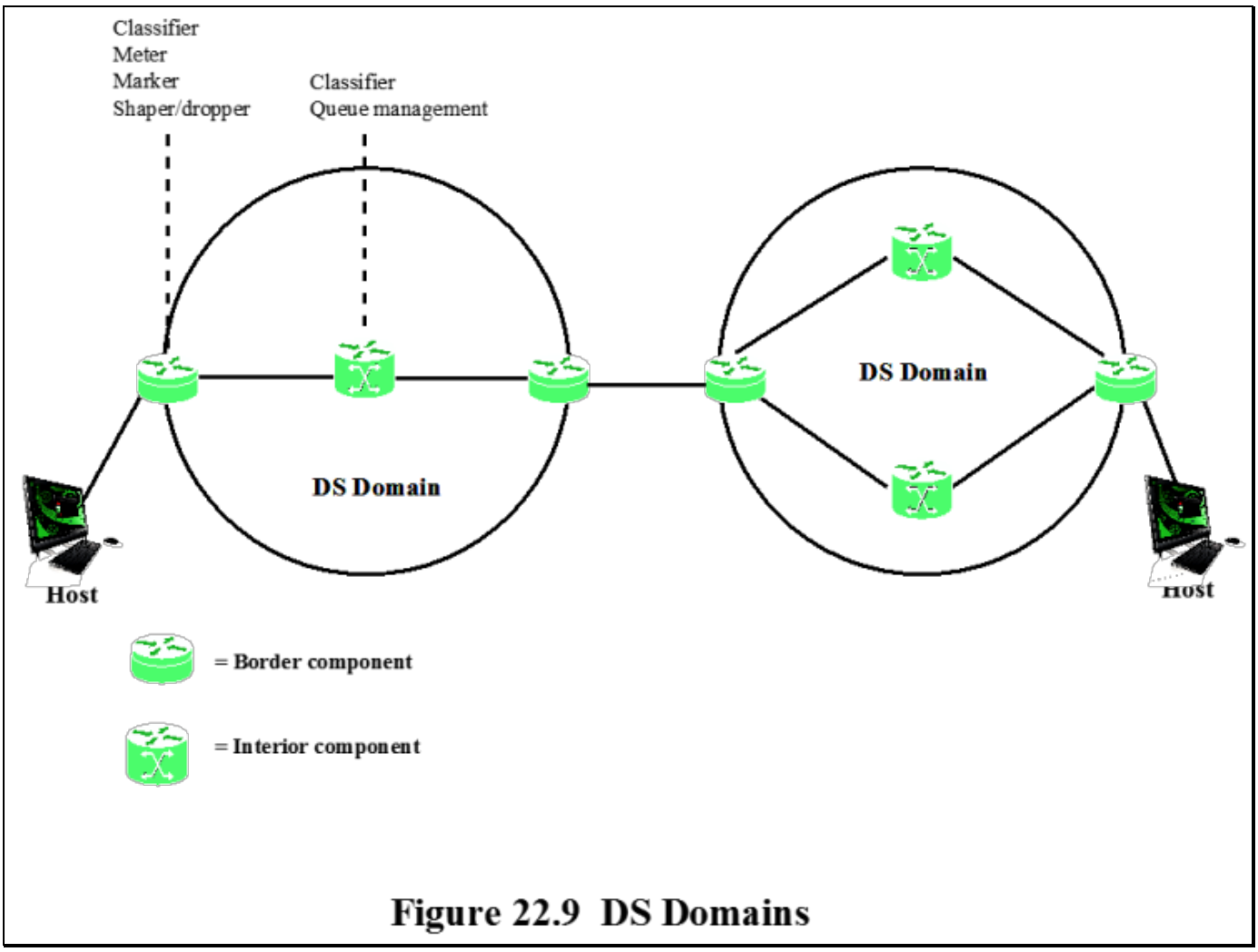


Figure 22.9 DS Domains

Figure 22.9 illustrates the type of configuration envisioned in the DS documents. A DS domain consists of a set of contiguous routers; that is, it is possible to get from any router in the domain to any other router in the domain by a path that does not include routers outside the domain. Within a domain, the interpretation of DS codepoints is uniform, so that a uniform, consistent service is provided.

Routers in a DS domain are either boundary nodes or interior nodes. Typically, the interior nodes implement simple mechanisms for handling packets based on their DS codepoint values. This includes queuing discipline to give preferential treatment depending on codepoint value, and packet dropping rules to dictate which packets should be dropped first in the event of buffer saturation. The DS specifications refer to the forwarding treatment provided at a router as per-hop behavior (PHB). This PHB must be available at all routers, and typically PHB is the only part of DS implemented in interior routers.

The boundary nodes include PHB mechanisms but more sophisticated traffic conditioning mechanisms are also required to provide the desired service. Thus, interior routers have minimal functionality and minimal overhead in providing the DS service, while most of the complexity is in the boundary nodes. The boundary node function can also be provided by a host system attached to the domain, on behalf of the applications at that host system.

The traffic conditioning function consists of five elements:

Classifier: Separates submitted packets into different classes. This is the foundation of providing differentiated services. A classifier may separate traffic only on the basis of the DS codepoint (behavior aggregate classifier) or based on multiple fields within the packet header or even the packet payload (multifield classifier).

Meter: Measures submitted traffic for conformance to a profile. The meter determines whether a given packet stream class is within or exceeds the service level guaranteed for that class.

Marker: Re-marks packets with a different codepoint as needed. This may be done for packets that exceed the profile; for example, if a given throughput is guaranteed for a particular service class, any packets in that class that exceed the throughput in some defined time interval may be re-marked for best effort handling. Also, re-marking may be required at the boundary between two DS domains. For example, if a given traffic class is to receive the highest supported priority, and this is a value of 3 in one domain and 7 in the next domain, then packets with a priority 3 value traversing the first domain are remarked as priority 7 when entering the second domain.

Shaper: Delays packets as necessary so that the packet stream in a given class does not exceed the traffic rate specified in the profile for that class.

Dropper: Drops packets when the rate of packets of a given class exceeds that specified in the profile for that class.

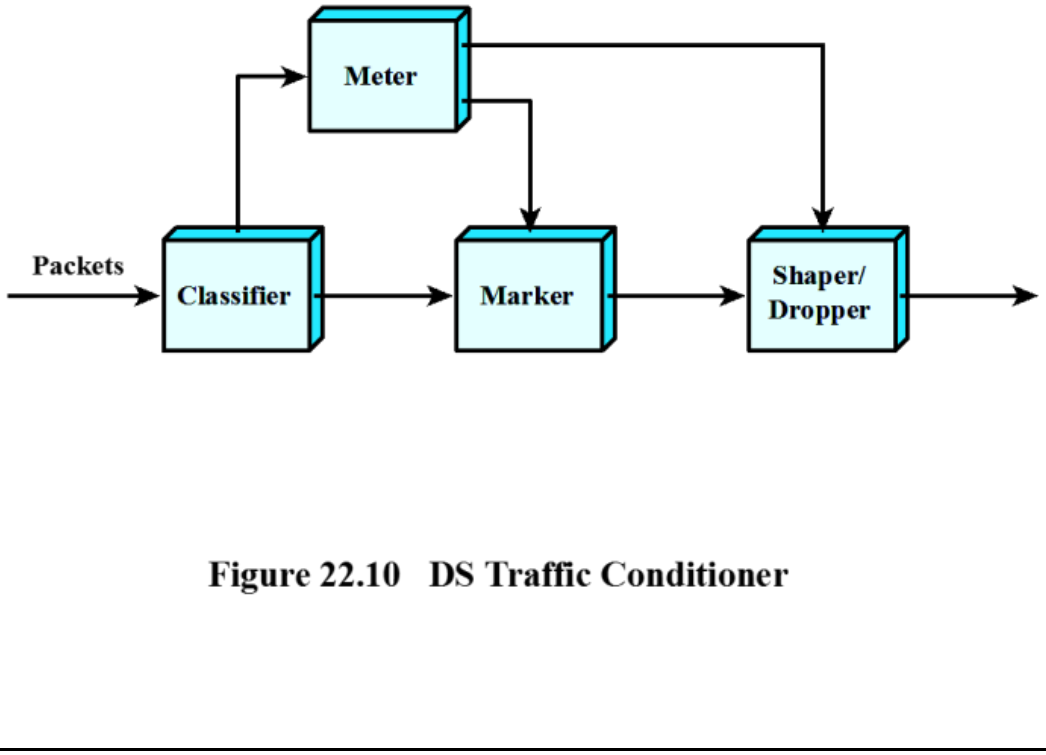


Figure 22.10 DS Traffic Conditioner

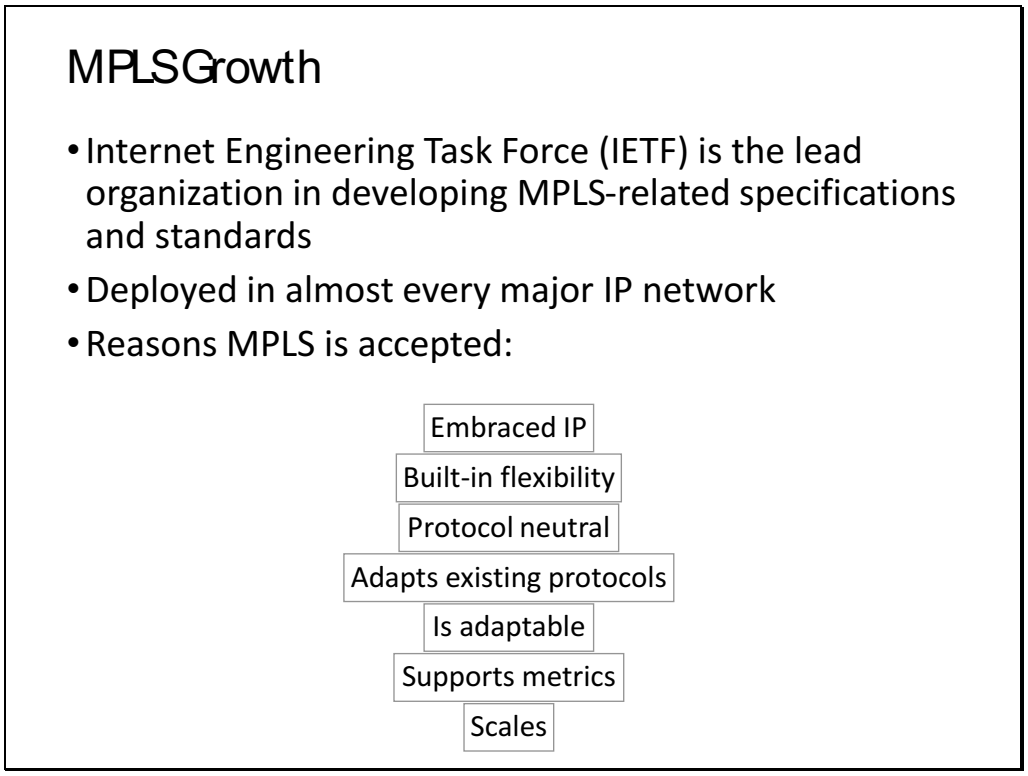
Figure 22.10 illustrates the relationship between the elements of traffic conditioning. After a flow is classified, its resource consumption must be measured. The metering function measures the volume of packets over a particular time interval to determine a flow's compliance with the traffic agreement. If the host is bursty, a simple data rate or packet rate may not be sufficient to capture the desired traffic characteristics. A token bucket scheme, such as that illustrated in Stallings DCC9e Figure 20.7, is an example of a way to define a traffic profile to take into account both packet rate and burstiness.

If a traffic flow exceeds some profile, several approaches can be taken. Individual packets in excess of the profile may be re-marked for lower-quality handling and allowed to pass into the DS domain. A traffic shaper may absorb a burst of packets in a buffer and pace the packets over a longer period of time. A dropper may drop packets if the buffer used for pacing becomes saturated.

Role of MPLS

- Efficient technique for forwarding and routing packets
- Designed with IP networks in mind
  - Can be used with any link-level protocol
- Fixed-length label encapsulates an IP packet or a data link frame
- MPLS label contains all information needed to perform routing, delivery, QoS, and traffic management functions
- Is connection oriented

In essence, MPLS is an efficient technique for forwarding and routing packets. MPLS was designed with IP networks in mind, but the technology can be used without IP to construct a network with any link-level protocol, including ATM and frame relay. In an ordinary packet-switching network packet switches must examine various fields within the packet heard to determine destination, route, quality of service (QoS), and any traffic management functions (such as discard or delay) that may be supported. Similarly, in an IP-based network, routers examine a number of fields in the IP header to determine these functions. In an MPLS network, a fixed-length label encapsulates an IP packet or a data link frame. The MPLS label contains all the information needed by an MPLS-enabled router to perform routing, delivery, QoS, and traffic management functions. Unlike IP, MPLS is connection oriented.



The IETF MPLS working group is the lead organization in developing MPLS-related specifications and standards. A number other working groups deal with MPLS-related issues. Briefly, the objectives of these groups are:

MPLS: Responsible for standardizing a base technology for using label switching and for the implementation of label-switched paths over various packet based link-level technologies, such as Packet-over-SONET, Frame Relay, ATM, and LAN technologies (e.g. all forms of Ethernet, Token Ring, etc.). This includes procedures and protocols for the distribution of labels between routers and encapsulation.

Common Control and Measurement Plane (CCAMP): responsible for defining a common control plane and a separate common measurement plane for physical path and core tunneling technologies of Internet and telecom service providers (ISPs and SPs), e.g. O-O and O-E-O optical switches, TDM Switches, Ethernet Switches, ATM and Frame Relay switches, IP encapsulation tunneling technologies, and MPLS.

Layer 2 Virtual Private Networks (l2vpn): Responsible for defining and specifying a limited number of solutions for supporting provider-provisioned Layer-2 Virtual Private Networks (L2VPNs). The objective is to support link layer interfaces, such as ATM and Ethernet, for providing VPNs for an MPLS-enabled IP packet switched network.

Layer 3 Virtual Private Networks (l3vpn): Responsible for defining and specifying a limited number of solutions for supporting provider-provisioned Layer-3 (routed) Virtual Private Networks (L3VPNs). Standardization includes supporting VPNs for end systems that have an IP interface over an MPLS-enabled IP packet switched network.

Pseudowire Emulation Edge to Edge (pwe3): Responsible for specifying protocols for pseudowire emulation over an MPLS network. A pseudowire emulates a point-to-point or point-to-multipoint link, and provides a single service that is perceived by its user as an unshared link or circuit of the chosen service.

Path Computation Element (PCE): Focuses on the architecture and techniques for constraint-based path computation.

The magnitude of this effort suggests the importance, indeed the coming dominance, of MPLS. A 2009 survey found that 84% of companies are now using MPLS for their wide area networks [REED09]. MPLS is deployed in almost every major IP network. [MARS09] lists the following reasons for the dramatic growth in MPLS acceptance.

1. MPLS embraced IP. In the early 1990s, the telecom industry was pinning their hopes on ATM as the network backbone technology of the future, and made substantial investment. But as Internet usage exploded, carriers needed to refocus their efforts. At the same time, IETF was looking for ways to make circuit-oriented ATM technology run over IP. The result was the MPLS effort, which was quickly adopted by ATM proponents.

2. MPLS has built-in flexibility in several ways. MPLS separates out a control component, which enables various applications to directly manipulate label bindings, and a forwarding component, which uses a simple label-swapping paradigm. Also, MPLS allows labels to be stacked, enabling multiple control planes to act on a packet.

- MPLS is protocol neutral. MPLS is designed to work in a multiple protocol environment. This enables MPLS to work with ATM, frame relay, SONET, or Ethernet at the core.

- MPLS is pragmatic. The architecture created only two new protocols: Label Distribution Protocol and Link Management Protocol. Everything else incorporates or adapts existing protocols.

5. MPLS is adaptable. MPLS has evolved over time to support new applications and services, including layer 2 and layer 3 virtual private networks (VPNs), Ethernet services, and traffic engineering.

6. MPLS supports metrics. MPLS allows carriers to collect a wide variety of statistics that can be used for network traffic trend analysis and planning. With MPLS, it's possible to measure traffic volume, latency and delay between two routers. Carriers also can measure traffic between hubs, metropolitan areas, and regions.

7. MPLS scales. For example, Verizon uses MPLS for several global networks including its public and private IP networks. Verizon's Public IP network spans 410 points of presence on six continents and spans more than 150 countries.

It is clear that MPLS will permeate virtually all areas of networking. Hence the need for students to gain a basic understanding of its technology and protocols.

Background of MPLS

- IP switching (Ipsilon)
- Tag switching (Cisco Systems)
- Aggregate route-based IP switching (IBM)
- Cascade (IP navigator)
- IETF set up the MPLS working group (1997)
  - First set of proposed standards (2001)
  - Key specification is RFC 3031

The roots of MPLS go back to a number of efforts in the mid-1990s to provide a comprehensive set of QoS and traffic engineering capabilities in IP-based networks. The first such effort to reach the marketplace was IP switching, developed by Ipsilon. To compete with this offering, numerous other companies announced their own products, notably Cisco Systems (tag switching), IBM (aggregate route-based IP switching), and Cascade (IP navigator). The goal of all these products was to improve the throughput and delay performance of IP, and all took the same basic approach: Use a standard routing protocol such as OSPF to define paths between endpoints and assign packets to these paths as they enter the network.

In response to these proprietary initiatives, the IETF set up the MPLS working group in 1997 to develop a common, standardized approach. The working group issued its first set of Proposed Standards in 2001. The key specification is RFC 3031. MPLS reduces the amount of per-packet processing required at each router in an IP-based network, enhancing router performance even more. More significantly, MPLS provides significant new capabilities in four areas that have ensured its popularity: QoS support, traffic engineering, virtual private networks, and multiprotocol support. Before turning to the details of MPLS, we briefly examine each of these.

Connection-Oriented QoS Support

- Connectionless networks cannot provide firm QoS commitments
- Has powerful traffic management and QoS capabilities
- MPLS imposes framework on an IP-based Internet
- Provides the foundation for sophisticated and reliable QoS traffic contracts

Network managers and users require increasingly sophisticated QoS support for a number of reasons. [SIKE00] lists the following key requirements:

Guarantee a fixed amount of capacity for specific applications, such as audio/video conference.

Control latency and jitter and ensure capacity for voice.

Provide very specific, guaranteed, and quantifiable service level agreements, or traffic contracts.

Configure varying degrees of QoS for multiple network customers.

A connectionless network, such as in IP-based internet, cannot provide truly firm QoS commitments. A differentiated service (DS) framework works in only a general way and upon aggregates of traffic from a number of sources. An IS framework, using RSVP, has some of the flavor of a connection-oriented approach but is nevertheless limited in terms of its flexibility and scalability. For services such as voice and video that require a network with high predictability, the DS and IS approaches, by themselves, may prove inadequate on a heavily loaded network. By contrast, a connection-oriented network, as we have seen, has powerful traffic management and QoS capabilities. MPLS imposes a connection-oriented framework on an IP-based internet and thus provides the foundation for sophisticated and reliable QoS traffic contracts.

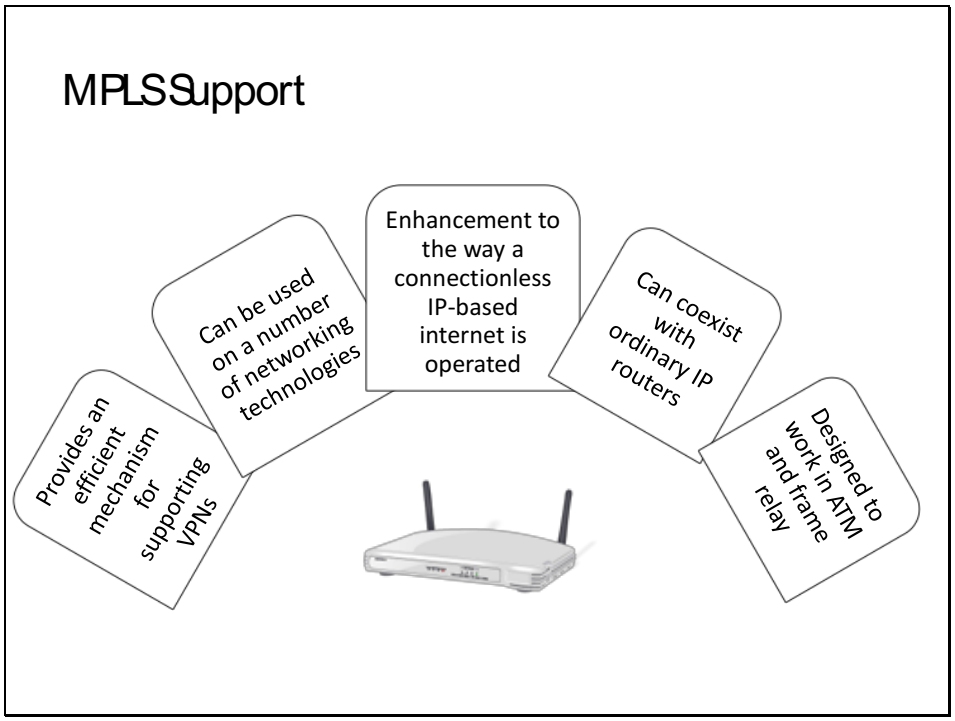
Traffic Engineering

- Ability to define routes dynamically, plan resource commitments on the basis of known demand, and optimize network utilization
- Effective use can substantially increase usable network capacity
- ATM provided strong traffic engineering capabilities prior to MPLS
- With basic IP there is a primitive form

MPLS
• Is aware of flows with QoS requirements • Possible to set up routes on the basis of flows • Paths can be rerouted intelligently

MPLS makes it easy to commit network resources in such a way as to balance the load in the face of a given demand and to commit to differential levels of support to meet various user traffic requirements. The ability to define routes dynamically, plan resource commitments on the basis of known demand, and optimize network utilization is referred to as **traffic engineering**. Prior to the advent of MPLS, the one networking technology that provided strong traffic engineering capabilities was ATM.

With the basic IP mechanism, there is a primitive form of automated traffic engineering. Specifically, routing protocols such as OSPF enable routers to dynamically change the route to a given destination on a packet-by-packet basis to try to balance load. But such dynamic routing reacts in a very simple manner to congestion and does not provide a way to support QoS. All traffic between two endpoints follows the same route, which may be changed when congestion occurs. MPLS, on the other hand, is aware of not just individual packets but flows of packets in which each flow has certain QoS requirements and a predictable traffic demand. With MPLS, it is possible to set up routes on the basis of these individual flows, with two different flows between the same endpoints perhaps following different routers. Further, when congestion threatens, MPLS paths can be rerouted intelligently. That is, instead of simply changing the route on a packet-by-packet basis, with MPLS, the routes are changed on a flow-by-flow basis, taking advantage of the known traffic demands of each flow. Effective use of traffic engineering can substantially increase usable network capacity.



MPLS provides an efficient mechanism for supporting VPNs. With a VPN, the traffic of a given enterprise or group passes transparently through an internet in a way that effectively segregates that traffic from other packets on the internet, proving performance guarantees and security.

MPLS can be used on a number of networking technologies. Our focus in this chapter is on IP-based internets, and this is likely to be the principal area of use. MPLS is an enhancement to the way a connectionless IP-based internet is operated, requiring an upgrade to IP routers to support the MPLS features. MPLS-enabled routers can coexist with ordinary IP routers, facilitating the introduction of evolution to MPLS schemes. MPLS is also designed to work in ATM and frame relay networks. Again, MPLS-enabled ATM switches and MPLS-enabled frame relay switches can be configured to coexist with ordinary switches. Furthermore, MPLS can be used in a pure IP-based internet, a pure ATM network, a pure frame relay network, or an internet that includes two or even all three technologies. This universal nature of MPLS should appeal to users who currently have mixed network technologies and seek ways to optimize resources and expand QoS support.

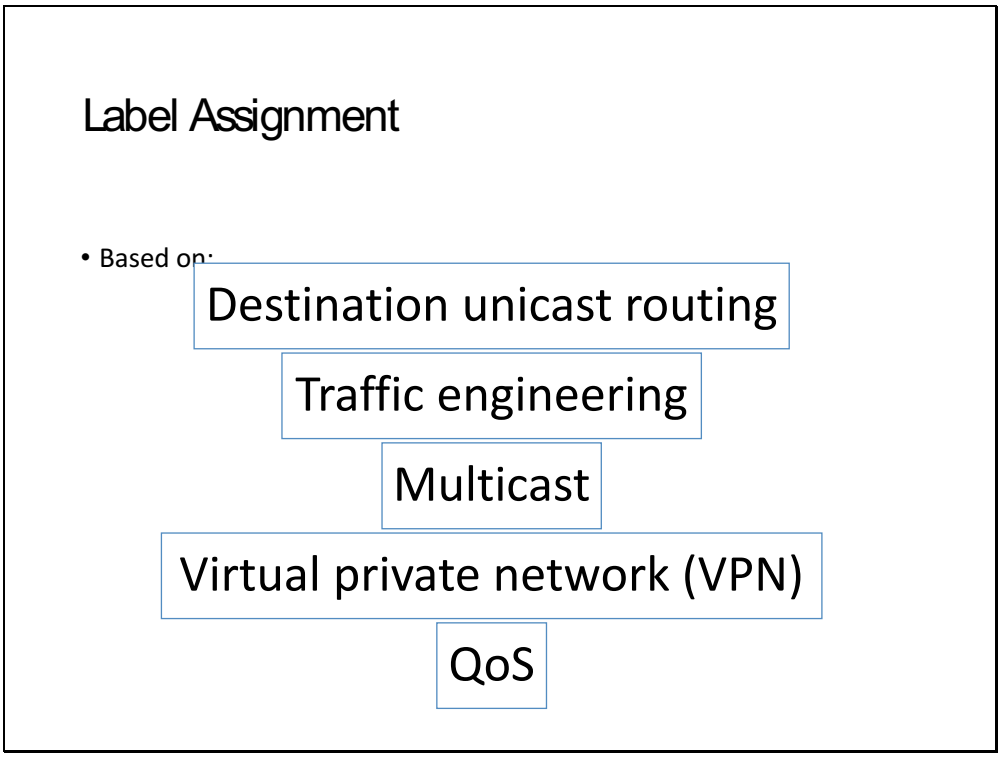
For the remainder of this discussion, we focus on the use of MPLS in IP-based internets, with brief comments about formatting issues for ATM and frame relay networks.

MPLS Operation

- Label switching routers (LSRs)
  - Nodes capable of switching and routing packets on the basis of label
- Labels define a flow of packets between two endpoints
- Assignment of a particular packet is done when the packet enters the network of MPLS routers
- Connection-oriented technology

An MPLS network or internet consists of a set of nodes, called **label switching routers** (LSRs) capable of switching and routing packets on the basis of which a label has been appended to each packet. Labels define a flow of packets between two endpoints or, in the case of multicast, between a source endpoint and a multicast group of destination endpoints. For each distinct flow, called a **forwarding equivalence class** (FEC), a specific path through the network of LSRs is defined, called a **label switched path** (LSP). In essence, an FEC represents a group of packets that share the same transport requirements. All packets in an FEC receive the same treatment en route to the destination. These packets follow the same path and receive the same QoS treatment at each hop. In contrast to the forwarding in ordinary IP networks, the assignment of a particular packet to a particular FEC is done just once, when the packet enters the network of MPLS routers.

Thus, MPLS is a connection-oriented technology. Associated with each FEC is a traffic characterization that defines the QoS requirements for that flow. The LSRs need not examine or process the IP header but rather simply forward each packet based on its label value. Each LSR builds a table, called a **label information base** (LIB), to specify how a packet must be treated and forwarded. Thus, the forwarding process is simpler than with an IP router.



Label assignment decisions (i.e., the assignment of a packet to a give FEC and hence a given LSP) may be based on the following criteria:

Destination unicast routing: in the absence of other criteria, packets flowing from one source to one destination may be assigned to the same FEC.

Traffic engineering: packet flows may be split up or aggregated to accommodate traffic engineering requirements.

Multicast: multicast routes through the network may be defined.

Virtual private network (VPN): Traffic among end systems for a particular customer may be segregated from other traffic on a public MPLS network by means of a dedicated set of LSPs.

QoS: Traffic may be assigned different FECs for different QoS requirements.

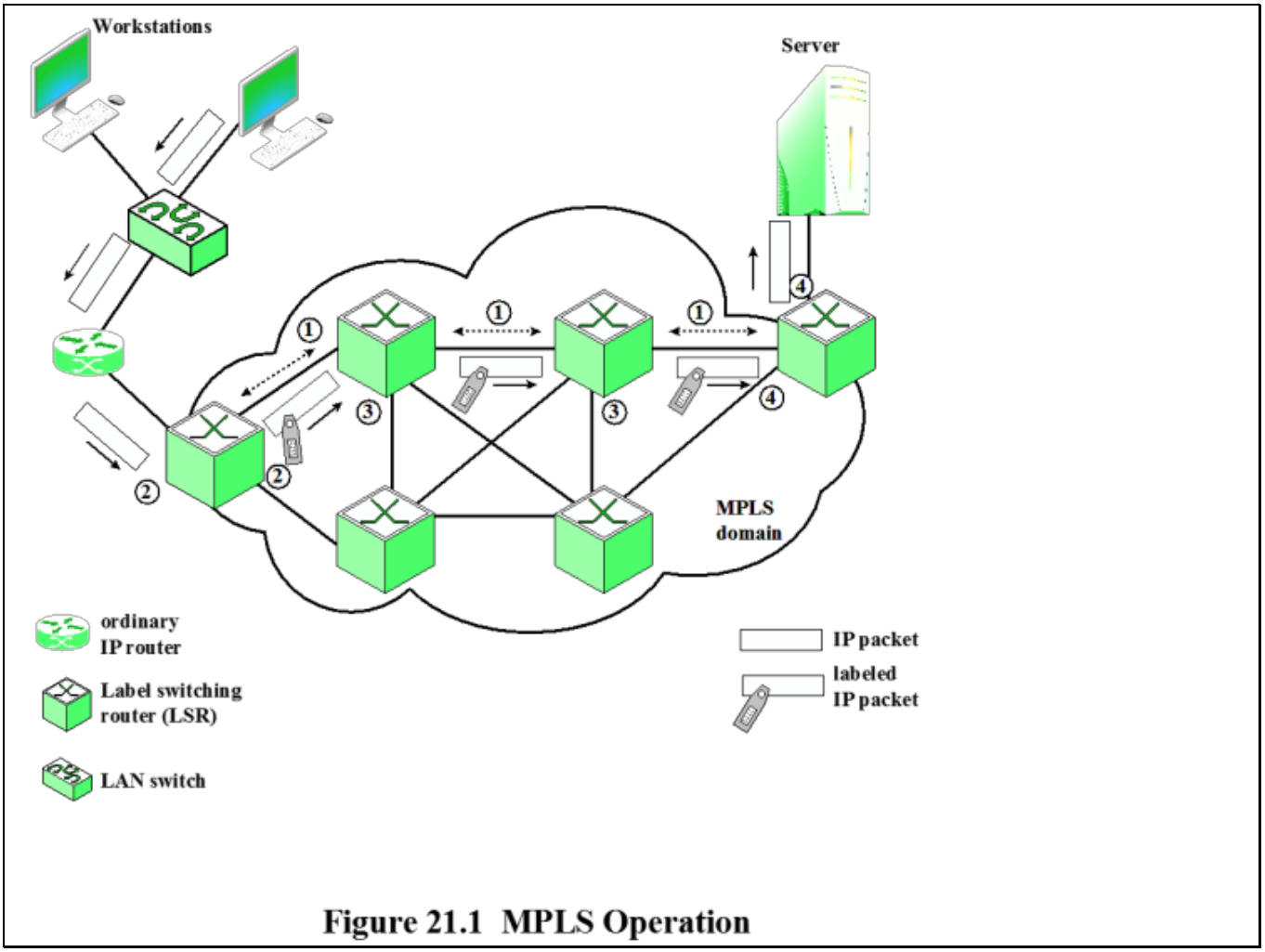


Figure 21.1 MPLS Operation

Figure 23.1 depicts the operation of MPLS within a domain of MPLS-enabled routers. The following are key elements of the operation:

1. Prior to the routing and delivery of packets in a given FEC, a path through the network, known as a **label switched path** (LSP), must be defined and the QoS parameters along that path must be established. The QoS parameters determine (1) how much resources to commit to the path, and (2) what queuing and discarding policy to establish at each LSR for packets in this FEC. To accomplish these tasks, two protocols are used to exchange the necessary information among routers:

(a) An interior routing protocol, such as OSPF, is used to exchange reachability and routing information.

(b) Labels must be assigned to the packets for a particular FEC. Because the use of globally unique labels would impose a management burden and limit the number of usable labels, labels have local significance only, as discussed subsequently. A network operator can specify explicit routes manually and assign the appropriate label values. Alternatively, a protocol is used to determine the route and establish label values between adjacent LSRs. Either of two protocols can be used for this purpose: the Label Distribution Protocol (LDP) or an enhanced version of RSVP. LDP is now considered the standard technique, with the RSVP approach deprecated.

2. A packet enters an MPLS domain through an ingress edge LSR, where it is processed to determine which network-layer services it requires, defining its QoS. The LSR assigns this packet to a particular FEC, and therefore a particular LSP; appends the appropriate label to the packet; and forwards the packet. If no LSP yet exists for this FEC, the edge LSR must cooperate with the other LSRs in defining a new LSP.

- Within the MPLS domain, as each LSR receives a labeled packet, it

  (a) Removes the incoming label and attaches the appropriate outgoing label to the packet

  (b) Forwards the packet to the next LSR along the LSP

4. The egress edge LSR strips the label, reads the IP packet header, and forwards the packet to its final destination.

Several key features of MLSP operation can be noted at this point:

1. An MPLS domain consists of a contiguous, or connected, set of MPLS-enabled routers. Traffic can enter or exit the domain from an endpoint on a directly connected network, as shown in the upper-right corner of Figure 21.1. Traffic may also arrive from an ordinary router that connects to a portion of the internet not using MPLS, as shown in the upper-left corner of Figure 21.1.

2. The FEC for a packet can be determined by one or more of a number of parameters, as specified by the network manager. Among the possible parameters:

—Source and/or destination IP addresses or IP network addresses

—Source and/or destination port numbers

—IP protocol ID

—Differentiated services codepoint

—IPv6 flow label

3. Forwarding is achieved by doing a simple lookup in a predefined table that maps label values to next hop addresses. There is no need to examine or process the IP header or to make a routing decision based on destination IP address. This not only makes it possible to separate types of traffic, such as best effort traffic from mission-critical traffic, it also renders an MPLS solution highly scalable. MPLS decouples packet forwarding from IP header information because it uses different mechanisms to assign labels. Labels have local significance only; therefore, it's nearly impossible to run out of labels. This characteristic is essential to implementing advanced IP services such as QoS, VPNs, and traffic engineering

4. A particular per-hop behavior (PHB) can be defined at an LSR for a given FEC. The PHB defines the queuing priority of the packets for this FEC and the discard policy.

5. Packets sent between the same endpoints may belong to different FECs. Thus, they will be labeled differently, will experience different PHB at each LSR, and may follow different paths through the network.

Slide 99

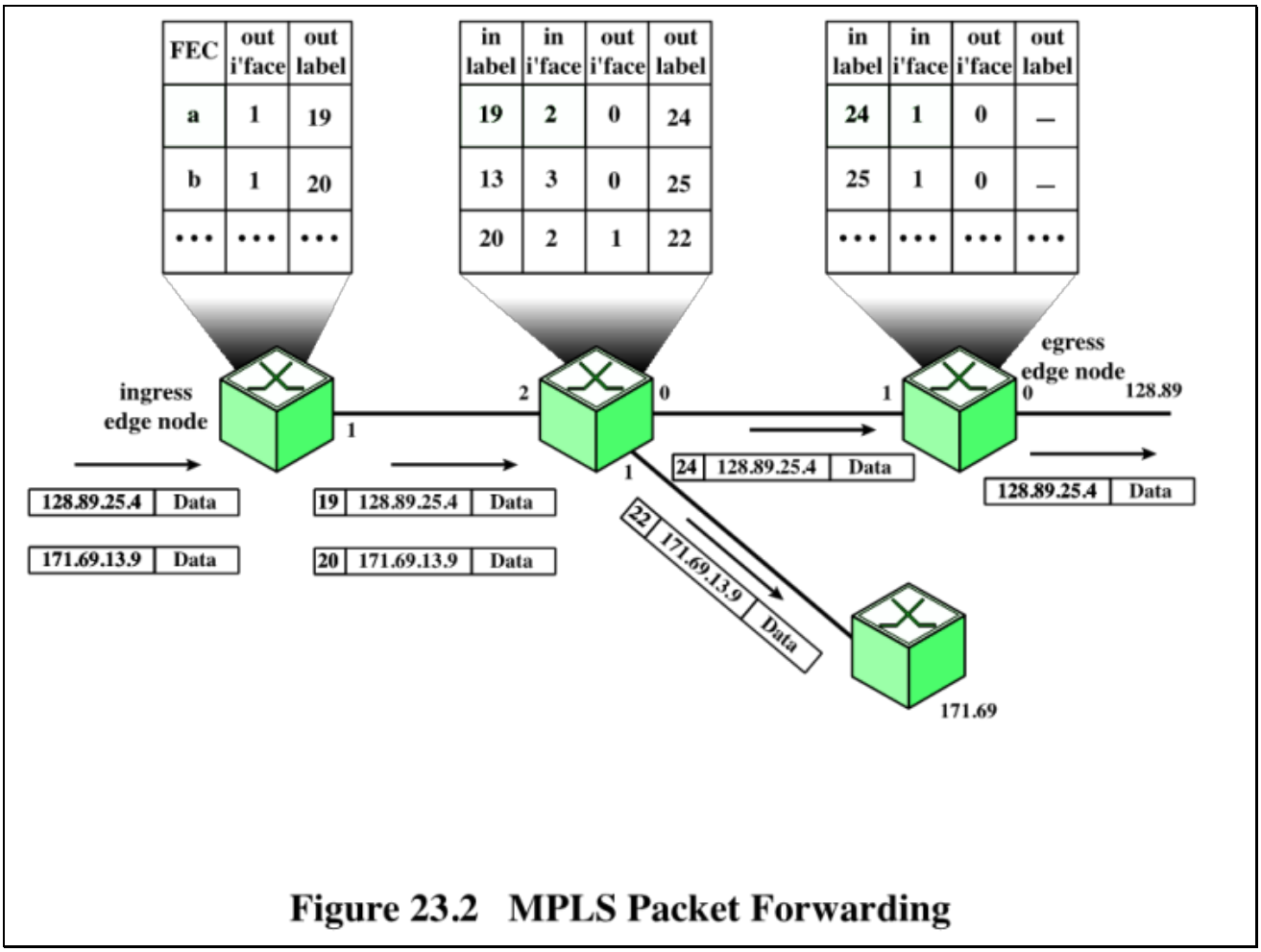


Figure 23.2 MPLS Packet Forwarding

Figure 23.2 shows the label-handling and forwarding operation in more detail. Each LSR maintains a forwarding table for each LSP passing through the LSR. When a labeled packet arrives, the LSR indexes the forwarding table to determine the next hop. For scalability, as was mentioned, labels have local significance only. Thus, the LSR removes the incoming label from the packet and attaches the matching outgoing label before forwarding the packet. The ingress edge LSR determines the FEC for each incoming unlabeled packet and, on the basis of the FEC, assigns the packet to a particular LSP, attaches the corresponding label, and forwards the packet. In this example, the first packet arrives at the edge LSR, which reads the IP header for the destination address prefix, 128.89. The LSR then looks up the destination address in the switching table, inserts a label with a 20-bit label value of 19, and forwards the labeled packet out interface 1. This interface is attached via a link to a core LSR, which receives the packet on its interface 2. The LSR in the core reads the label and looks up its match in its switching table, then replaces label 19 with label 24, and forwards it out interface 0. The egress LSR reads and looks up label 4 in its table, which says to strip the label and forward the packet out interface 0.

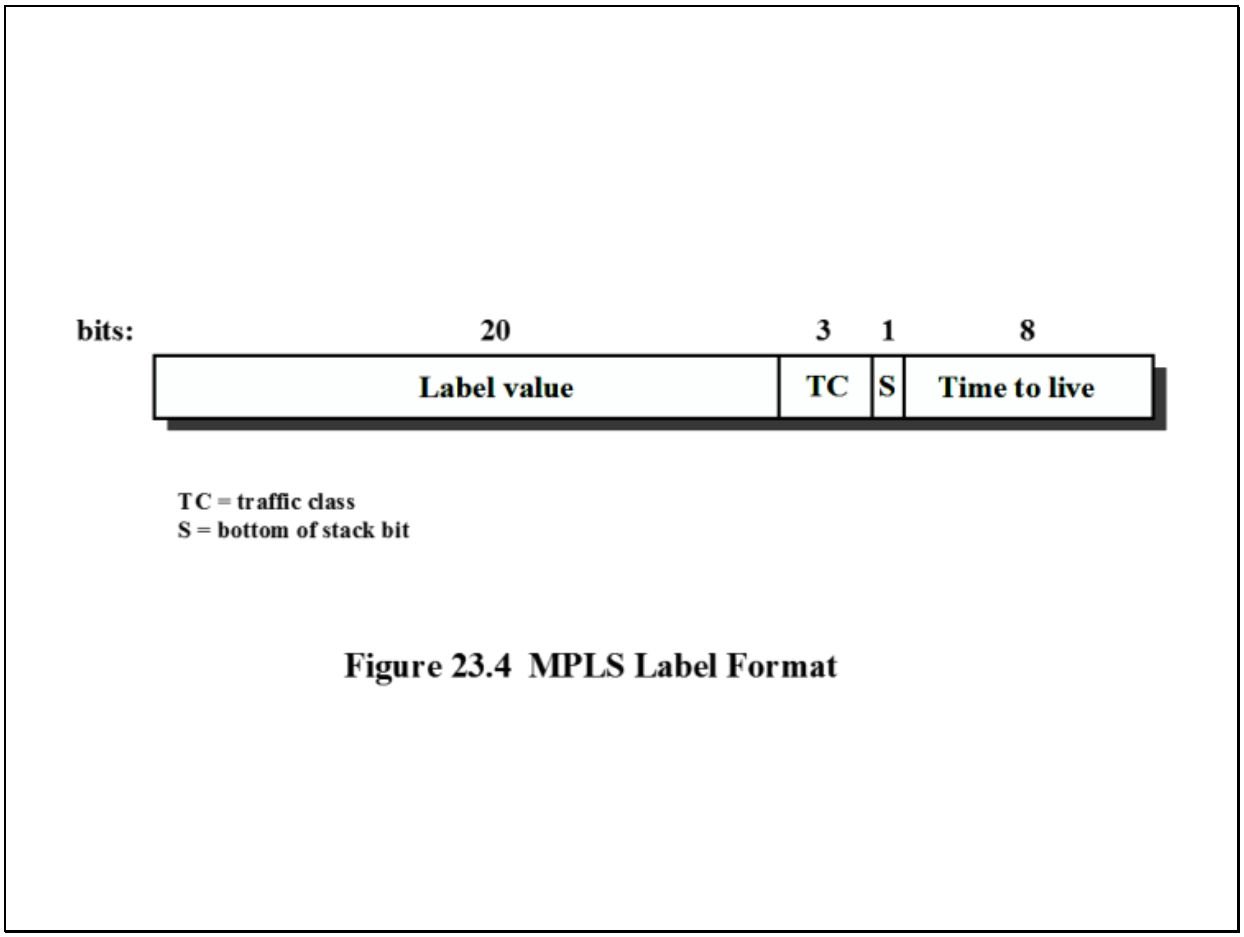


Figure 23.4 MPLS Label Format

An MPLS label is a 32-bit field consisting of the following elements (Figure 23.4), defined in RFC 3032:

Label value: Locally significant 20-bit label. Values 0 through 15 are reserved.

Traffic class (TC): 3 bits used to carry traffic class information.

S: Set to one for the oldest entry in the stack, and zero for all other entries. Thus, this bit marks the bottom of the stack.

Time to live (TTL): 8 bits used to encode a hop count, or time to live, value.

Slide 101

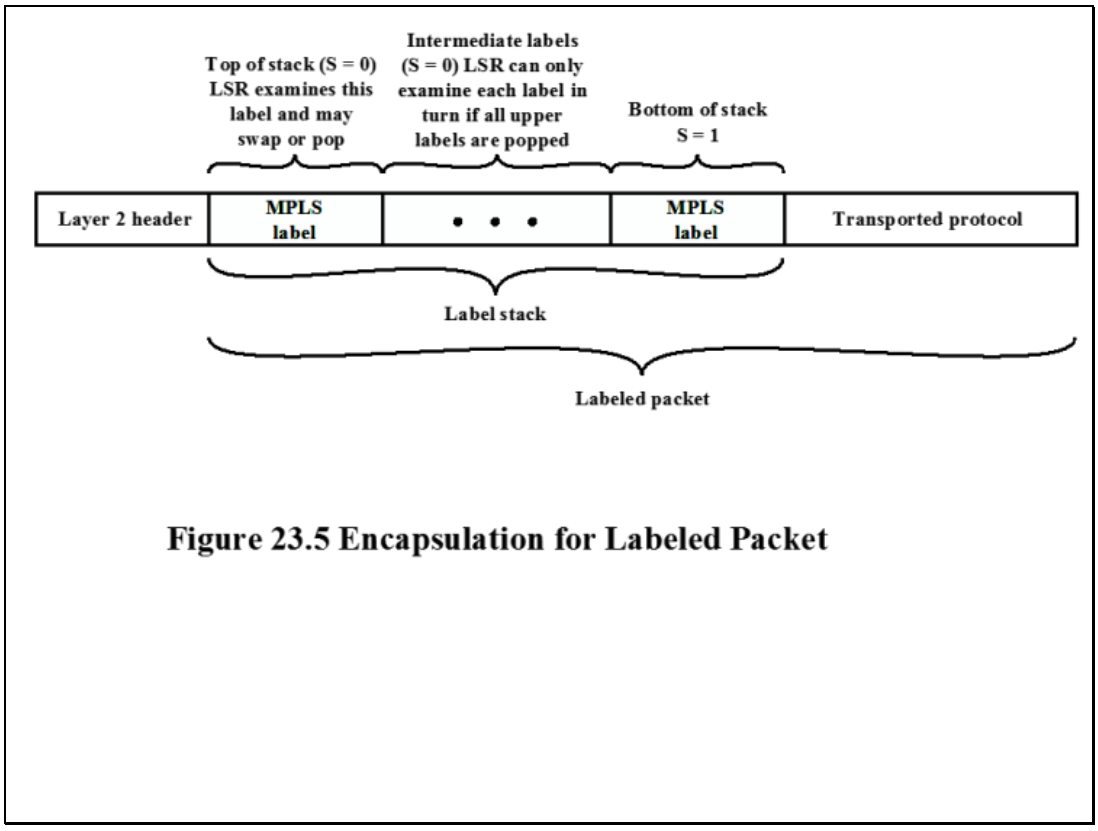


Figure 23.5 Encapsulation for Labeled Packet

The label stack entries appear after the data link layer headers, but before any network layer headers. The top of the label stack appears earliest in the packet (closest to the data link header), and the bottom appears latest (closest to the network layer header), as shown in Figure 23.5. The network layer packet immediately follows the label stack entry that has the S bit set.

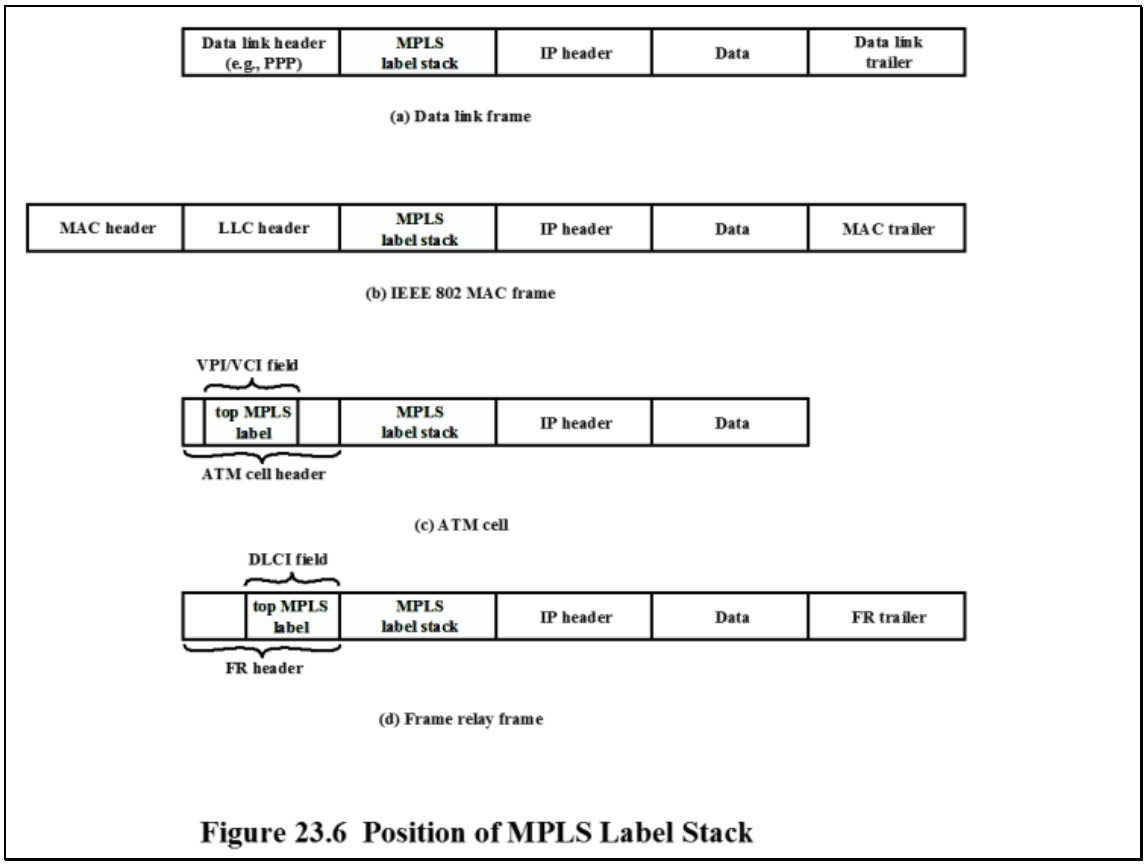


Figure 23.6 Position of MPLS Label Stack

In data link frames, such as for PPP (point-to-point protocol), the label stack appears between the IP header and the data link header (Figure 23.6a). For an IEEE 802 frame, the label stack appears between the IP header and the LLC (logical link control) header (Figure 23.6b).

If MPLS is used over a connection-oriented network service, a slightly different approach may be taken, as shown in Figures 23.6c and d. For ATM cells, the label value in the topmost label is placed in the VPI/VCI field in the ATM cell header. The entire top label remains at the top of the label stack, which is inserted between the cell header and the IP header. Placing the label value in the ATM cell header facilitates switching by an ATM switch, which would, as usual, only need to look at the cell header. Similarly, the topmost label value can be placed in the DLCI (data link connection identifier) field of a frame relay header. Note that in both these cases, the time to live field is not visible to the switch and so is not decremented. The reader should consult the MPLS specifications for the details of the way this situation is handled.

Slide 103

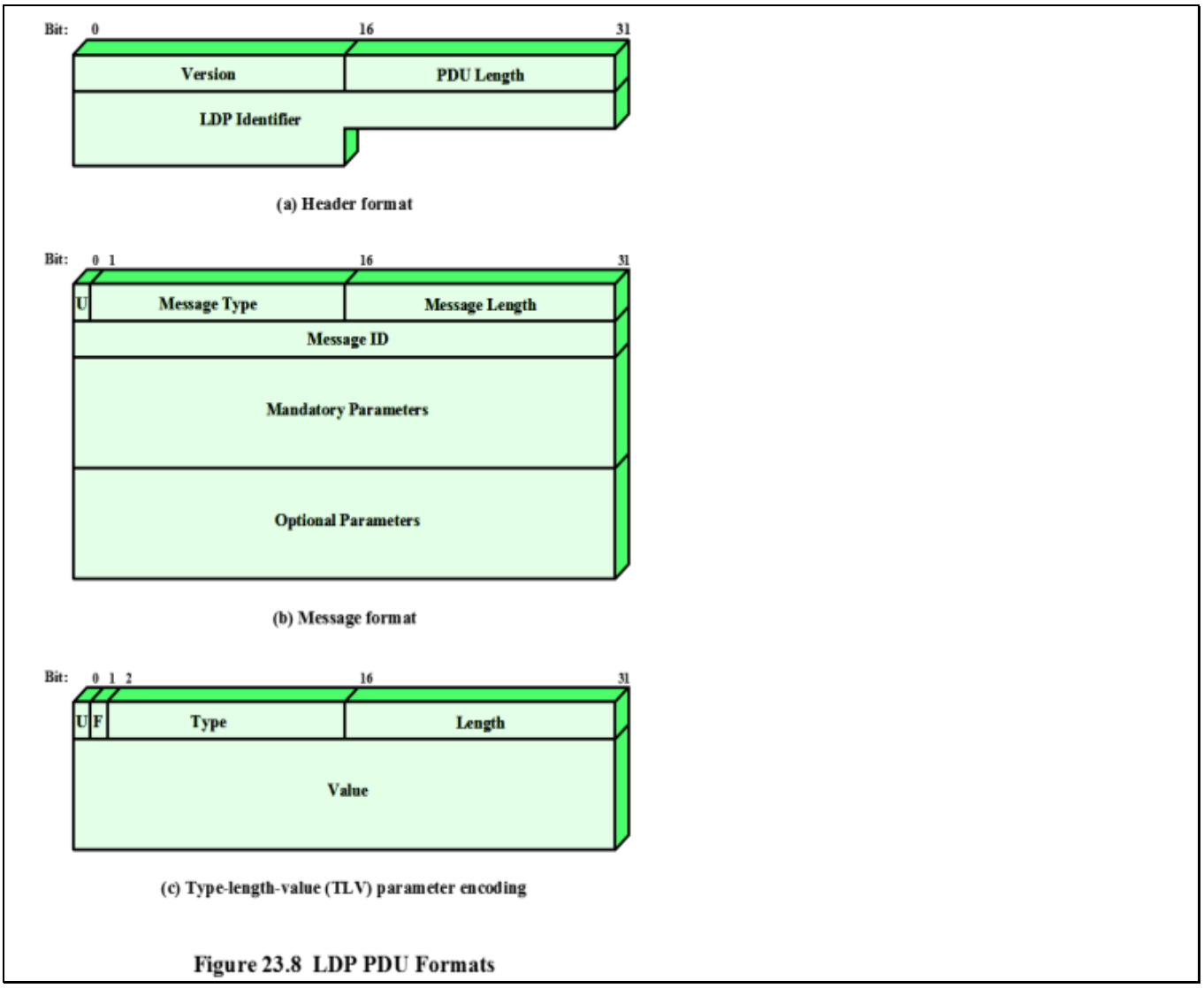


Figure 23.8 LDP PDU Formats

Figure 23.8 illustrates the format of LDP messages. Each LDP protocol data unit (PDU) consists of an LDP header followed by one or more LDP messages. The header contains three fields:

Version: Version number for this protocol. The current version is 1.

PDU length: Length of this PDU in octets.

LDP identifier: Identifies the LSR uniquely.

Each message consists of the following fields:

U bit: Indicates how to handle a message of an unknown type (forward or discard). This facilitates backward compatibility.

Message type: Identifies the specific type of message (e.g., Hello).

Message length: Length of this message in octets.

Message ID: Identifies this specific message with a sequence number. This is used to facilitate identifying subsequent Notification messages that may apply to this message.

Mandatory parameters: Variable length set of required message parameters. Some messages have no required parameters.

Optional parameters: Variable length set of optional message parameters. Many messages have no optional parameters.

Each parameter has a type-length-value (TLV) format. An LDP TLV is encoded as a 2 octet field that uses 14 bits to specify a Type and 2 bits to specify behavior when an LSR doesn't recognize the Type, followed by a 2 octet Length field, followed by a variable length Value field.

Slide 104

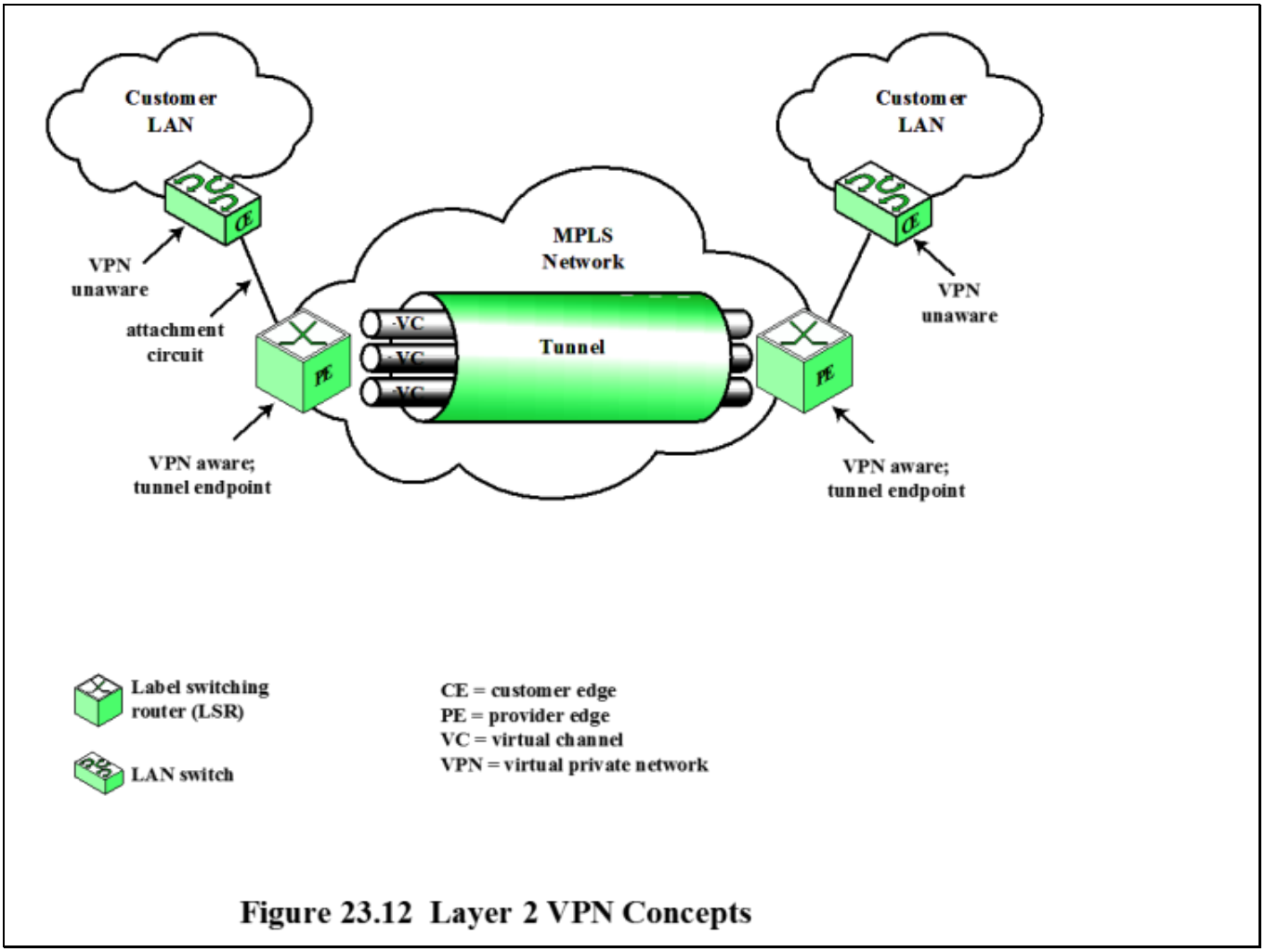


Figure 23.12 Layer 2 VPN Concepts

With a layer 2 VPN, there is mutual transparency between the customer network and the provider network. In effect, the customer requests a mesh of unicast LSPs among customer switches that attach to the provider network. Each LSP is viewed as a layer 2 circuit by the customer. In a L2VPN, the provider's equipment forwards customer data based on information in the Layer 2 headers, such as an Ethernet MAC address, an ATM virtual channel identifier, or a frame relay data link connection identifier.

Figure 23.12 depicts key elements in a L2VPN. Customers connect to the provider by means of a layer 2 device, such as an Ethernet switch, or a frame relay or ATM node; the customer device that connects to the MPLS network is generally referred to as a customer edge (CE) device. The MPLS edge router is referred to as a provider edge (PE) device. The link between the CE and the PE operates at the link layer (e.g., Ethernet), and is referred to as an attachment circuit. The MPLS network then sets up an LSP that acts as a tunnel between two edge routers (i.e., two PEs) that attach to two networks of the same enterprise. This tunnel can carry multiple virtual channels (VCs) using label stacking.

When a link-layer frame arrives at the PE from the CE, the PE creates an MPLS packet. The PE pushes a label that corresponds to the VC assigned to this frame. Then the PE pushes a second label onto the label stack for this packet that corresponds to the tunnel between the source and destination PE for this VC. The packet is then routed across the LSP associated with this tunnel, using the top label for label switched routing. At the destination edge, the destination PE pops

the tunnel label and examines the VC label. This tells the PE how to construct a link layer frame to deliver the payload across to the destination CE.

If the payload of the MPLS packet is, for example, an ATM AAL5 PDU, the VC label will generally correspond to a particular ATM VC at PE2. That is, PE2 needs to be able to infer from the VC label the outgoing interface and the VPI/VCI (Virtual Path Identifier/Virtual Circuit Identifier) value for the AAL5 PDU. If the payload is a Frame Relay PDU, then PE2 needs to be able to infer from the VC label the outgoing interface and the DLCI (Data Link Connection Identifier) value. If the payload is an Ethernet frame, then PE2 needs to be able to infer from the VC label the outgoing interface, and perhaps the VLAN identifier. This process is unidirectional, and will be repeated independently for bidirectional operation.

The virtual circuits in the tunnel can all belong to a single enterprise, or it is possible for a single tunnel to manage virtual circuits from multiple enterprises. In any case, from the point of view of the customer, a virtual circuit is a dedicated link-layer point-to-point channel. If multiple VCs connect a PE to a CE this is logically the multiplexing of multiple link-layer channels between the customer and the provider.