Data Communications and Networking Fourth Edition

Chapter 23

Process-to-Process Delivery: UDP and TCP

23-1 PROCESS-TO-PROCESS DELIVERY

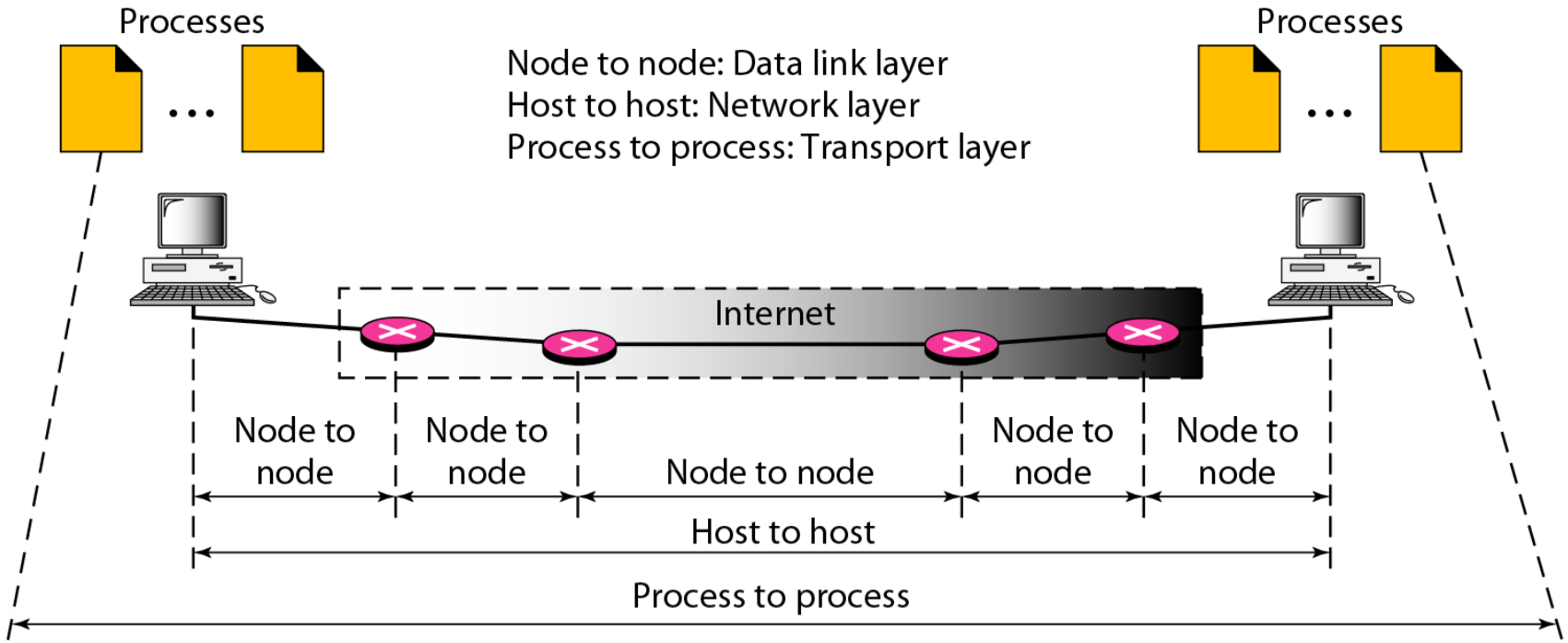
The transport layer is responsible for process-to-process delivery—the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship, as we will see later.

Process-to-process Delivery

- The data link layer is responsible for delivery of frames between two neighboring nodes over a link.
  - This is called node-to-node delivery.
- The network layer is responsible for delivery of datagram between two hosts.
  - This is called host-to-host delivery.

Process-to-process Delivery(cont'd)

- Transport layer is responsible for process-to-process delivery, the delivery of a packet, part of a message, from one process to another.
- Two processes communicate in a client-server relationship



Process-to-process Delivery(cont'd)

- **Client/Server Paradigm**
  - **The most common process-to-process communication is though the Client/Server Paradigm.**
    - **Client : A process on the local host**
    - **Server : A process on the remote host to provide services.**
  - **Both processes (Client and Server) have the same name.**
    - **Ex.) day time Client process and day time Server process.**
  - **For communication, we must define the following;**
    - **Local host**
    - **Local process**
    - **Remote host**
    - **Remote process**

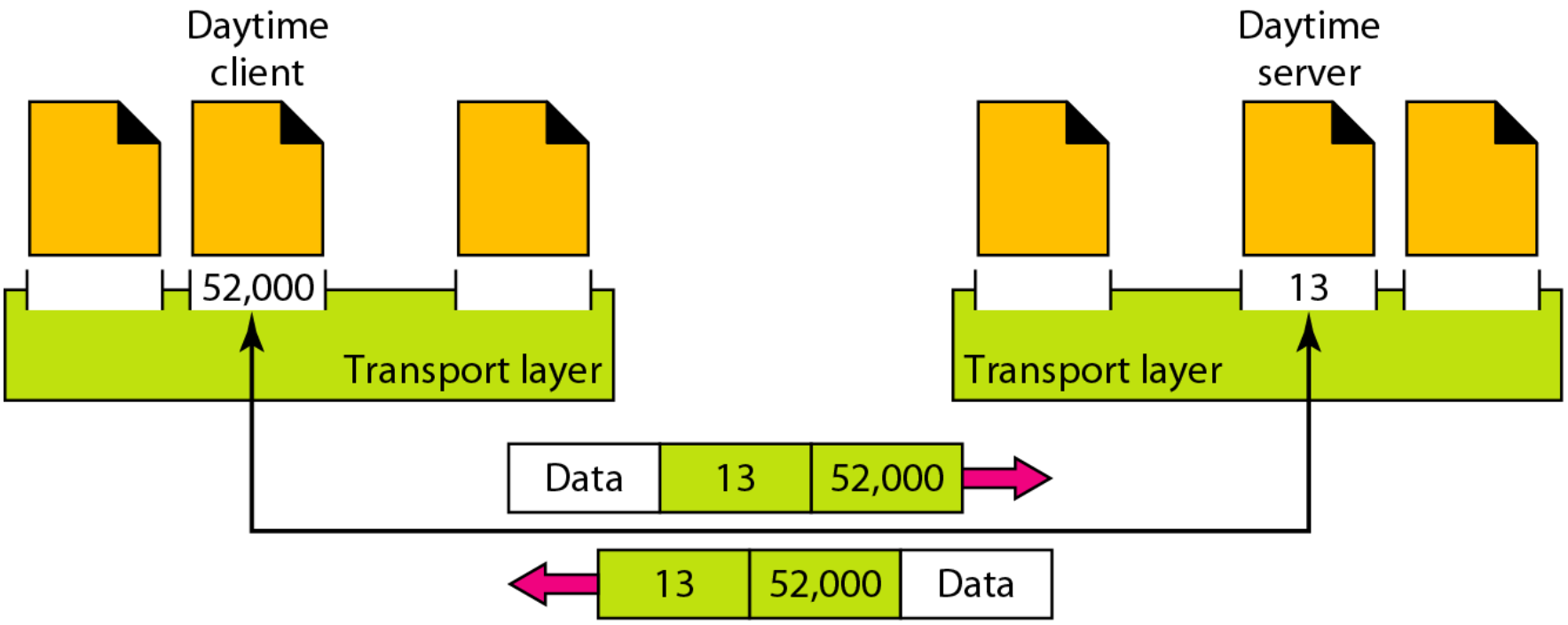
Process-to-process Delivery(cont'd)

- Addressing
  - A frame in the data link layer needs a destination MAC address.
  - At the network layer, we need an IP address.
  - At the transport layer, we need a transport layer address, called a Port number to choose among multiple processes running on the host.
  - In the Internet model
    - The port No. : 0~65,535 (16-bit integer)
  - Well-known port number
    - Universal port No. for server (fixed value, ranging : 0 ~1,023)
  - Ephemeral port number
    - A port No. chosen randomly by the transport layer SW running on the client host.

Process-to-process Delivery(cont'd)

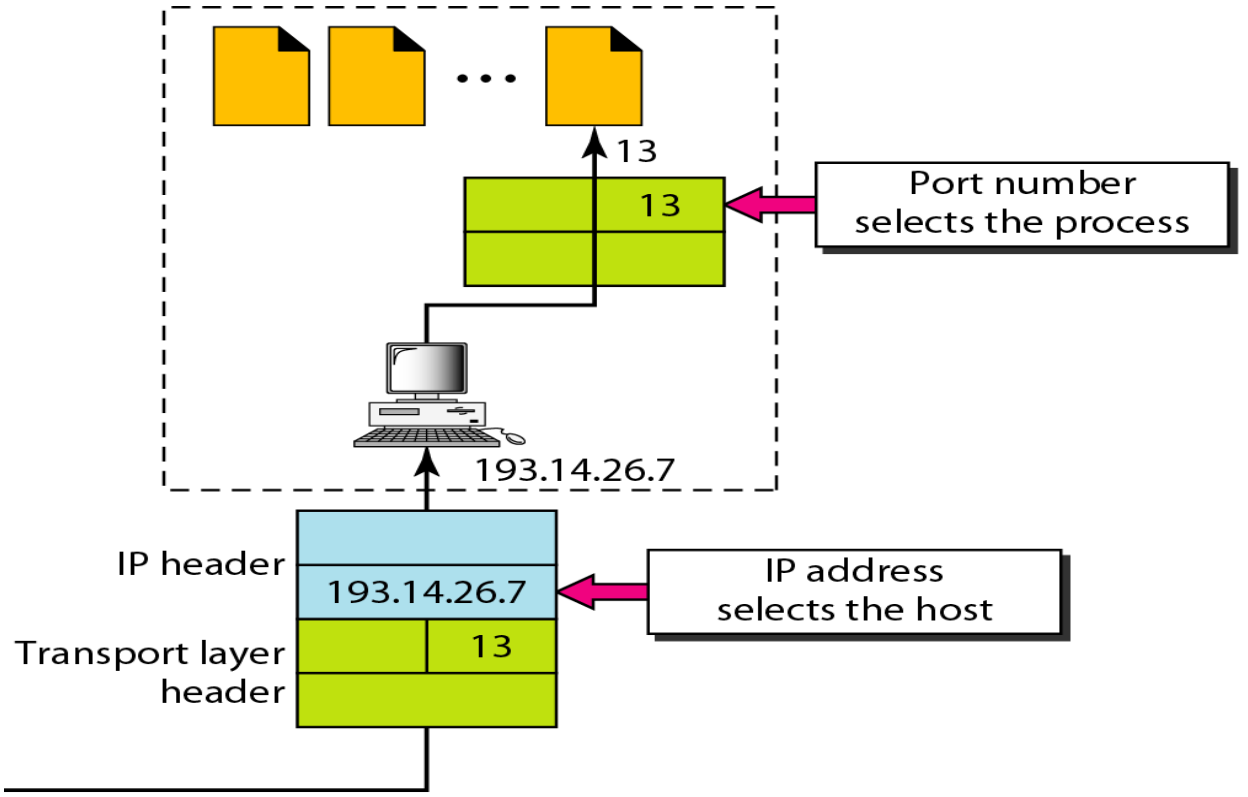
Addressing

Port number



Process-to-process Delivery(cont'd)

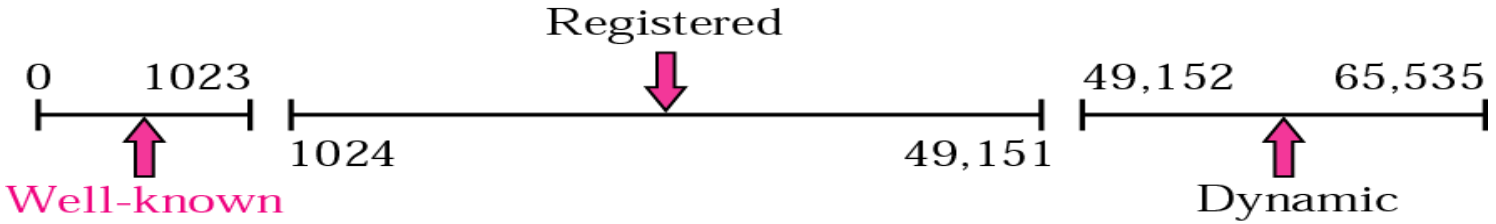
- Addressing
  - IP addresses Versus Port numbers



Process-to-process Delivery(cont'd)

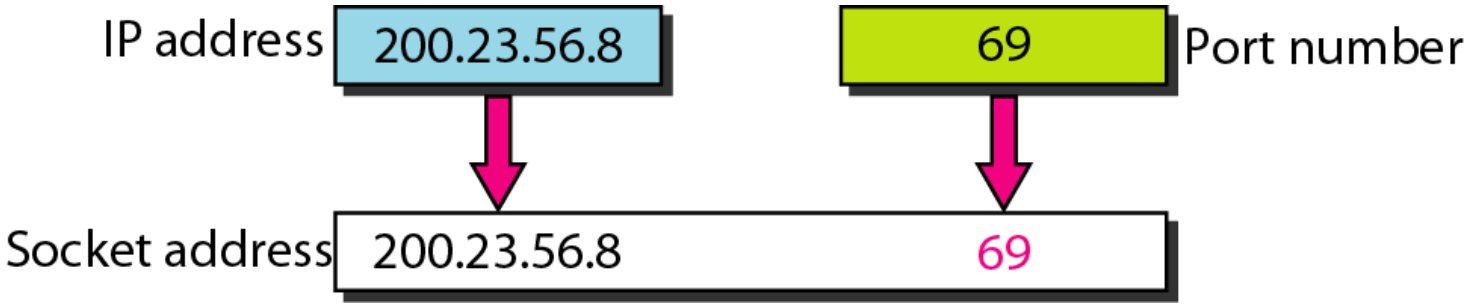
IANA Ranges

- IANA (Internet Assigned Number Authority) had divided the port numbers into three ranges :
- Well-known ports
  - Ranging : 0 ~ 1,023
  - Controlled by IANA.
- Registered Port
  - Ranging : 1,024 ~ 49,151
  - Are not assigned or controlled by IANA. Can be registered with IANA to avoid duplication.
- Dynamic Port
  - Ranging : 49,152 ~ 65,535
  - Are neither controlled nor registered.
  - These are the ephemeral port.



Process-to-process Delivery(cont'd)

- **Socket Addresses**
  - **The combination of an IP address and a port number.**
  - **A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.**

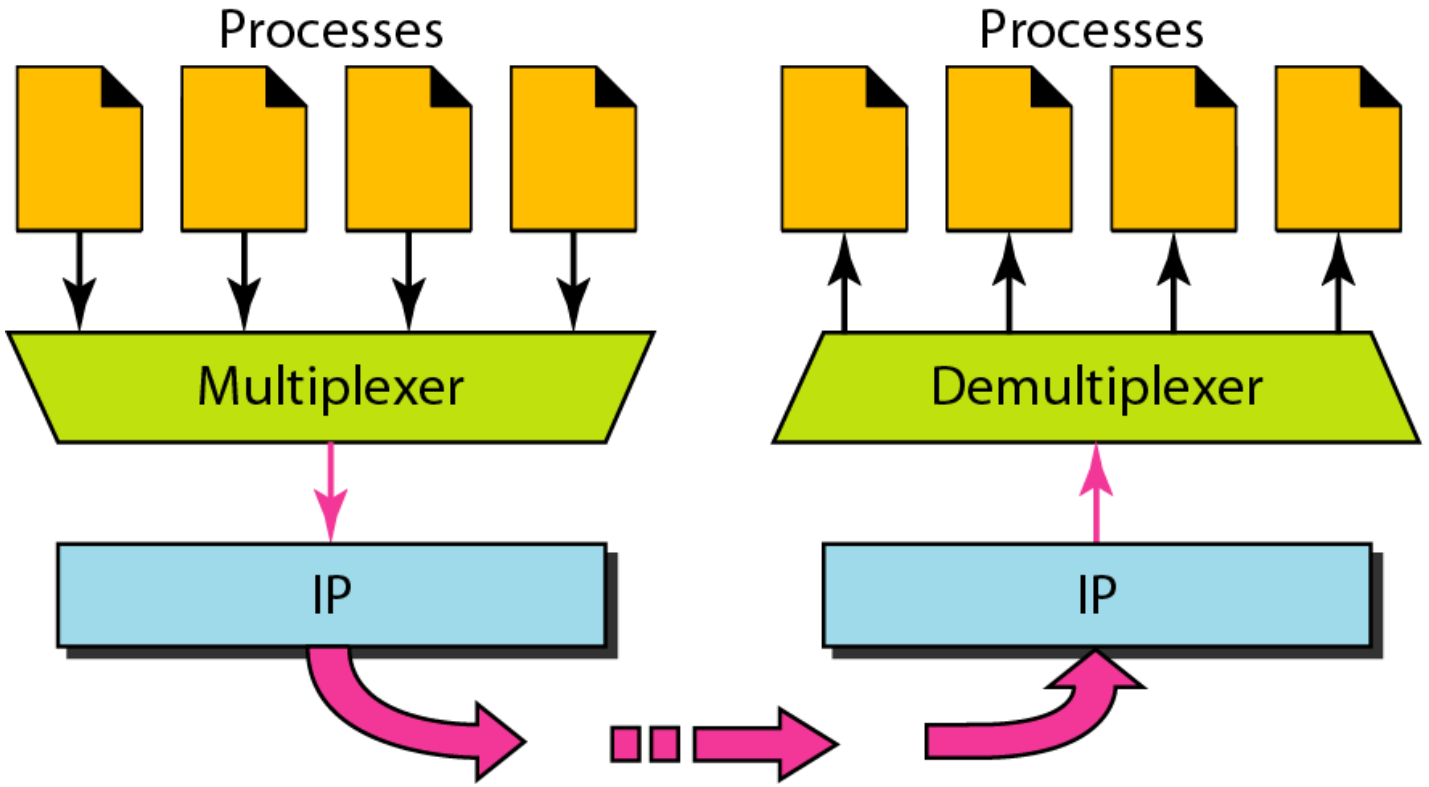


Process-to-process Delivery(cont'd)

- Multilexing and Demultiplexing
  - Multiplexing
    - At the sender site, there may be several processes to send packets, and there is only one transport layer protocol (UDP or TCP) at any time.
    - This is a many-to-one relationship and requires multiplexing.
    - After adding the header, the transport layer passes the packet to the network layer.
  - Demultiplexing
    - At the receiver site, the relationship is one-to-many and requires demultiplexing.
    - The transport layer receives datagrams from the network layer.
      - After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.

Process-to-process Delivery(cont'd)

- Multiplexing and Demultilexing



Connectionless Vs Connection-Oriented Service

- **Connectionless Service**
  - **The packets are not numbered; they may be delayed or connection release.**
  - **There is no acknowledgment either.**
  - **UDP protocol is connectionless in the Internet model.**
- **Connection-Oriented Service**
  - **A connection is first established between the sender and the receiver.**
  - **Data are transferred.**
  - **At the end, the connection is released.**
  - **TCP and SCTP are connection-Oriented protocols.**

Reliable Versus Unreliable

- **Reliable Service**
  - If the application layer program needs reliability, we use a reliable transport protocol such as TCP and SCTP.
  - This means a slower and more complex service.
- **Unreliable Service**
  - If the application layer program does not need reliability because it uses its own flow and error control mechanism or it needs fast service or the nature of the service does not demand flow and error control (real-time application), then unreliable protocol such as UDP can be used.
- **Do we need reliability control at the transport layer, even the data link layer is reliable and has flow and error control ?**
  - The answer is yes.
  - The network layer in the Internet is unreliable (best-effort delivery), we need to implement reliability at the transport layer.

Error control

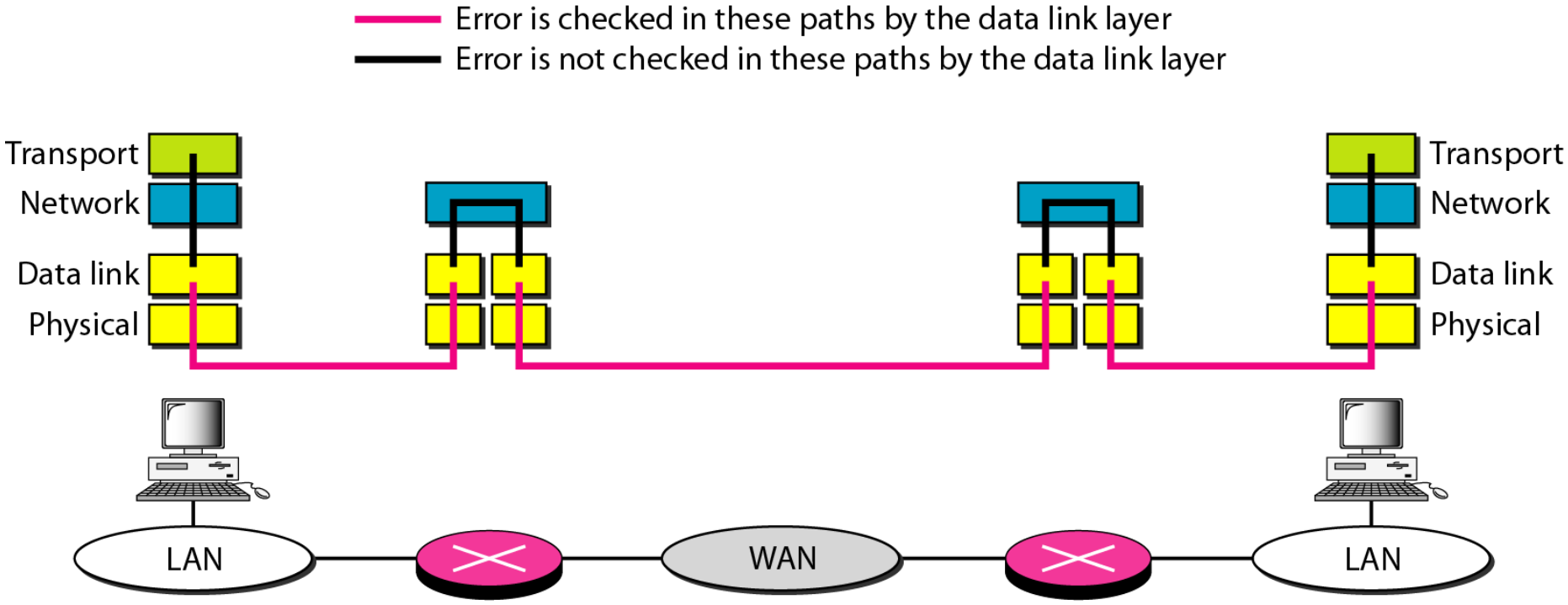
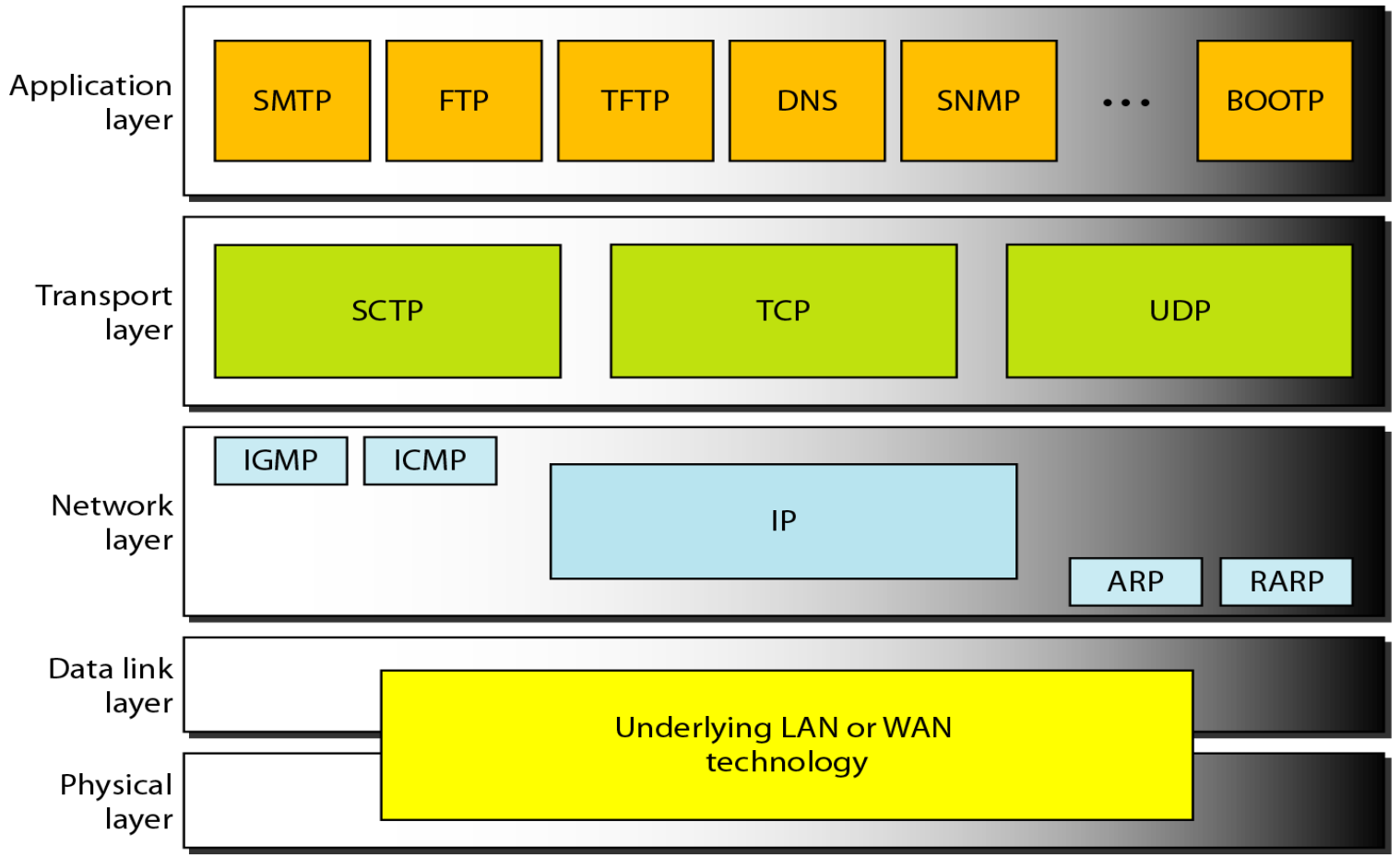


Figure 23.7 Error control

Three Protocols

Position of UDP,TCP, and SCTP in TCP/IP suite



23-2 USER DATAGRAM PROTOCOL (UDP)

The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.

Topics discussed in this section:

Well-Known Ports for UDP
User Datagram
Checksum
UDP Operation
Use of UDP

UDP (cont'd)

- **If UDP is so powerless, why would a process want to use it ?**
  - **very simple protocol using a minimum of overhead**
    - **if a process wants to send a small message and does not care much about reliability, it can use UDP**
    - **if it sends a small message, taking much less interaction between the sender and receiver than it does using TCP**

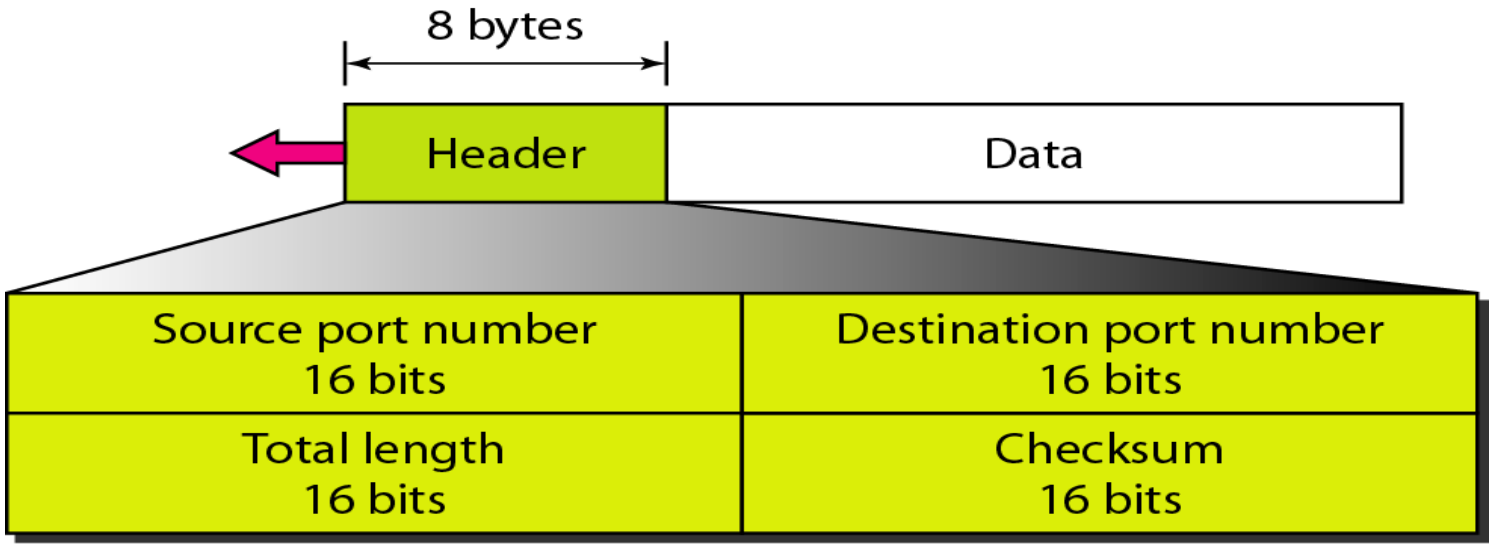
Table 23.1 Well-known ports used with UDP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

User Datagram Protocol (UDP)

User Datagram

- **Including fixed 8 byte header**



- **The calculation of checksum and its inclusion in the user datagram are optional.**

UDP length = IP length – IP header's length

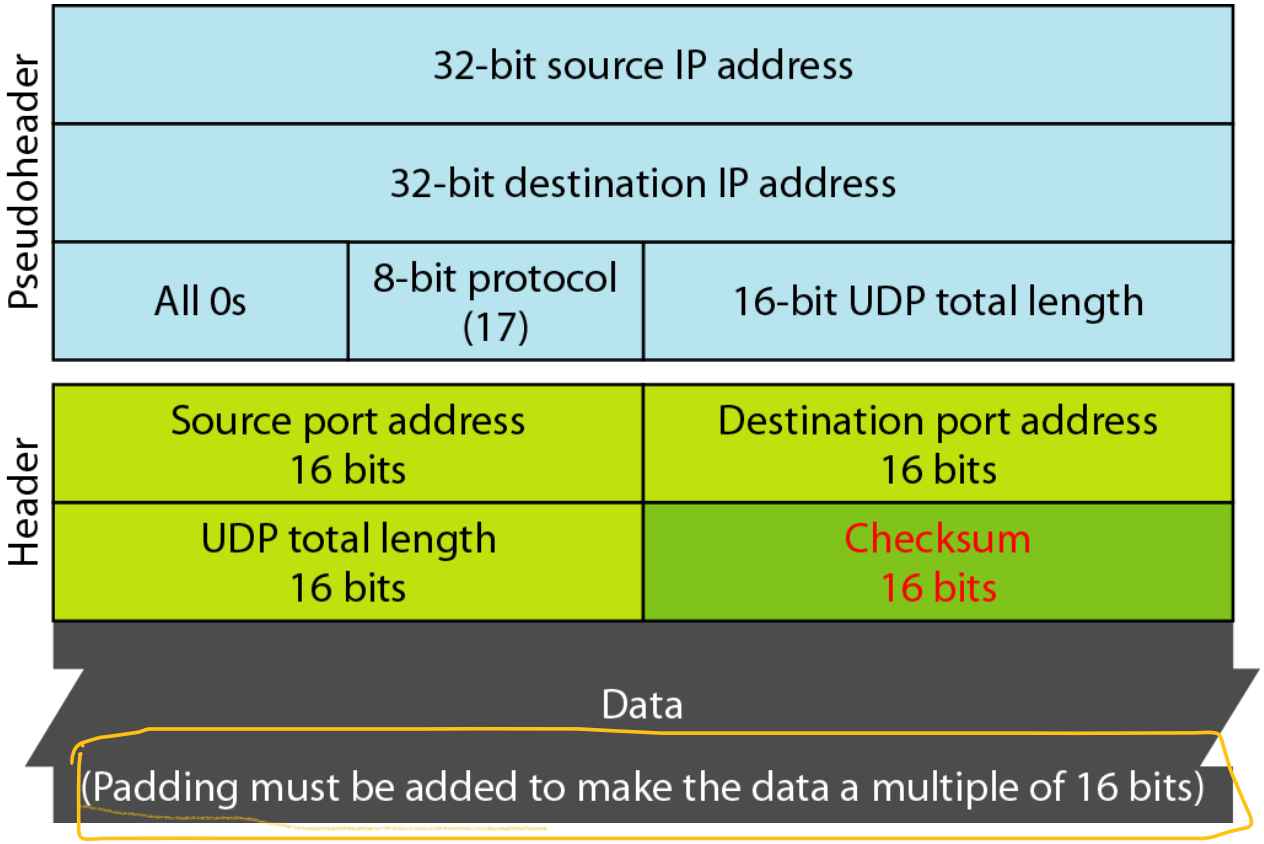
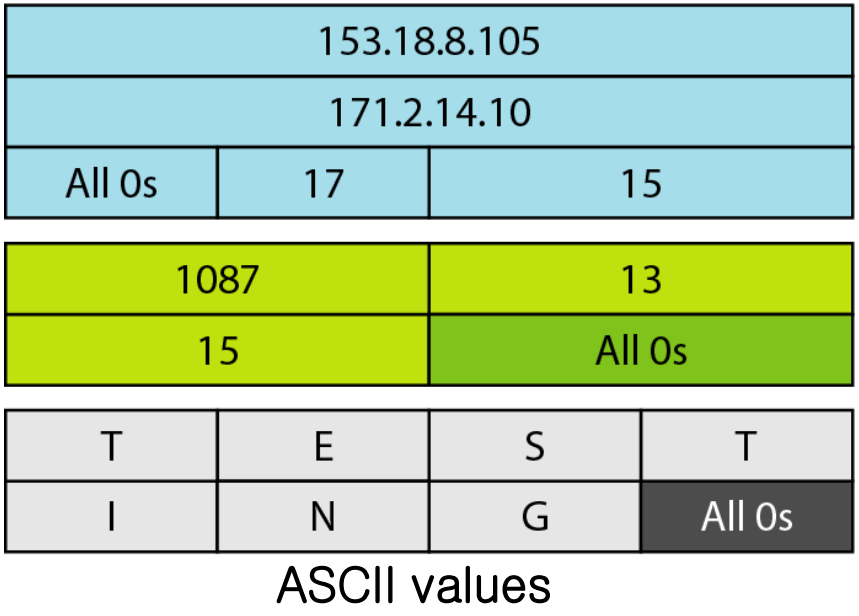
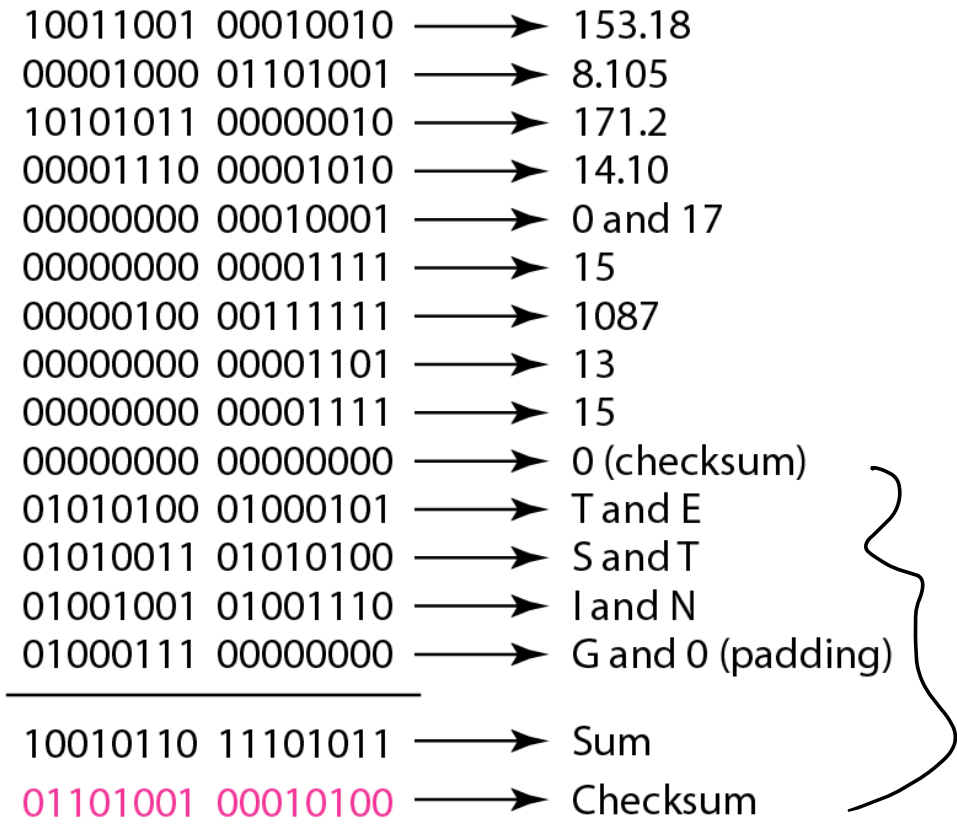


Figure 23.10 Pseudoheader for checksum calculation

Figure 23.11 Checksum calculation of a simple UDP user datagram





UDP Operation

- **Connectionless Services**
  - UDP provides a connectionless services that each datagram sent by UDP is an independent datagram.
- **Flow and Error Control**
  - There is no flow control and hence no window mechanism.
  - There is no error control mechanism in UDP except for the checksum.
- **Encapsulation and Decapsulation**
  - The UDP protocol encapsulates and decapsulates messages in an IP datagram.

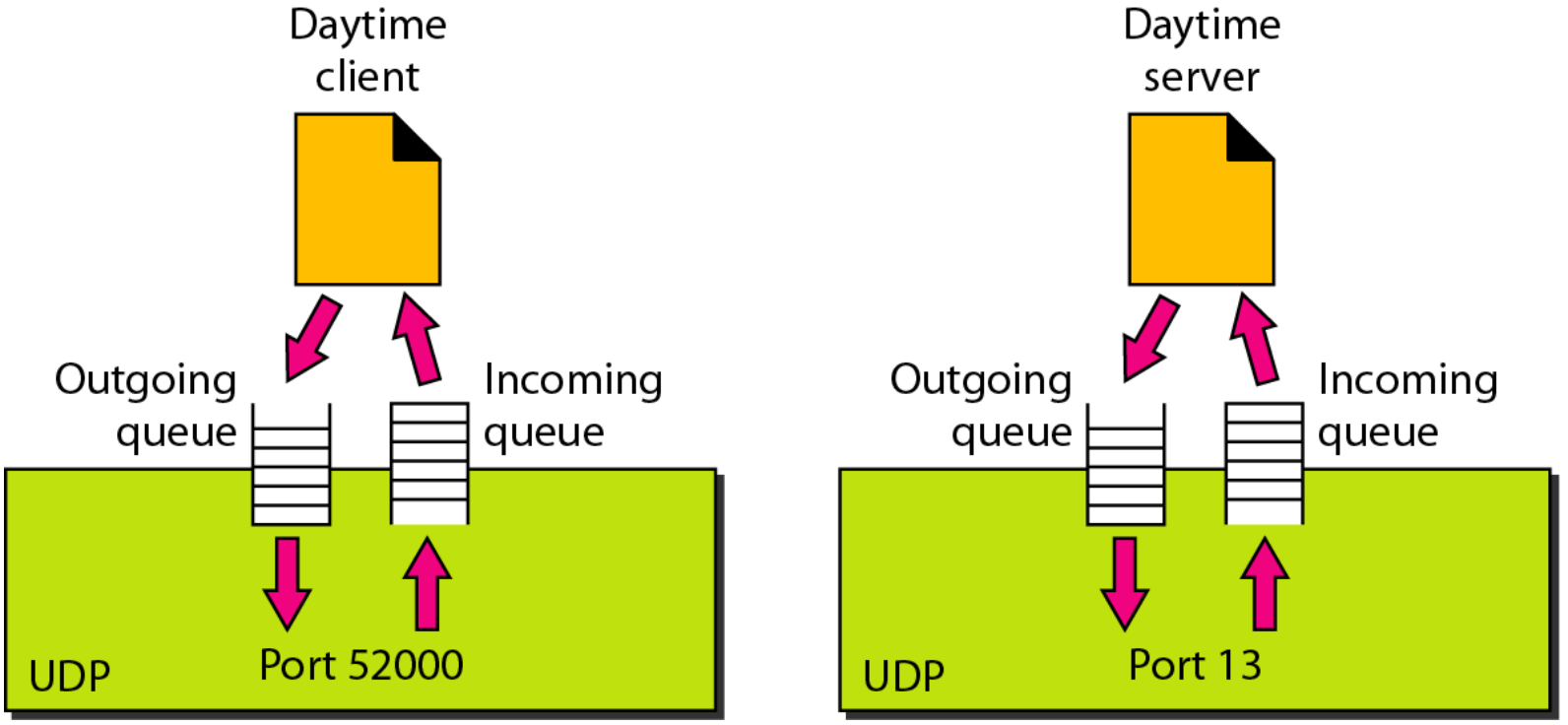
UDP Operation (cont'd)

- Queuing
  - At client site
    - When a process starts, it requests a port No. from the operating system.
    - The client process sends messages to the outgoing queue by using the source port No. specified in the request.
    - UDP removes the messages one by one and, after adding the UDP header, delivers them to IP.
    - An outgoing queue can overflow. If this happens, the OS can ask the client process to wait before sending any more messages.
    - When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port No. specified in the destination port No. field of the user datagram. If there is a queue, UDP sends the received user datagram to the end of the queue.

UDP Operation (cont'd)

- Queuing
  - At Server site
    - When it starts running, a server asks for incoming and outgoing queues, using its well-known port.
    - When a message arrives for a server,
    - UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.
    - If there is a queue, UDP sends the received user datagram to the end of the queue.
    - When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port No. specified in the request.
    - UDP removes the messages one by one and, after adding the UDP header, delivers them to IP.

Figure 23.12 Queues in UDP



Applications of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
  - Not used for sending bulk data, such as FTP
  - TFTP including internal flow and error control
  - Suitable transport protocol for multicasting
  - Used for some route updating protocols such as Routing Information Protocol (RIP)
  - Used for management processes such as SNMP.

23-3 TCP

TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

Topics discussed in this section:

TCP Services
TCP Features
Segment
A TCP Connection
Flow Control
Error Control

TCP Services

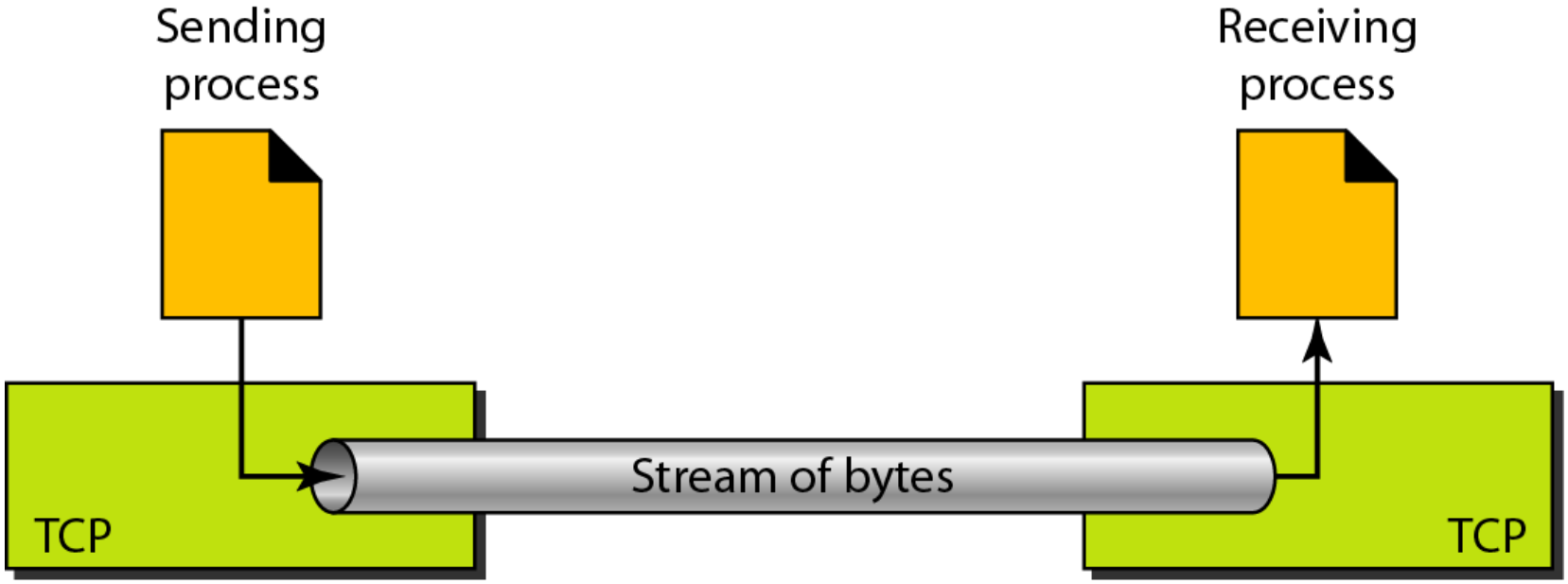
Process-to-process Communication

- Like UDP, TCP provides process-to-process communication using port numbers.
  - well-known ports used by TCP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

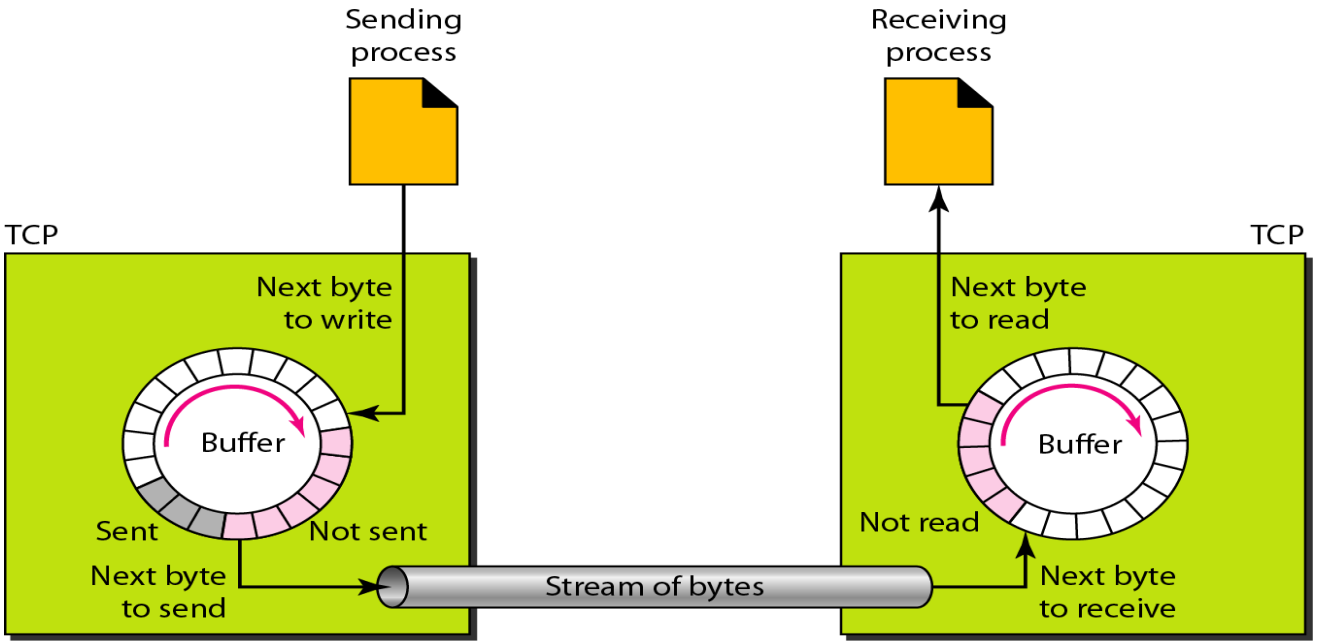
TCP Services

- **Stream Delivery Service**
  - **TCP is a stream-oriented protocol**
  - **TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their data across the Internet.**



TCP Services (cont'd)

- Sending and Receiving Buffers
  - Because the sending and receiving processes may not produce and consume data at the same speed, TCP needs buffers for storage.
  - One way to implement is to use a circular array

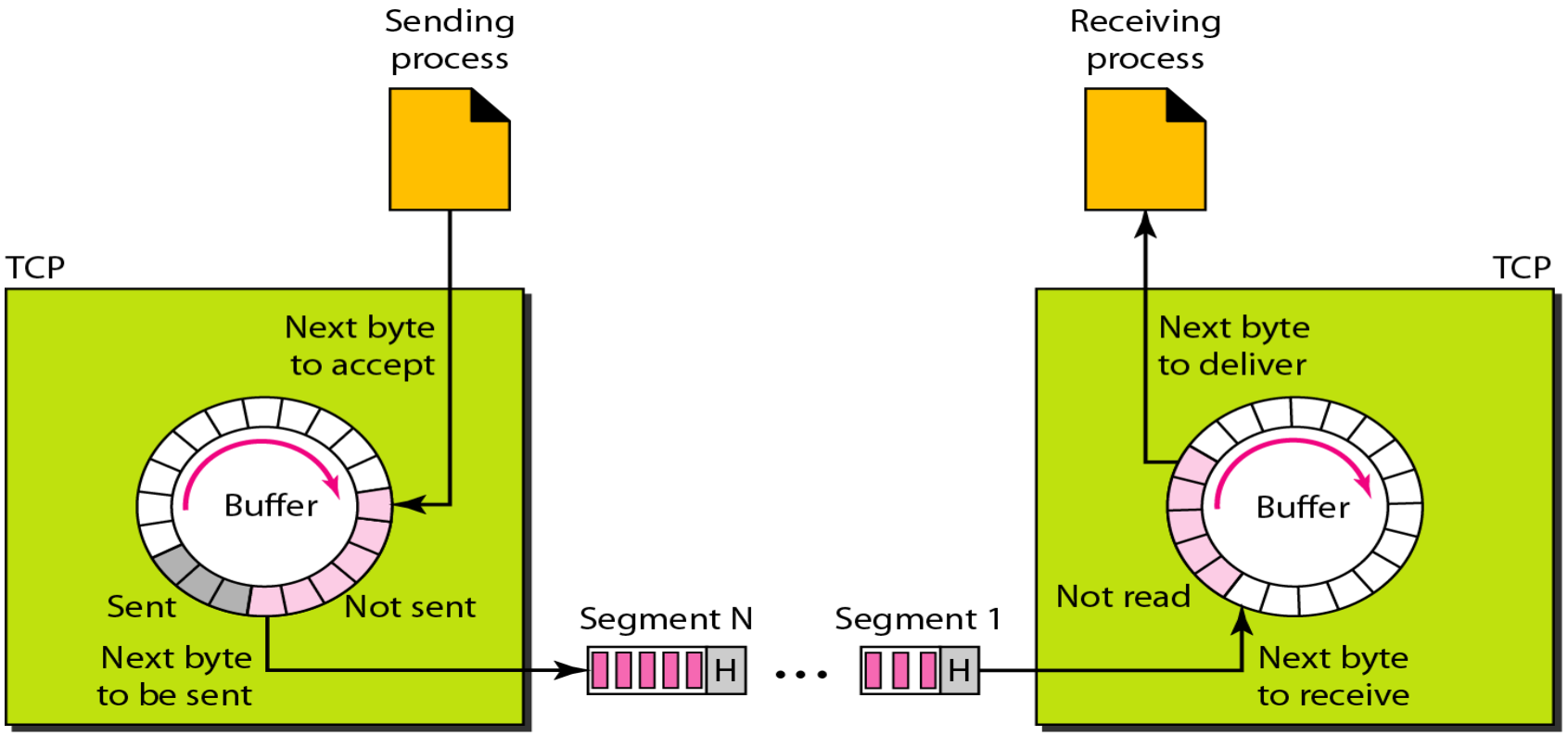


TCP Services (cont'd)

- **Segments**
  - **At the transport layer, TCP groups a number of bytes together into a packet called a segment.**
  - **TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission.**
  - **The segments are encapsulated in IP datagrams and transmitted.**
  - **This entire operation is transparent to the receiving process.**
  - **Segments received out of order, lost, or corrupted may be resent.**

TCP Services (cont'd)

- **TCP Segments**



TCP Services (cont'd)

- **Full-Duplex Service**
  - **TCP offers full-duplex service**
    - After two application programs are connected to each other, they can both send and receive data.
  - **Piggybacking**
    - When a packet is going from A to B, it can also carry an acknowledgment of the packets received from B

TCP Services (cont'd)

- **Connection-Oriented Services**
  1. A's TCP informs B's TCP and gets approval from B's TCP
  2. A's TCP and B's TCP exchange data in both directions
  3. After both processes have no data left to send and the buffers are empty, two TCPs destroy their buffers
- **Reliable Service**
  - TCP uses the acknowledgment mechanism to check the safe and sound arrival of data

TCP Features

Byte numbers

- There is no field for a segment number value. Instead, there are two fields called the sequence No. and the acknowledgment No. These two fields refer to the byte No.
- All data bytes being transferred in each connection are numbered by TCP.
- The numbering starts with a randomly generated number.
- Number range for first byte : 0 ~ 2^{32} -1
  - If random number is 1,057 and total number 6,000bytes, the bytes are numbered from 1,057 to 7,056
- Byte numbering is used for flow and error control.

TCP Features (cont'd)

Note

The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.

TCP Features (cont'd)

- **Sequence number**
  - **After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.**
  - **Segment number for each segment is number of the first byte carried in that segment.**

TCP Features (cont'd)

Note

The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.

TCP Features (cont'd)

- Acknowledgment Number

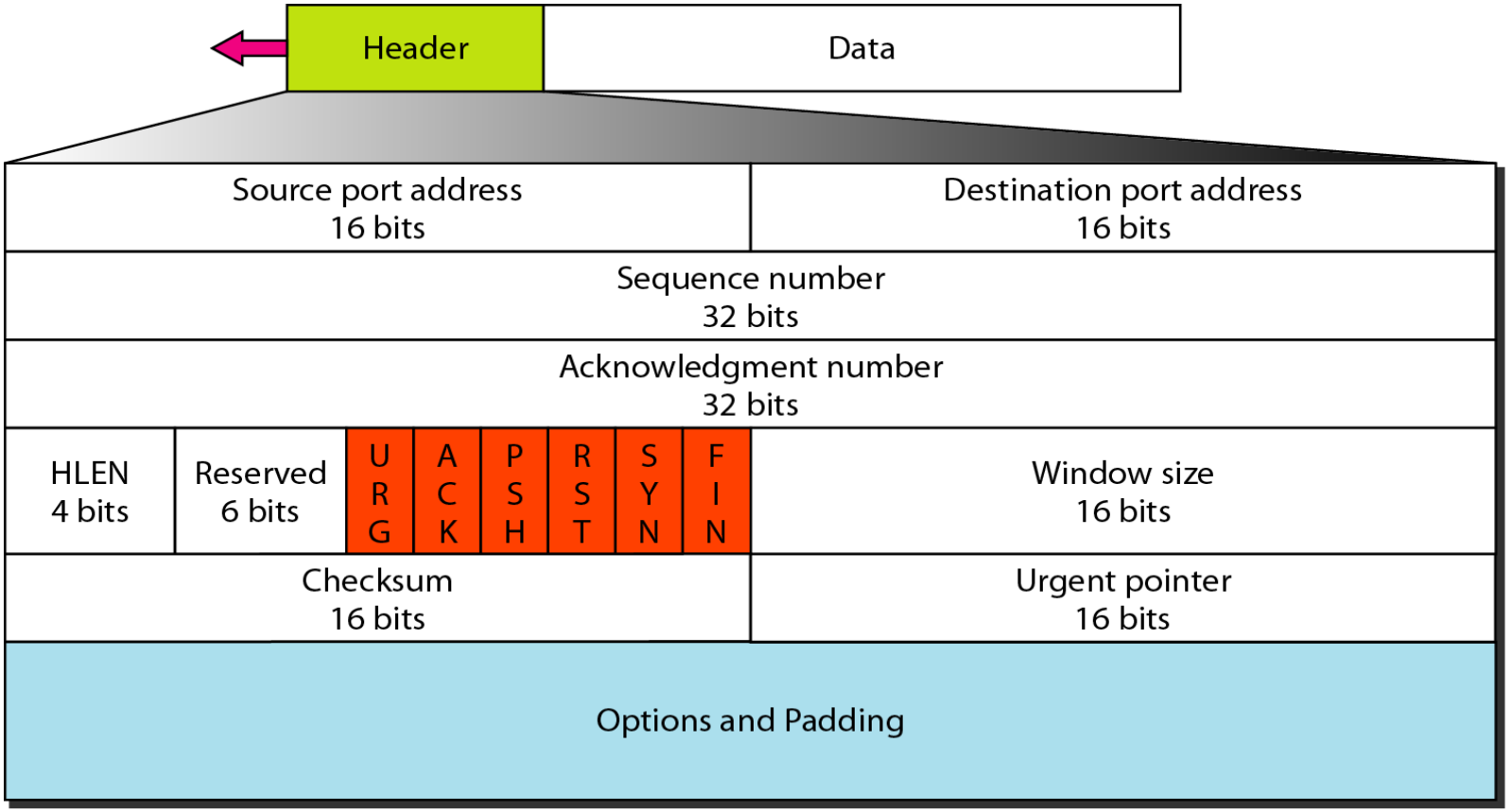
The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive. The acknowledgment number is cumulative.

TCP Features (cont'd)

- TCP provides flow control, error control, and congestion control.

TCP Segment format

- A packet in TCP is called segment



Segment (cont'd)

- Source port address
  - defining the port number of application program in the host that is sending the segment
- Destination port address
  - defining the port number of application program in the host that is receiving the segment
- Sequence number
  - defining the number assigned to the first byte of data contained in this segment
  - during the connection establishment, each party uses a random number generator to create an *initial sequence number (ISN)*

Segment (cont'd)

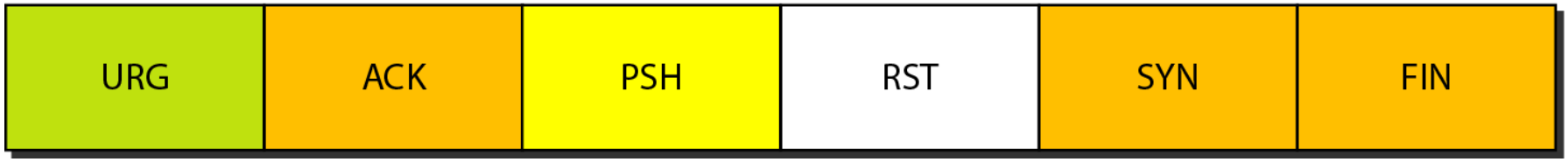
- **Acknowledgment number**
  - The byte number that the sender of the segment is expecting to receive from the other party.
  - If the source of the segment has successfully received byte number x from the other party, it defines x+1 as the acknowledgment number
- **Header length**
  - Indicating the number of 4-byte words in the TCP header
    - the value between 5 and 15 (20 and 60 bytes)
- **Reserved**
  - For future use

Segment (cont'd)

- **Control**
  - **Enabling flow control, connection establishment and termination, and mode of data transfer in TCP**

URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push

RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



Segment : Description of flags in the control field

Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

Segment (cont'd)

- Window size
  - defining the size of the window, in bytes, that the other party must maintain.
  - maximum size of window : 65,535 bytes
- Checksum : same as UDP
- Urgent pointer
  - used when the segment contains urgent data
  - defining the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment
- Options : 40 bytes

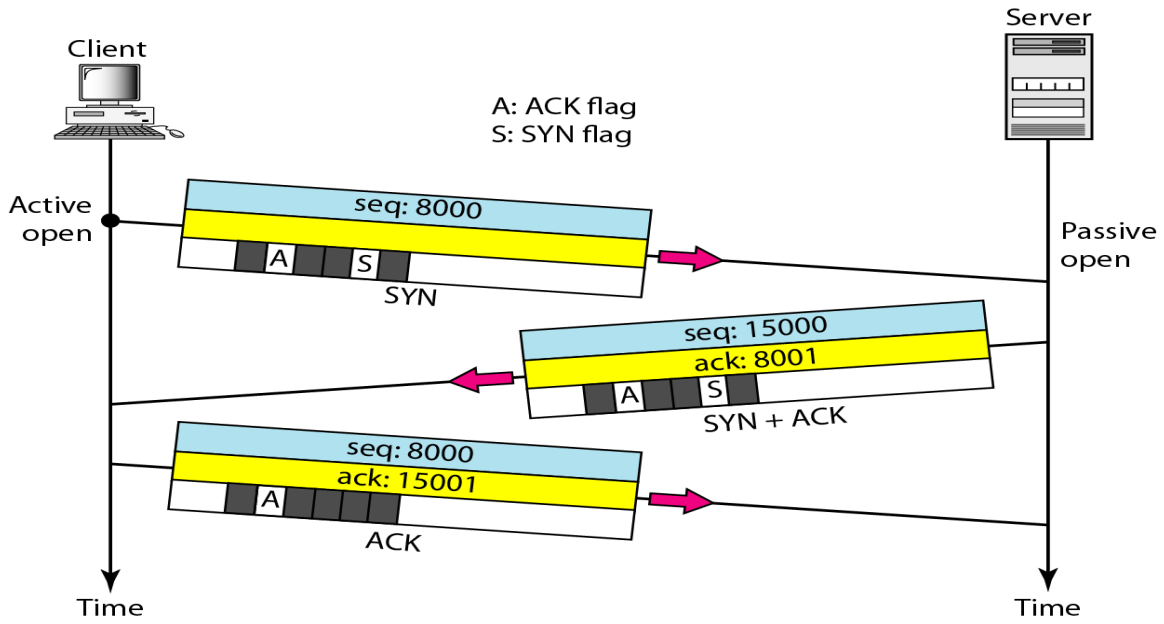
TCP Connection

- **TCP is connection-oriented**
  - A connection-oriented transport protocol establishes a virtual path between the source and destination.
  - All the segments belonging to a message are then sent over this virtual path.
  - Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
  - You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented.
  - The point is that a TCP connection is virtual, not physical.
  - TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself.

TCP Connection

- **Connection Establishment**
  - **TCP transmits data in full-duplex mode.**
  - **When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.**
  - **This implies that each party must initialize communication and get approval from the other party before any data are transferred.**

Figure 23.18 Connection establishment using three-way handshaking

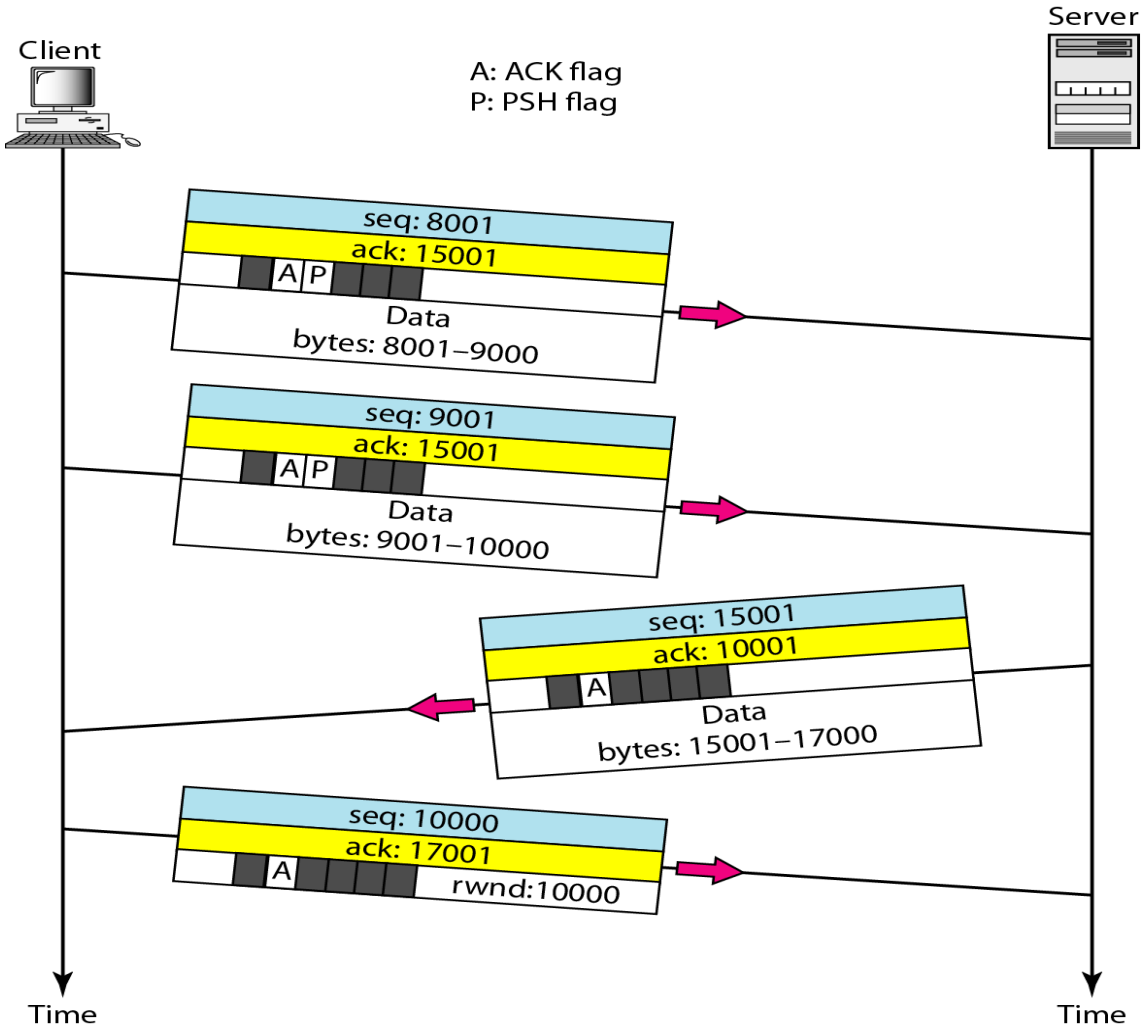


- The client sends the 1st segment, a SYN segment, in which only the SYN flag is set.
- The server sends the 2nd segment, a SYN+ACK segment, with 2 flag bits set: SYN and ACK.
- The client sends the 3rd segment. This is just an ACK segment.

Connection Establishment

- **Simultaneous Open**
  - Simultaneous Open may occur when both processes issue an active open.
  - In this case, both TCPs transmit a SYN+ACK segment to each other, and one single-connection is established between them.
- **SYN Flooding Attack**
  - SYN Flooding Attack happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming from a different client by faking the source IP addresses in the data-grams.
  - The SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

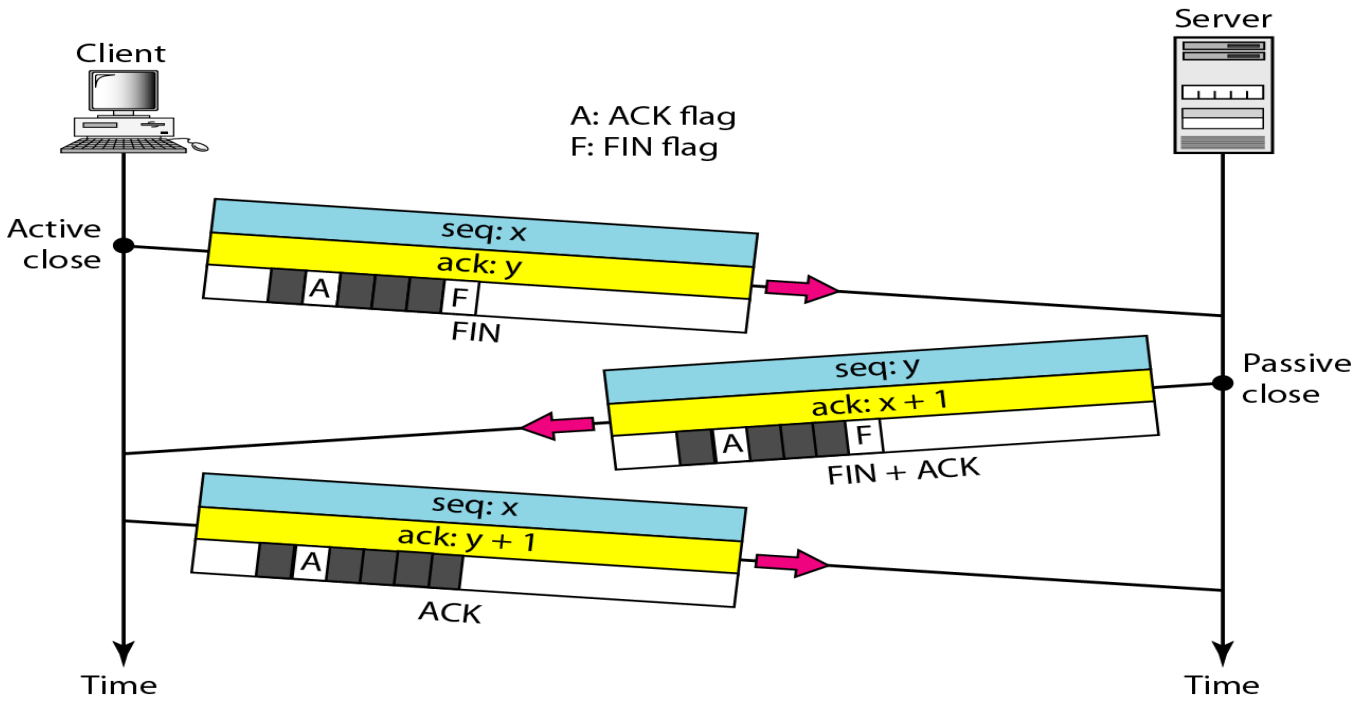
Figure 23.19 Data transfer



Data Transfer

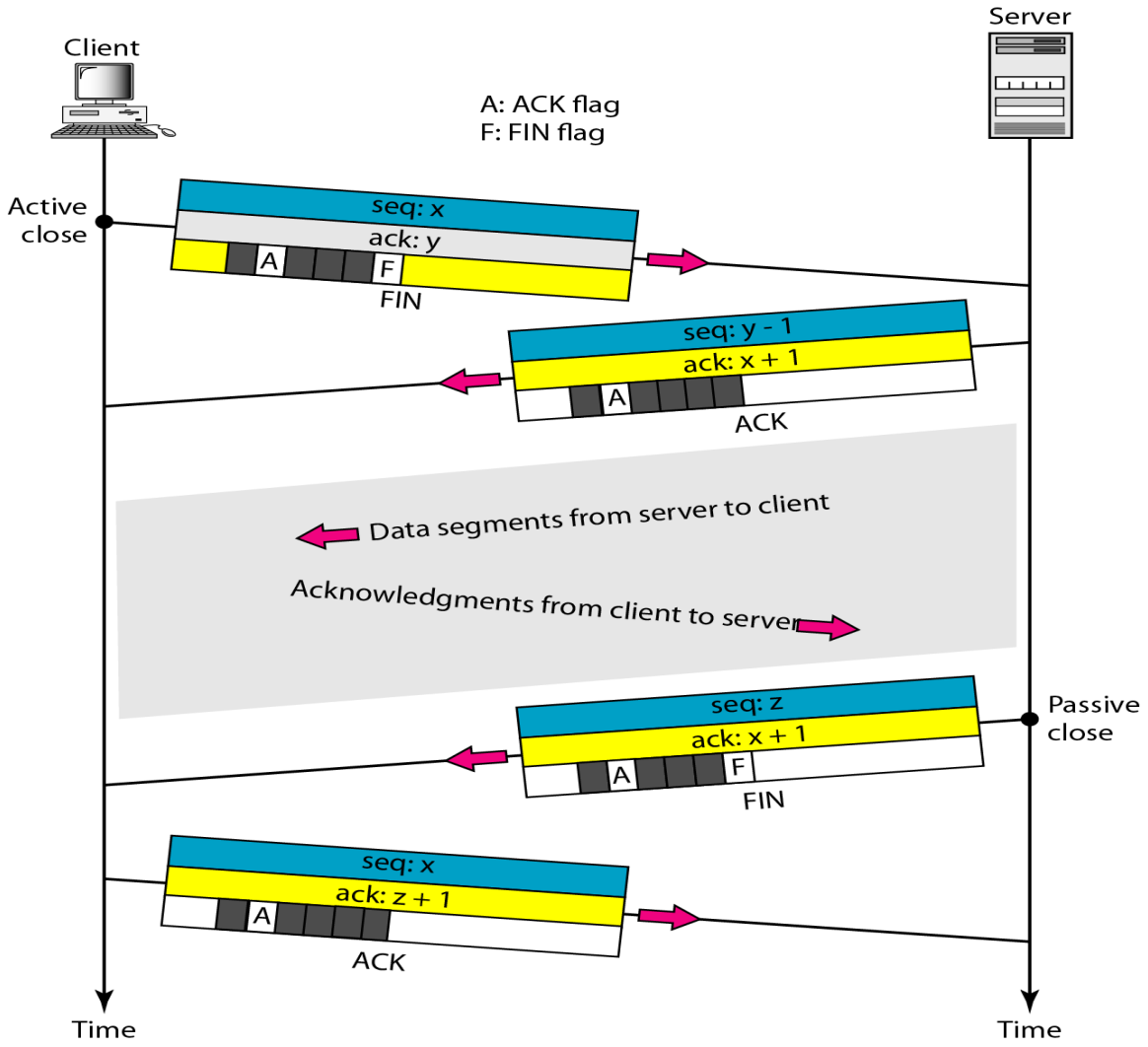
- **Pushing Data**
  - The application program at the sending site can request a push operation that the sending TCP must not wait for the window to be filled.
  - It must create a segment and send it immediately.
- **Urgent Data**
  - When the sending application program wants a piece of data to be read out of order by the receiving application program.
  - Sender can send a segment with the URG bit set.
  - When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment, using the value of the pointer, and delivers them, out of order, to the receiving application program.

Figure 23.20 Connection termination using three-way handshaking



- The client TCP, after receiving a close command from the client process, sends the 1st segment, a FIN segment in which the FIN flag is set.
- The server TCP, after receiving the FIN segment, inform its process of the situation and sends the 2nd segment, a FIN+ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction.

Figure 23.21 Half-close



Summary

- In the client/sever paradigm, an application program on the local host, called the client, needs services from an application program on the remote host, called a server.
- Each application program has a port number that distinguishes it from other programs running at the same time on the same machine.
- The client program is assigned a random port number called an ephemeral port number; the server program is assigned a universal port number called a well-known port number.
- The combination of the IP address and the port number, called the socket address, defines a process and a host.
- UDP is a connectionless, unreliable transport protocol with no embedded flow or error control mechanism except the checksum for error detection.
- The UDP packet is called user datagram. A user datagram is encapsulated in the data field of an IP datagram.

Summary(2)

- Transmission Control Protocol (TCP) is one of the transport layer protocols in the TCP/IP protocol suite.
- TCP provides process-to-process, full-duplex, and connection-oriented service.
- The unit of data transfer between two devices using TCP software is called a segment; it has 20 to 60 bytes of header, followed by data from the application program.
- A TCP connection normally consists of three phases: connection establishment, data transfer, and connection termination.
- Connection establishment requires three-way handshaking; connection termination requires three- or four-way handshaking.
- TCP uses flow control, implemented as a sliding window mechanism, to avoid overwhelming a receiver with data.

Thanks !