Unit 4

The Transport Layer

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.

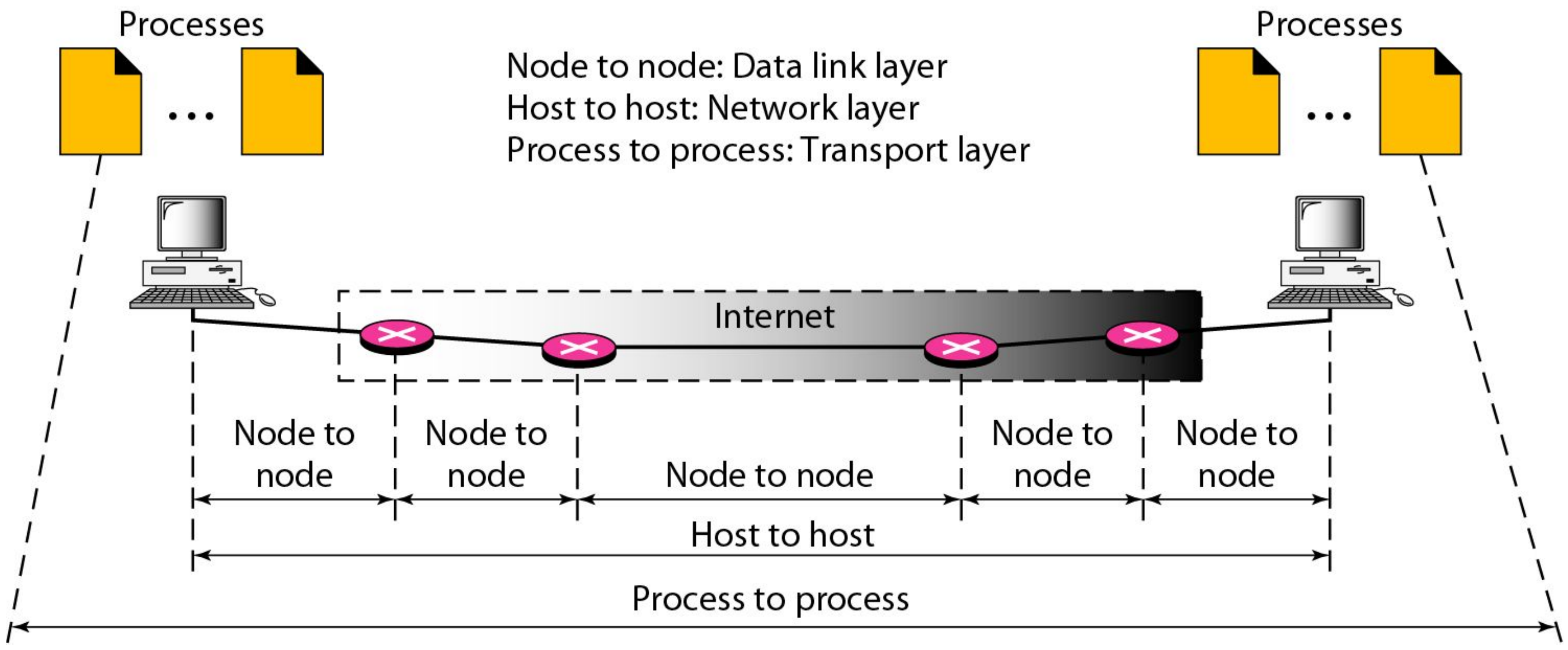
Topics discussed in this section:

Client/Server Paradigm
Multiplexing and Demultiplexing
Connectionless Versus Connection-Oriented Service
Reliable Versus Unreliable
Three Protocols

Note

The transport layer is responsible for process-to-process delivery.

Figure 23.1 *Types of data deliveries*



Client/Server Paradigm

- There are several ways to achieve process-to-process communication, the most common one is through the client/server paradigm
- A process on the local host, called a client, needs services from a process usually on the remote host, called a server

Figure 23.2 ***Port numbers : At the transport layer, we need a transport layer address, called a port number, to choose among multiple processes running on the destination host***

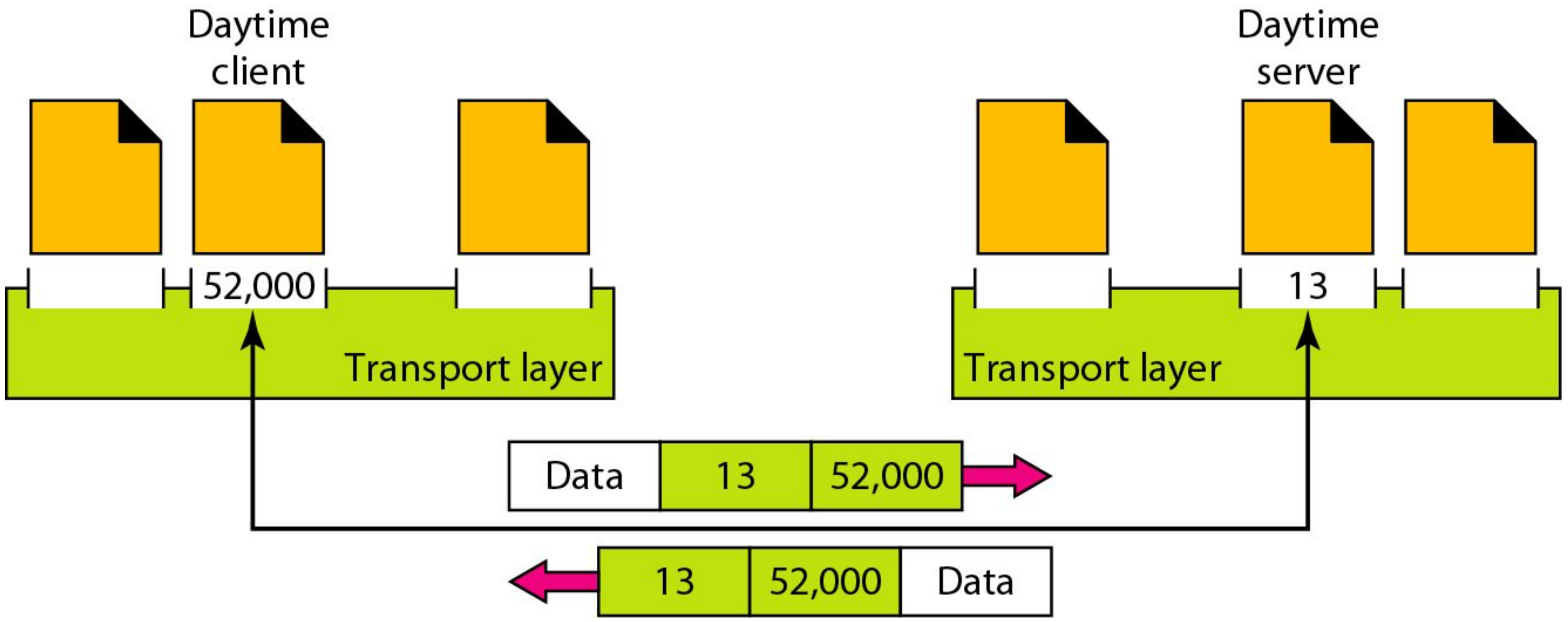


Figure 23.3 *IP addresses versus port numbers*

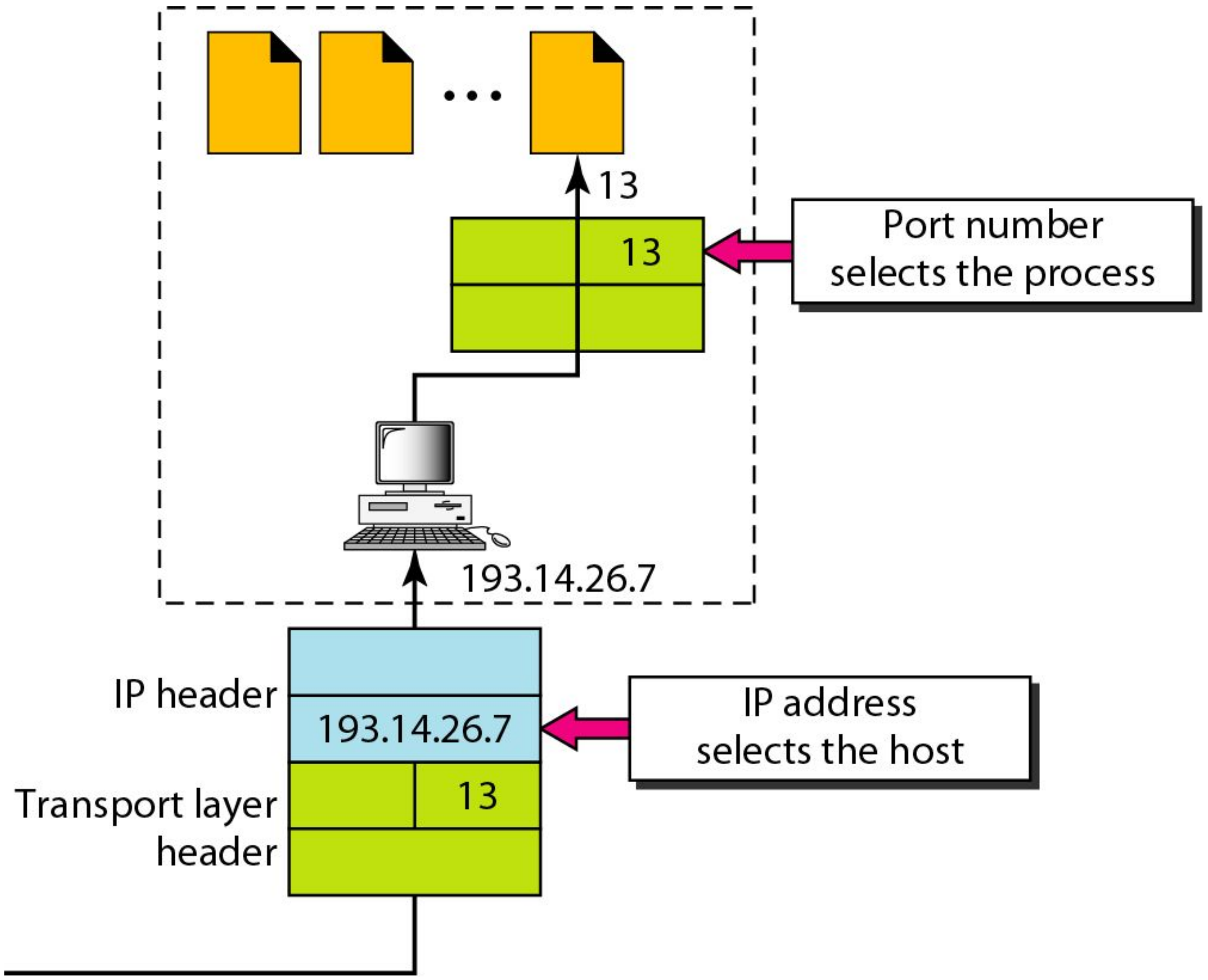


Figure 23.4 ***IANA ranges : The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (or private)***

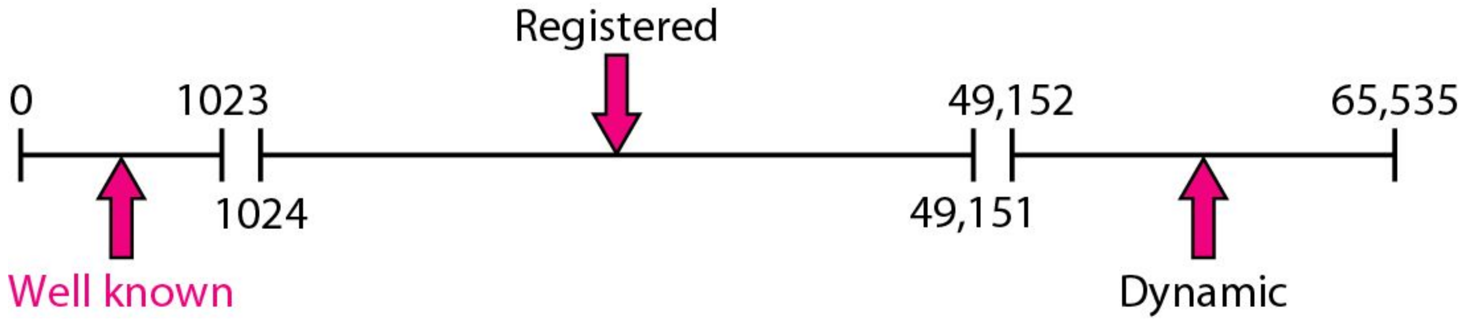


Figure 23.5 *Socket address*

- Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection

- The combination of an IP address and a port number is called a socket address.

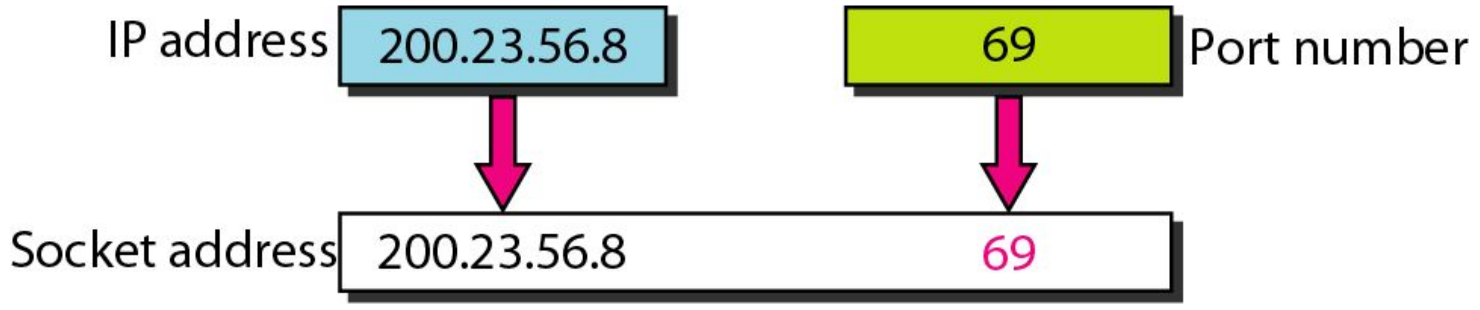


Figure 23.6 *Multiplexing and demultiplexing at transport layer*

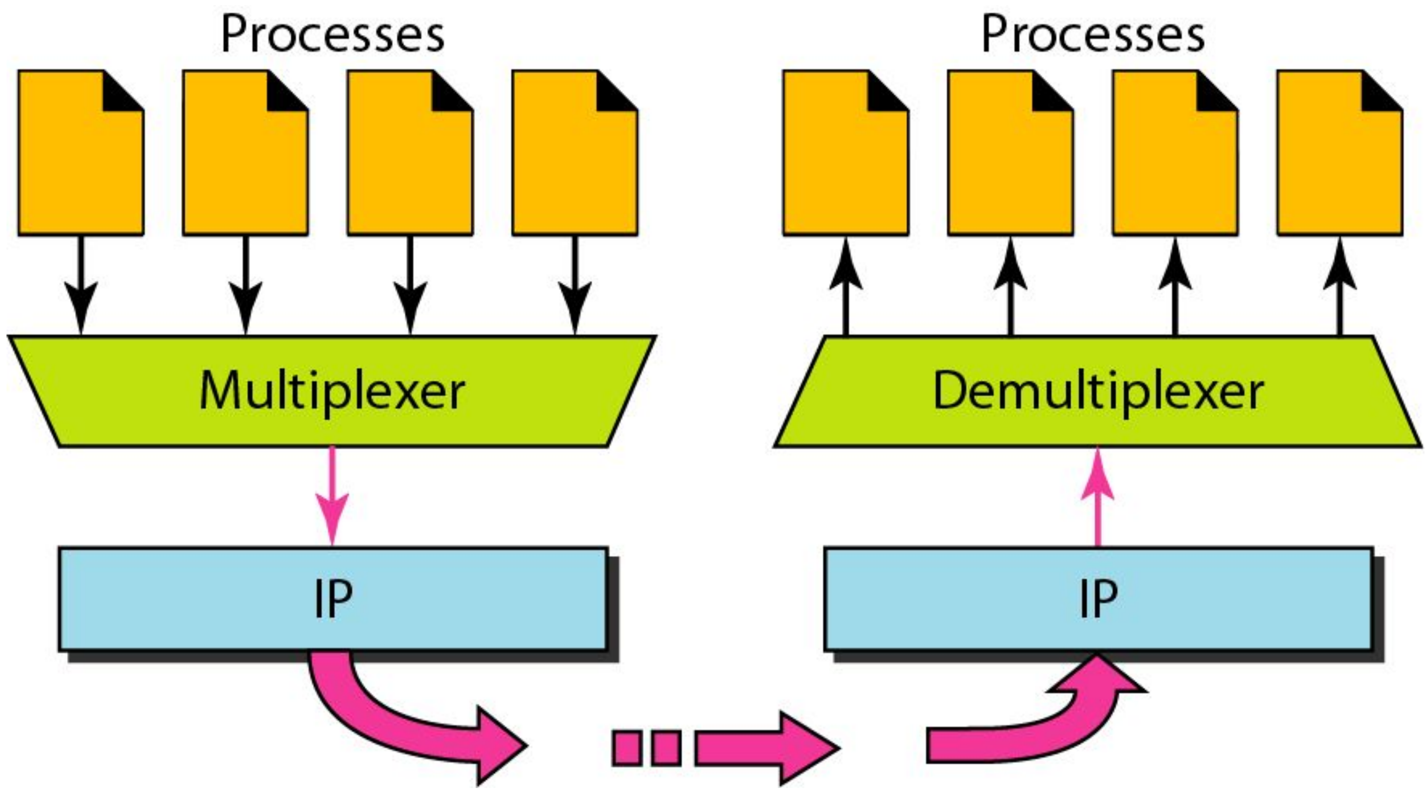


Figure 23.7 *Error control*

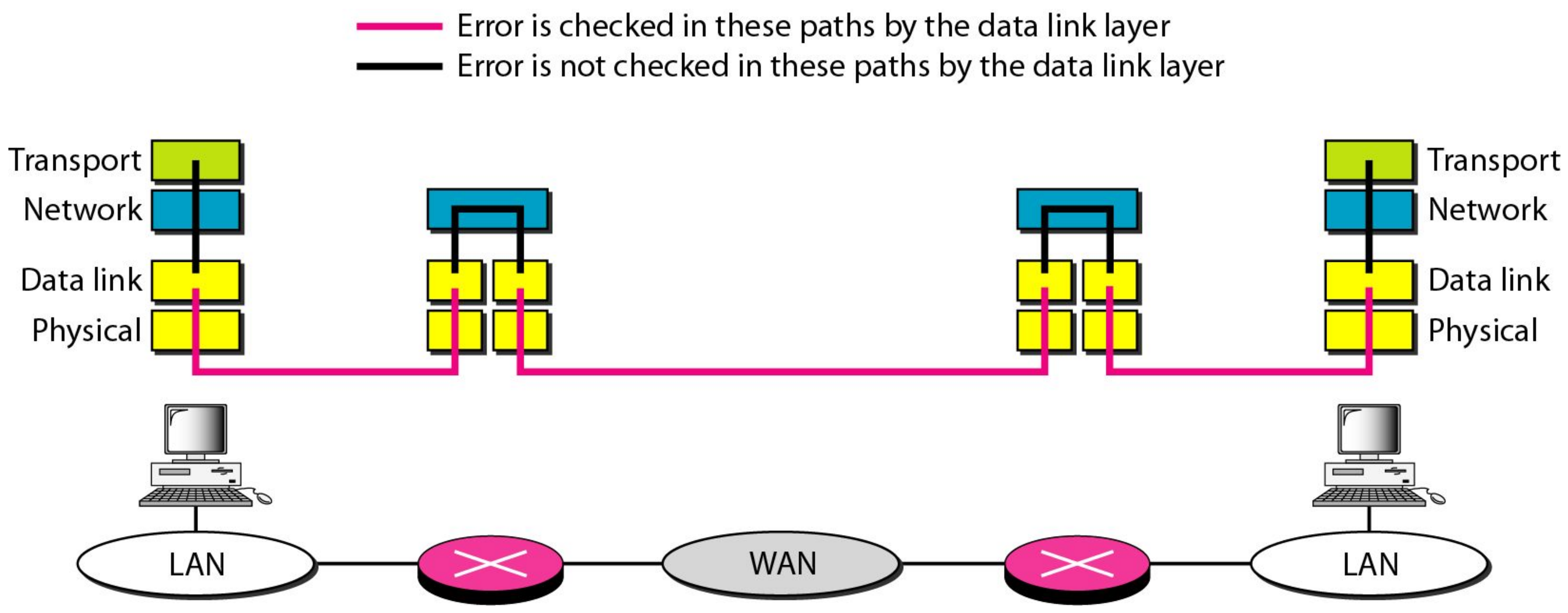
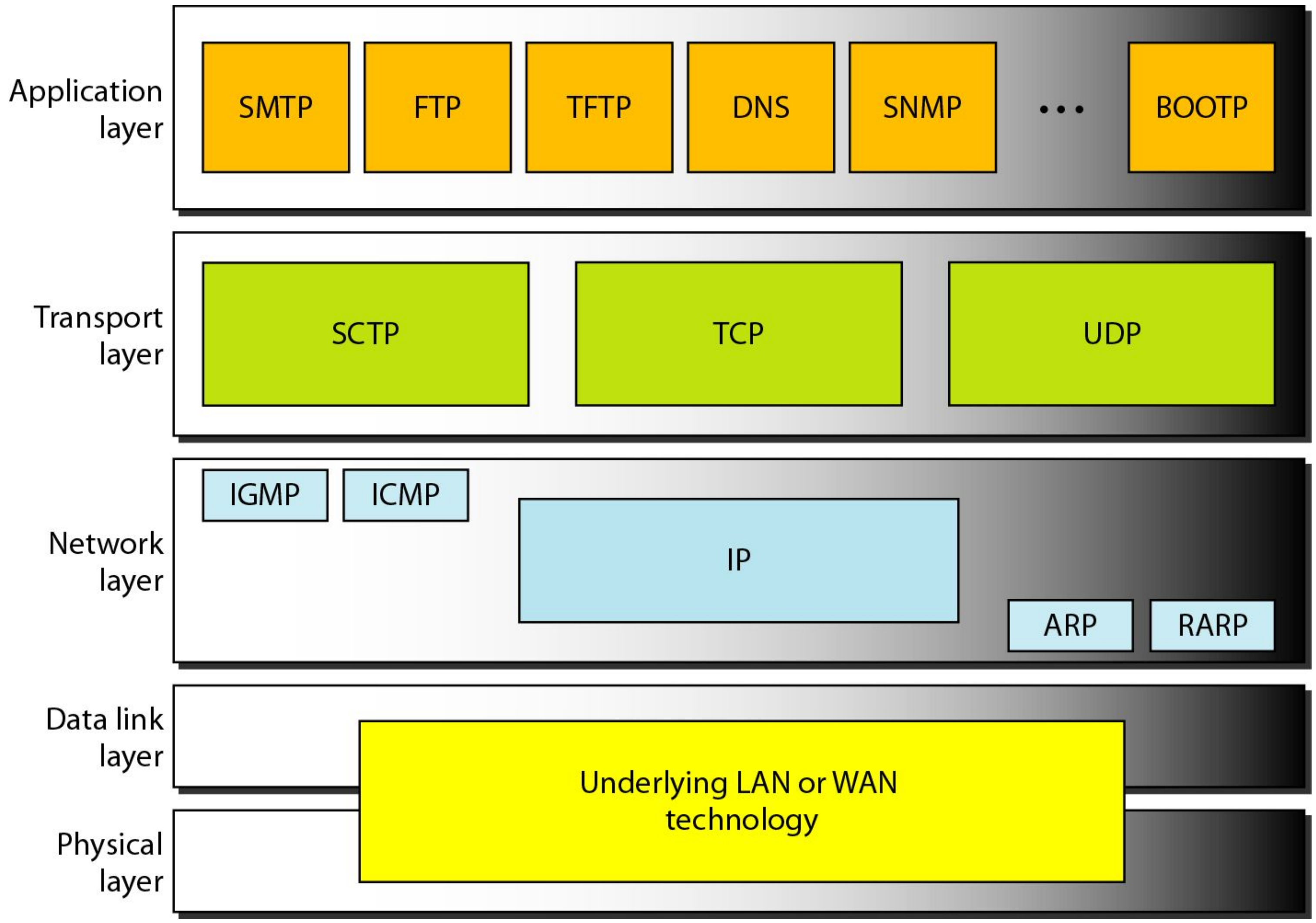

Error is checked in these paths by the data link layer
Error is not checked in these paths by the data link layer
Transport
Network
Data link
Physical
Transport
Network
Data link
Physical
LAN
WAN
LAN

Figure 23.8 Transport layer Protocol: *Position of UDP, TCP, and SCTP in TCP/IP suite*



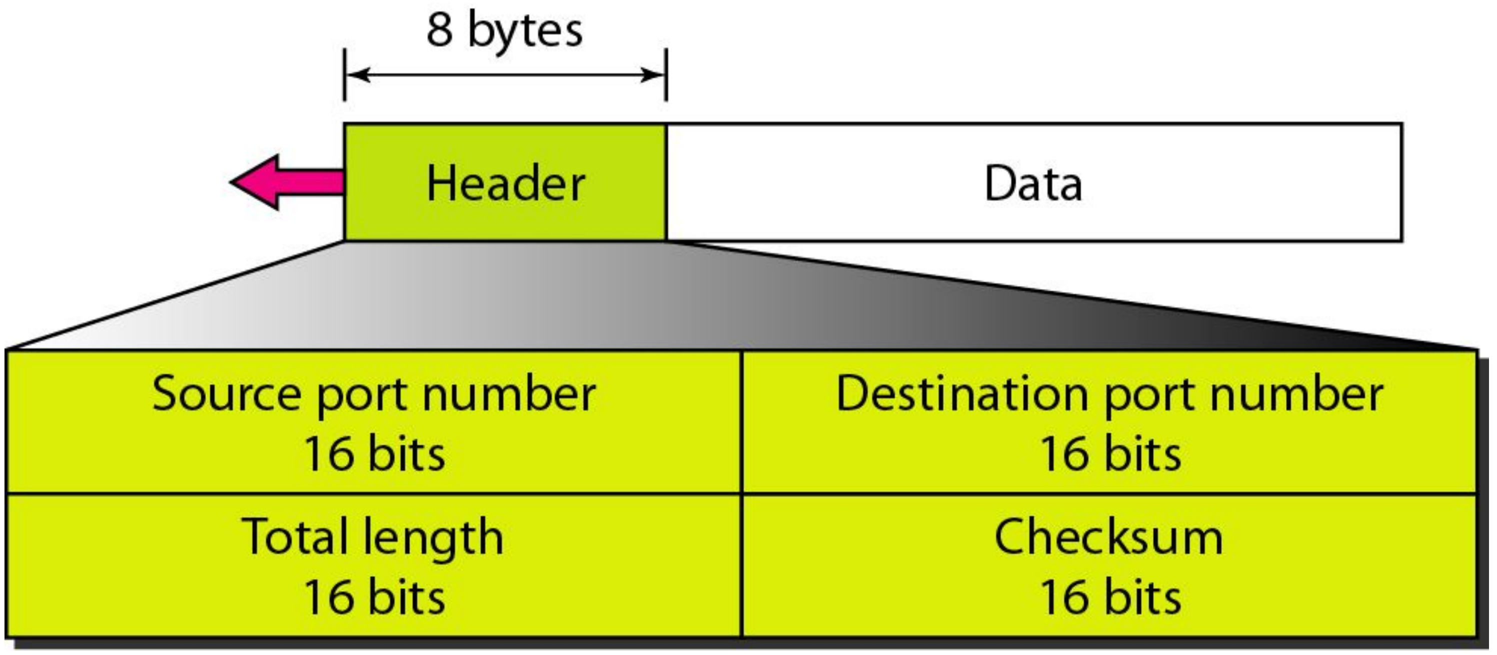
23-2 USER DATAGRAM PROTOCOL (UDP)

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

Figure 23.9 *User datagram format*

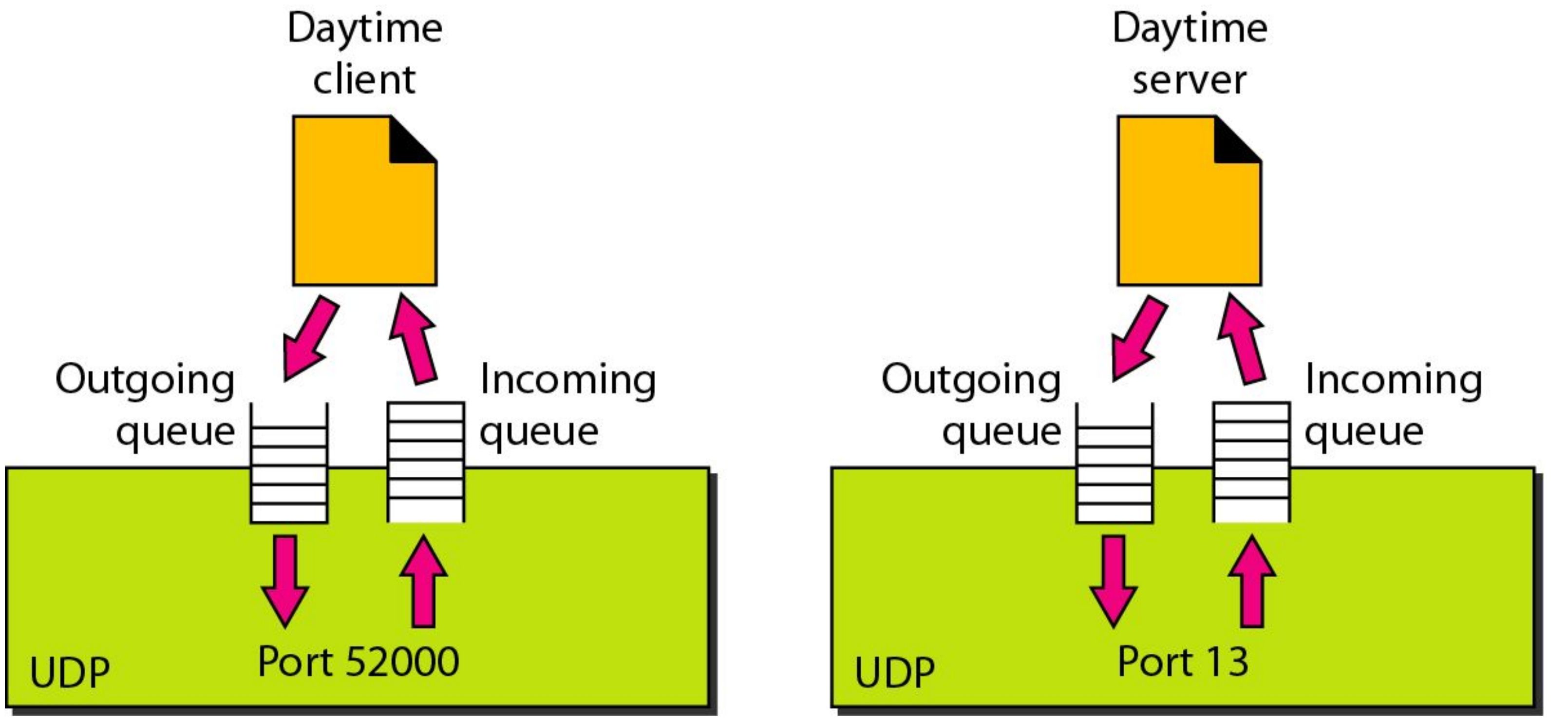


- Advantages of UDP
  - Simple Protocol
  - Minimum Overhead
- Disadvantages of UDP
  - Connectionless
  - Unreliable
  - Limited error checking

UDP Operation

- Connectionless Services
  - Each user datagram can travel on a different path
- Flow and Error Control
  - UDP is a very simple, unreliable transport protocol. There is no flow control
  - Error control is provided by only checksum
- Encapsulation and Decapsulation
  - UDP protocol encapsulates and decapsulates messages in an IP datagram
- Queuing
  - With one port number one outgoing and one incoming queue is associated

Figure 23.12 *Queues in UDP*



Uses of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control

- UDP is suitable for a process with internal flow and error control mechanism

- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software

- UDP is used for management processes such as SNMP

- UDP is used for some route updating protocols such as Routing Information Protocol (RIP)

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

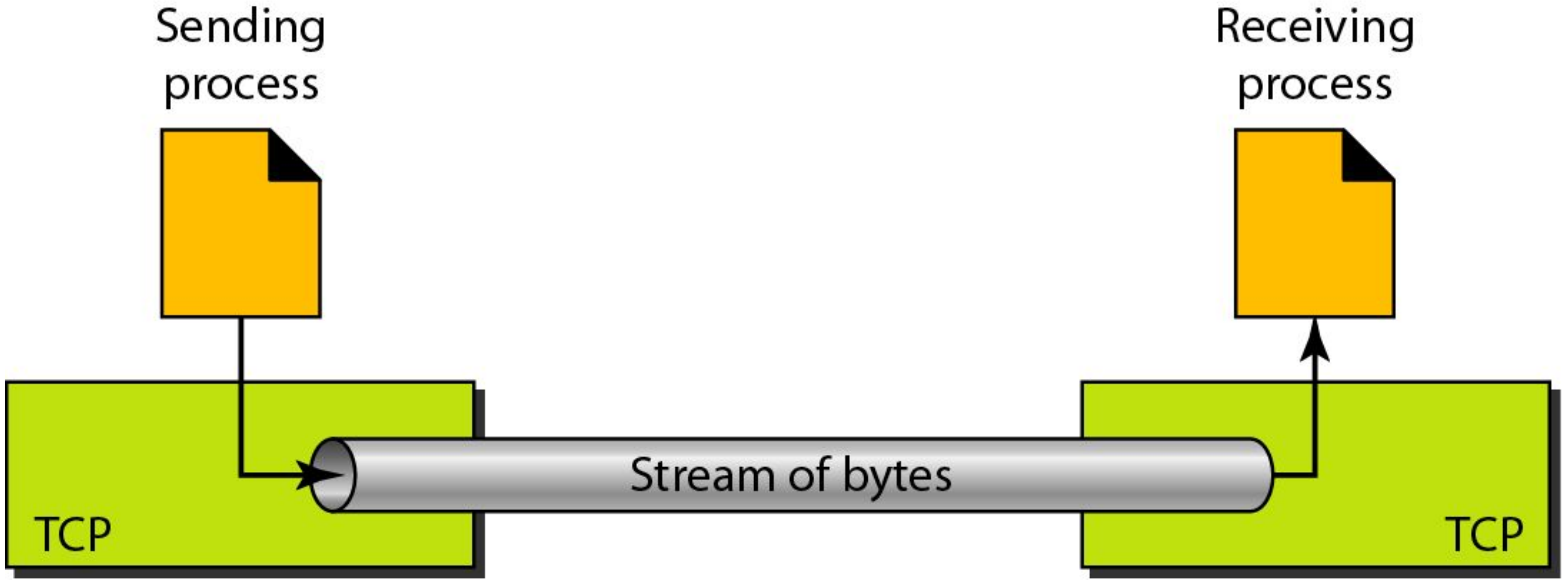
TCP Services

- Like UDP, TCP provides process-to-process communication using port numbers

- Unlike UDP, TCP provides reliable service by using an acknowledgement service

- TCP, unlike UDP, is a stream-oriented protocol that allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes

Figure 23.13 *Stream delivery*



Connection-Oriented Service

- When a process at site A wants to send and receive data from another process at site B, the following occurs:
  - The two TCPs establish a connection between them
  - Data are exchanged in both directions
  - The connection is terminated

TCP Features

- Numbering System
- Sequence Number
- Acknowledgement Number
- Flow Control
- Error Control
- Congestion Control

Figure 23.16 *TCP segment format*

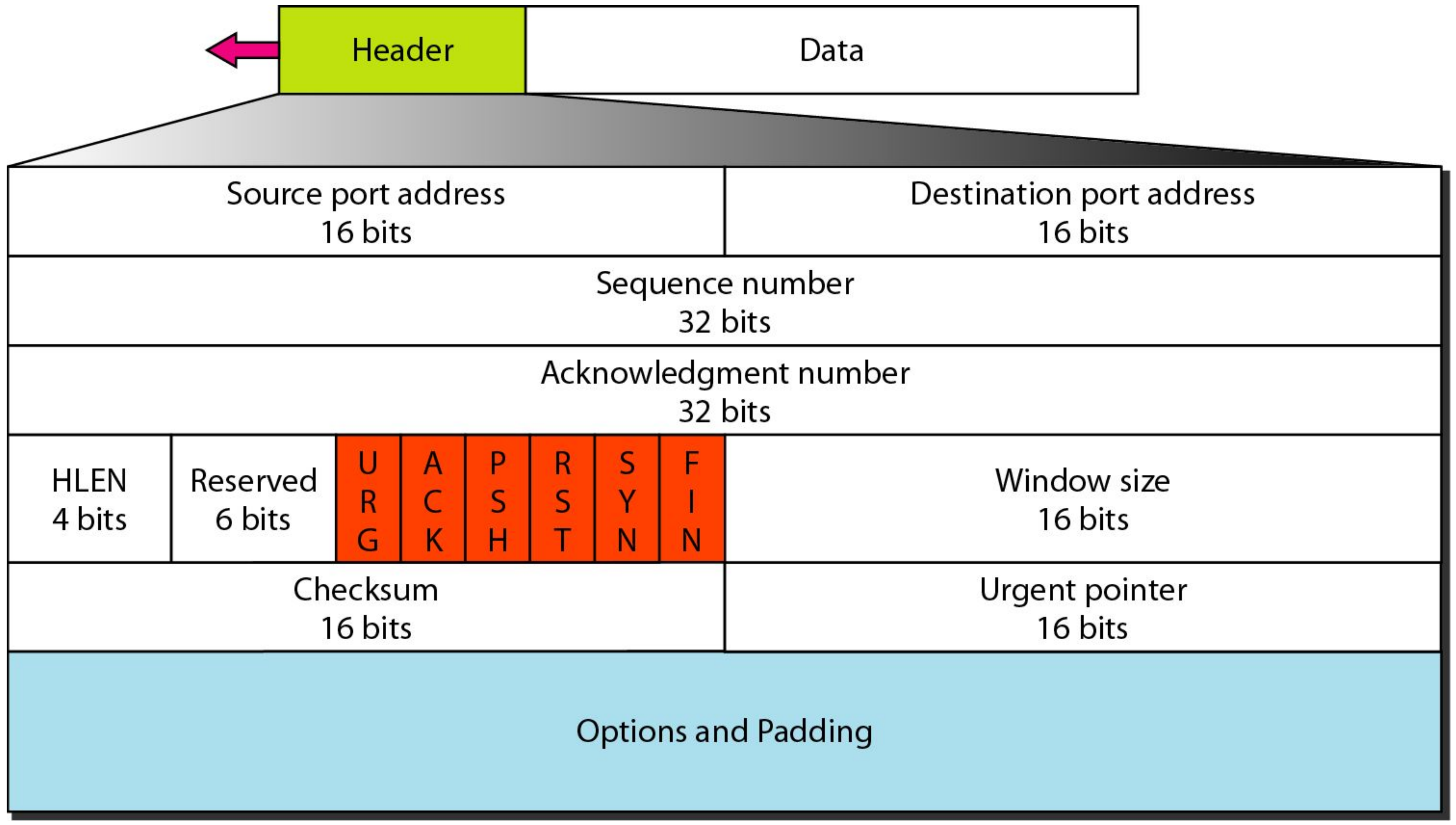


Figure 23.17 *Control field*

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

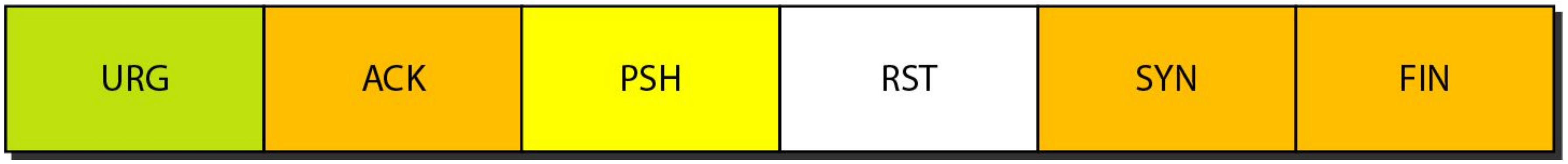


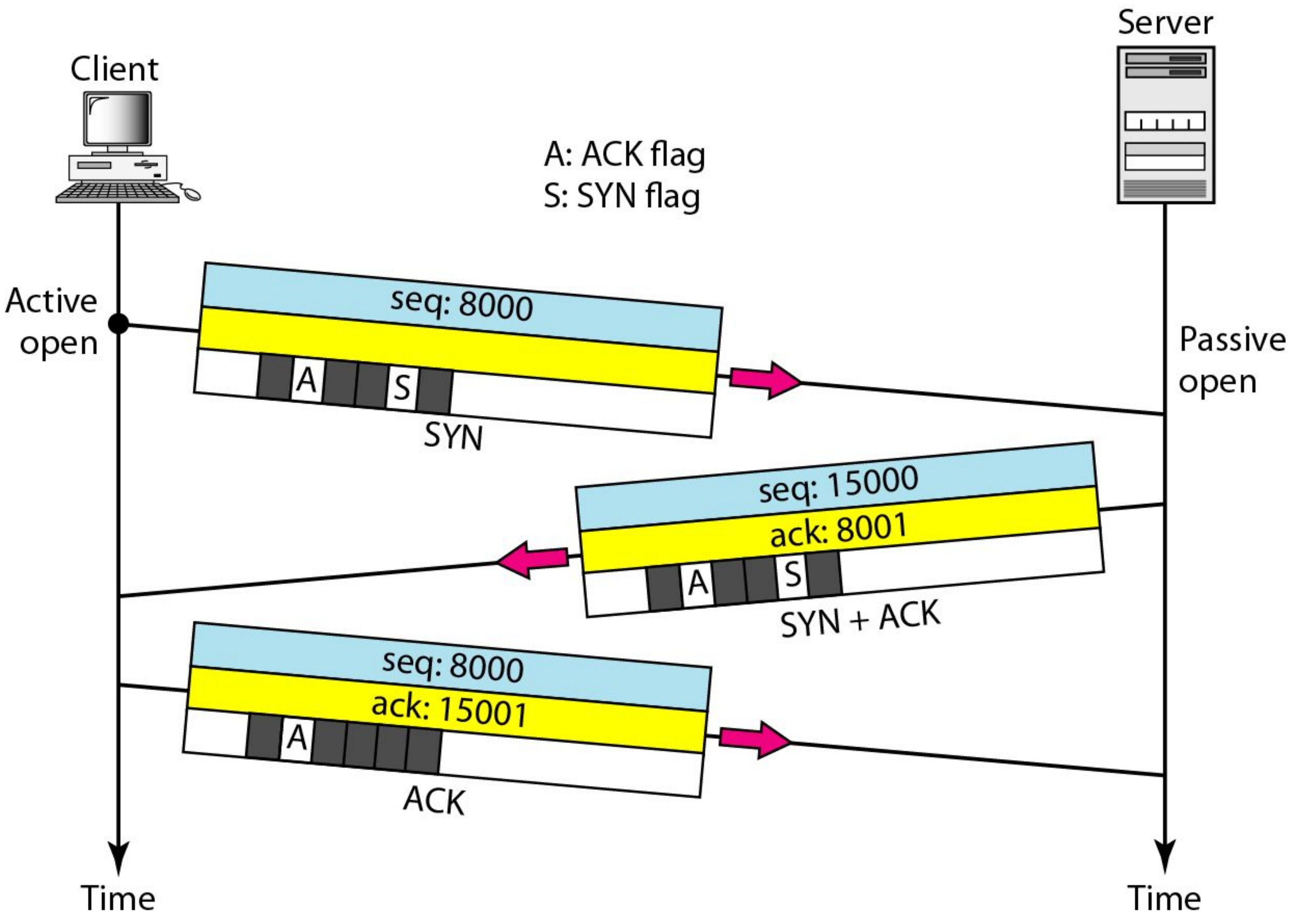
Table 23.3 ***Description of flags in the control field***

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.

TCP Connection

- In TCP, connection-oriented transmission requires three phases
  - Connection Establishment
  - Data Transfer
  - Connection Termination

Figure 23.18 *Connection establishment using three-way handshaking*



Note

A SYN segment cannot carry data, but it consumes one sequence number.

Note

A SYN + ACK segment cannot carry data, but does consume one sequence number.

Note

An ACK segment, if carrying no data, consumes no sequence number.

Figure 23.19 *Data transfer*

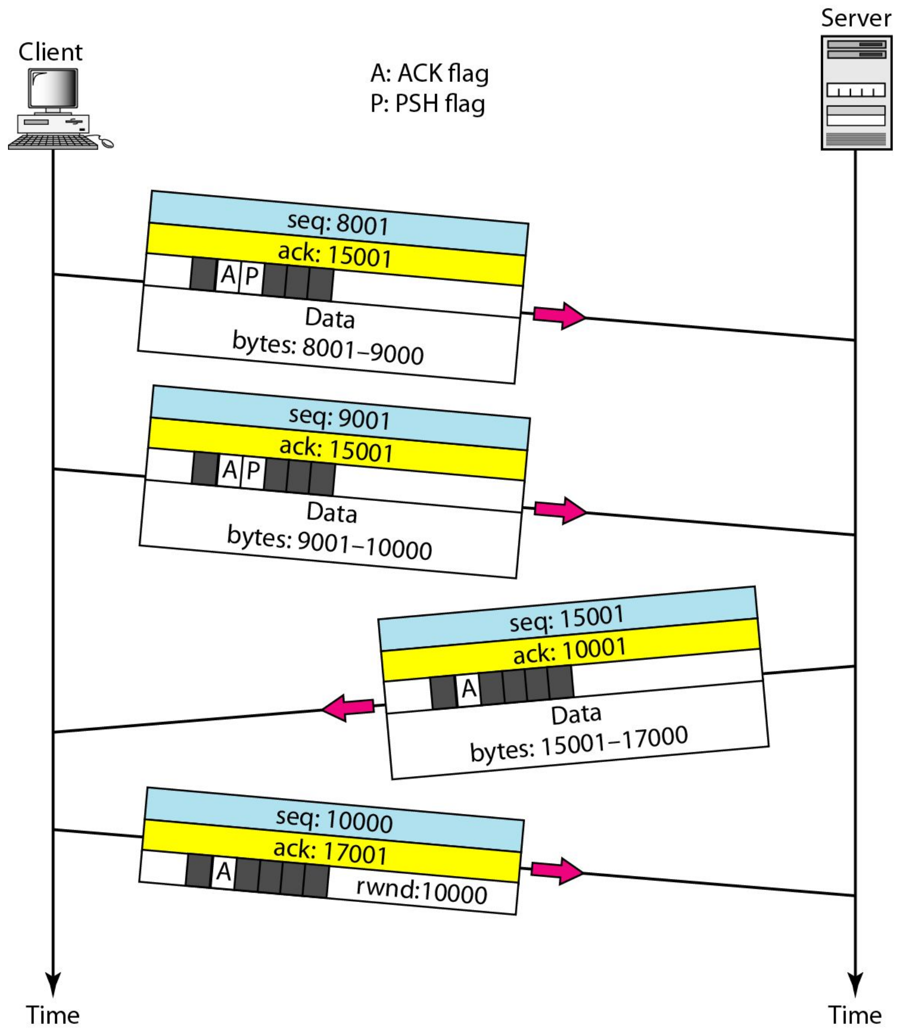
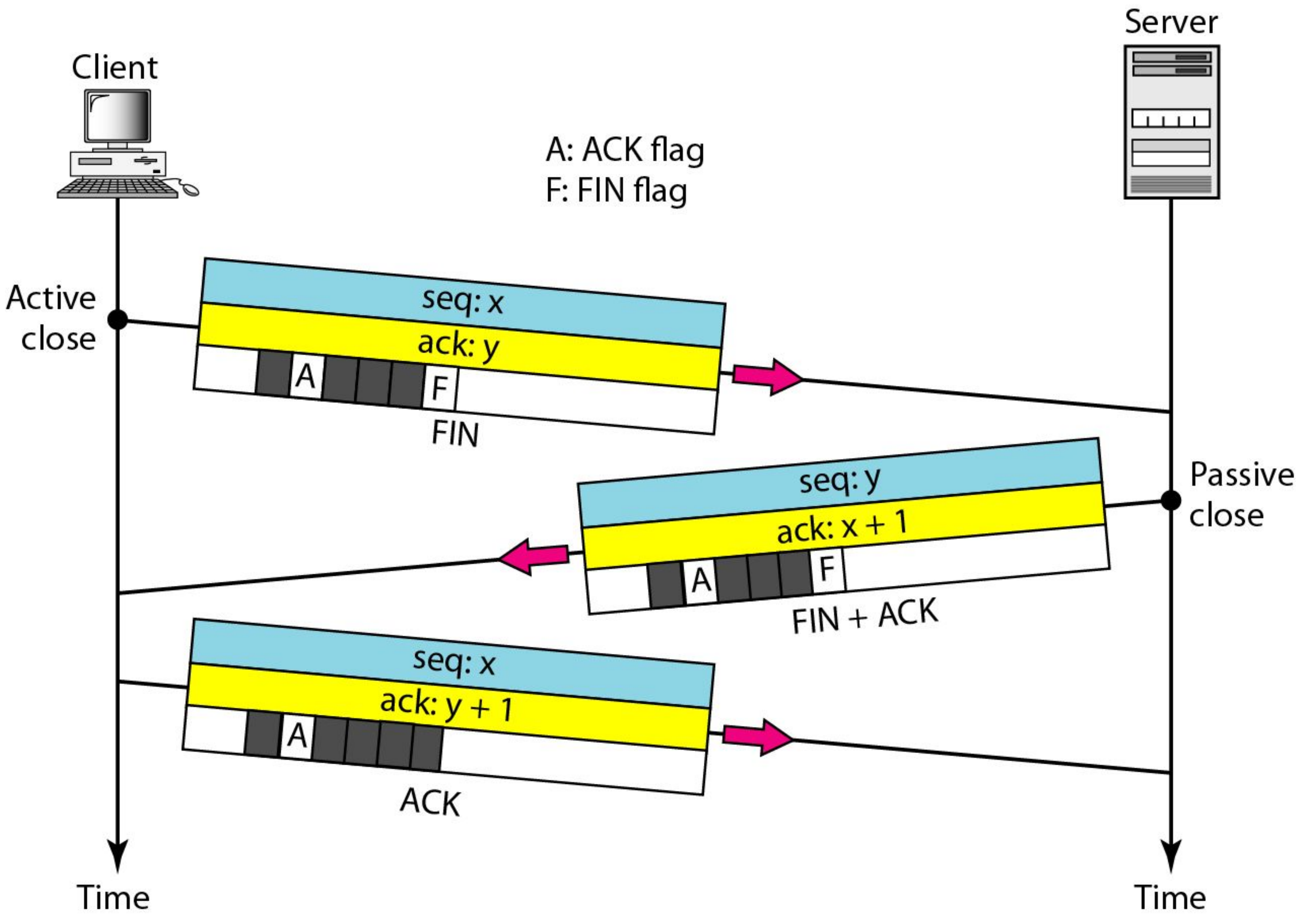


Figure 23.20 *Connection termination using three-way handshaking*



Note

The FIN segment consumes one sequence number if it does not carry data.

Note

The FIN + ACK segment consumes one sequence number if it does not carry data.

Note

ACK segments do not consume sequence numbers and are not acknowledged.

Note

Data may arrive out of order and be temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process.

Note

The receiver TCP delivers only ordered data to the process.

24-2 CONGESTION

Congestion in a network may occur if the load on the network—the number of packets sent to the network—is greater than the capacity of the network—the number of packets a network can handle. Congestion control refers to the mechanisms and techniques to control the congestion and keep the load below the capacity.

~~Topics discussed in this section:~~

Network Performance

Figure 24.3 ***Queues in a router***

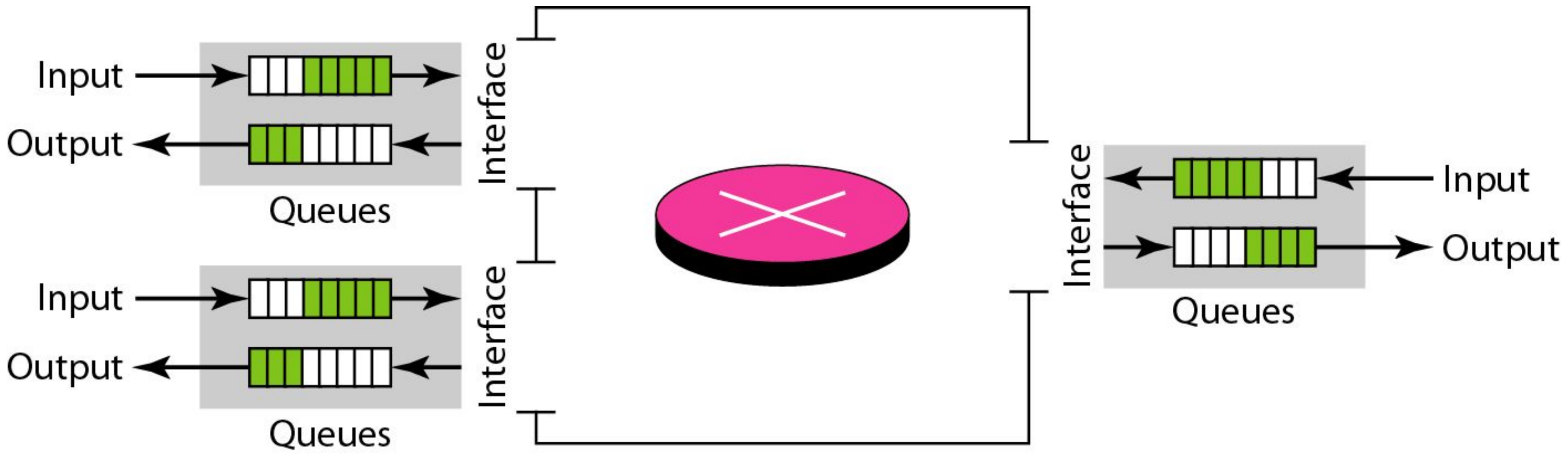
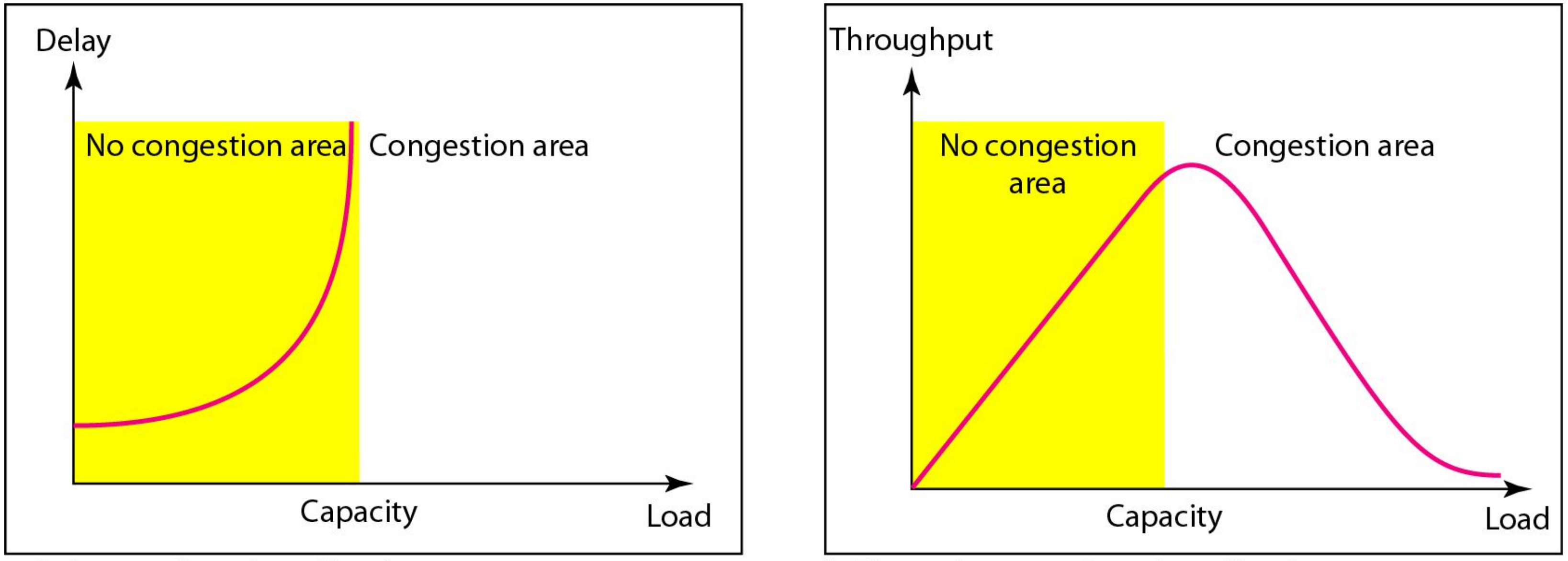


Figure *Packet delay and throughput as functions of load*



a. Delay as a function of load

b. Throughput as a function of load

24-3 CONGESTION CONTROL

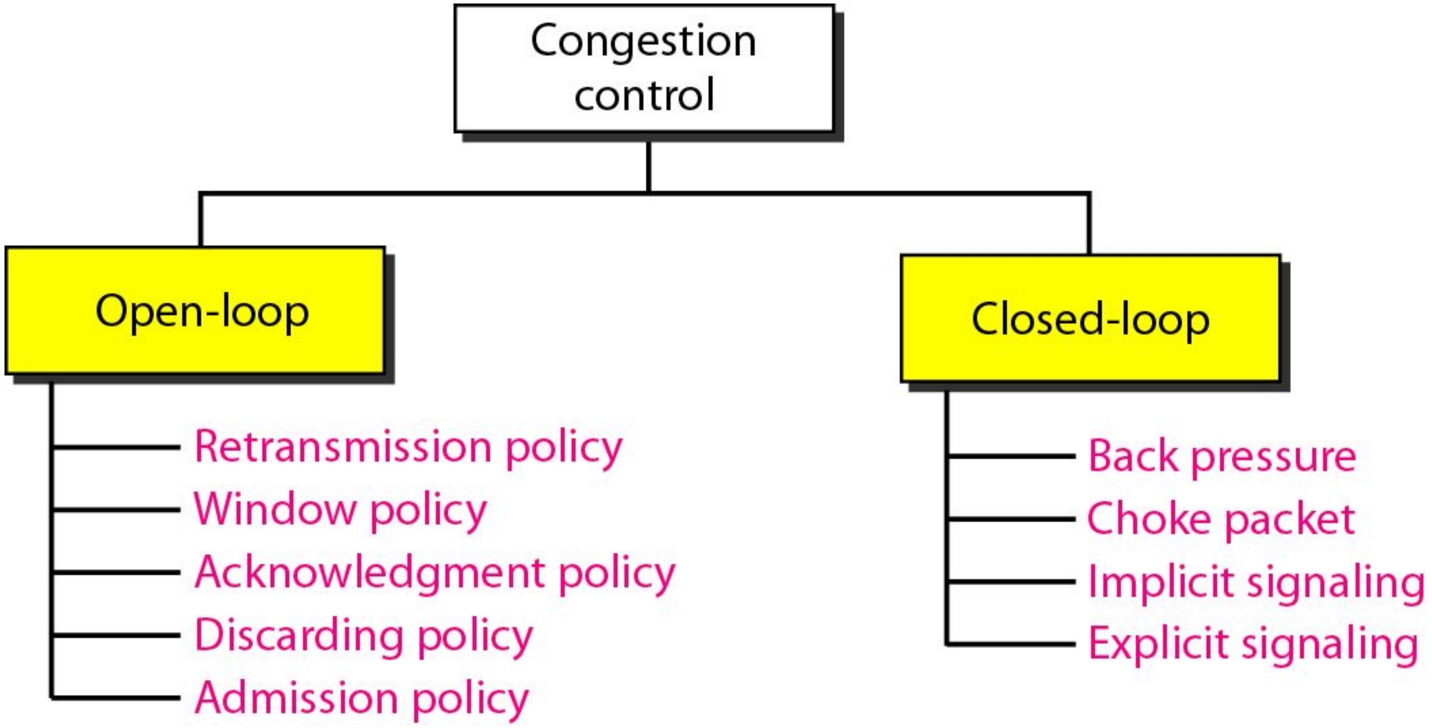
Congestion control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. In general, we can divide congestion control mechanisms into two broad categories: open-loop congestion control (prevention) and closed-loop congestion control (removal).

~~Topics discussed in this section:~~

Open-Loop Congestion Control

Closed-Loop Congestion Control

Figure 24.5 *Congestion control categories*



Open-Loop Congestion Control

- In open-loop congestion control, policies are applied to prevent congestion before it happens
- congestion control is handled by either the source or the destination.

Cont..

- Retransmission Policy
  - The retransmission policy and the retransmission timers must be designed to optimize efficiency and at the same time prevent congestion
- Window Policy
  - The type of window at the sender may also affect congestion

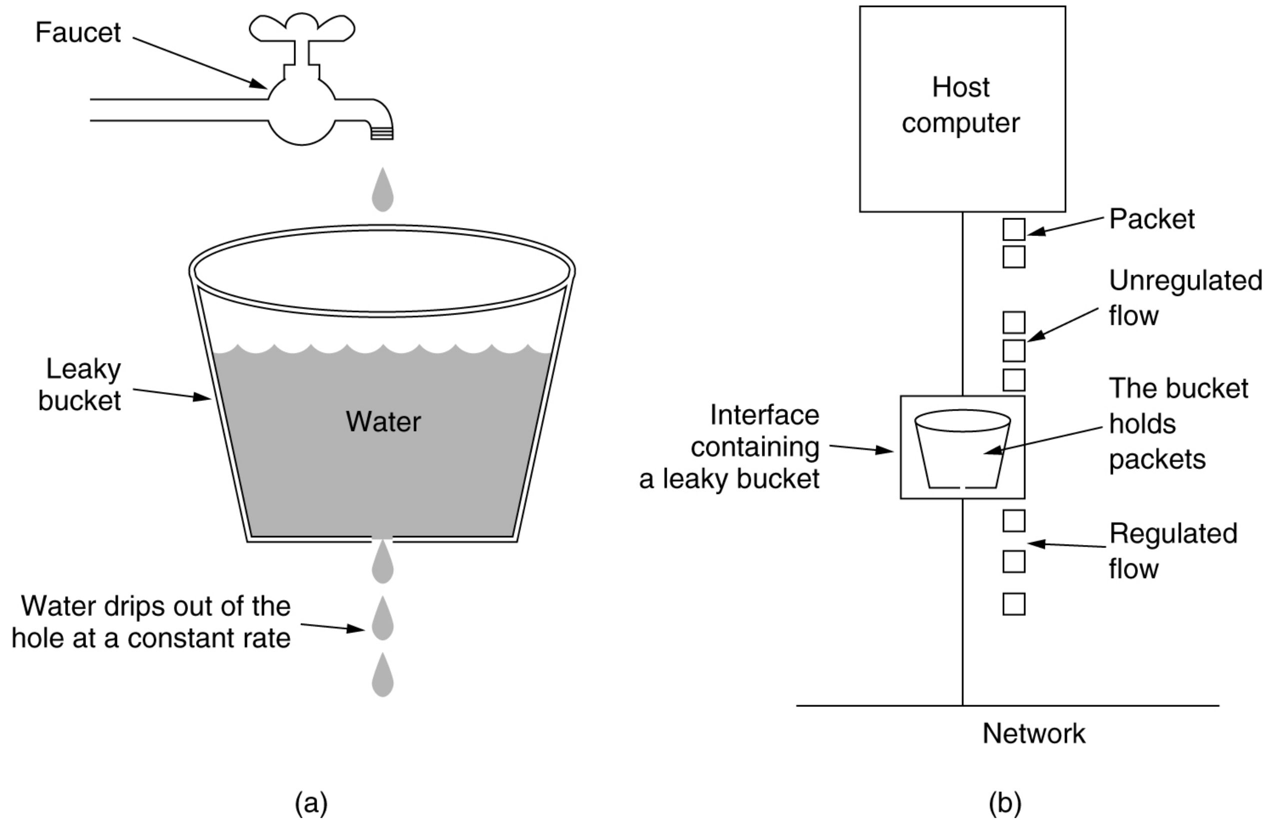
- Acknowledgment Policy
  - The acknowledgment policy imposed by the receiver may also affect congestion. If the receiver does not acknowledge every packet it receives, it may slow down the sender and help prevent congestion
- Discarding Policy
  - A good discarding policy by the routers may prevent congestion and at the same time may not harm the integrity of the transmission
- Admission Policy
  - Switches in a flow first check the resource requirement of a flow before admitting it to the network

- Traffic shaping controls the *rate* at which packets are sent (not just how many)
- At connection set-up time, the sender and carrier negotiate a traffic pattern (shape)
- Two traffic shaping algorithms are:
  - Leaky Bucket
  - Token Bucket

The Leaky Bucket Algorithm

- The **Leaky Bucket Algorithm** used to control rate in a network.
- It is implemented as a single-server queue with constant service time. If the bucket (buffer) overflows then packets are discarded.

The Leaky Bucket Algorithm



(a) A leaky bucket with water (b) a leaky bucket with packets.

Leaky Bucket Algorithm (contd.)

- The leaky bucket enforces a constant output rate regardless of the burstiness of the input. Does nothing when input is idle.

- The host injects one packet per clock tick onto the network. This results in a uniform flow of packets, smoothing out bursts and reducing congestion.

- When packets are the same size (as in ATM cells), the one packet per tick is okay. For variable length packets though, it is better to allow a fixed number of bytes per tick.

Token Bucket Algorithm

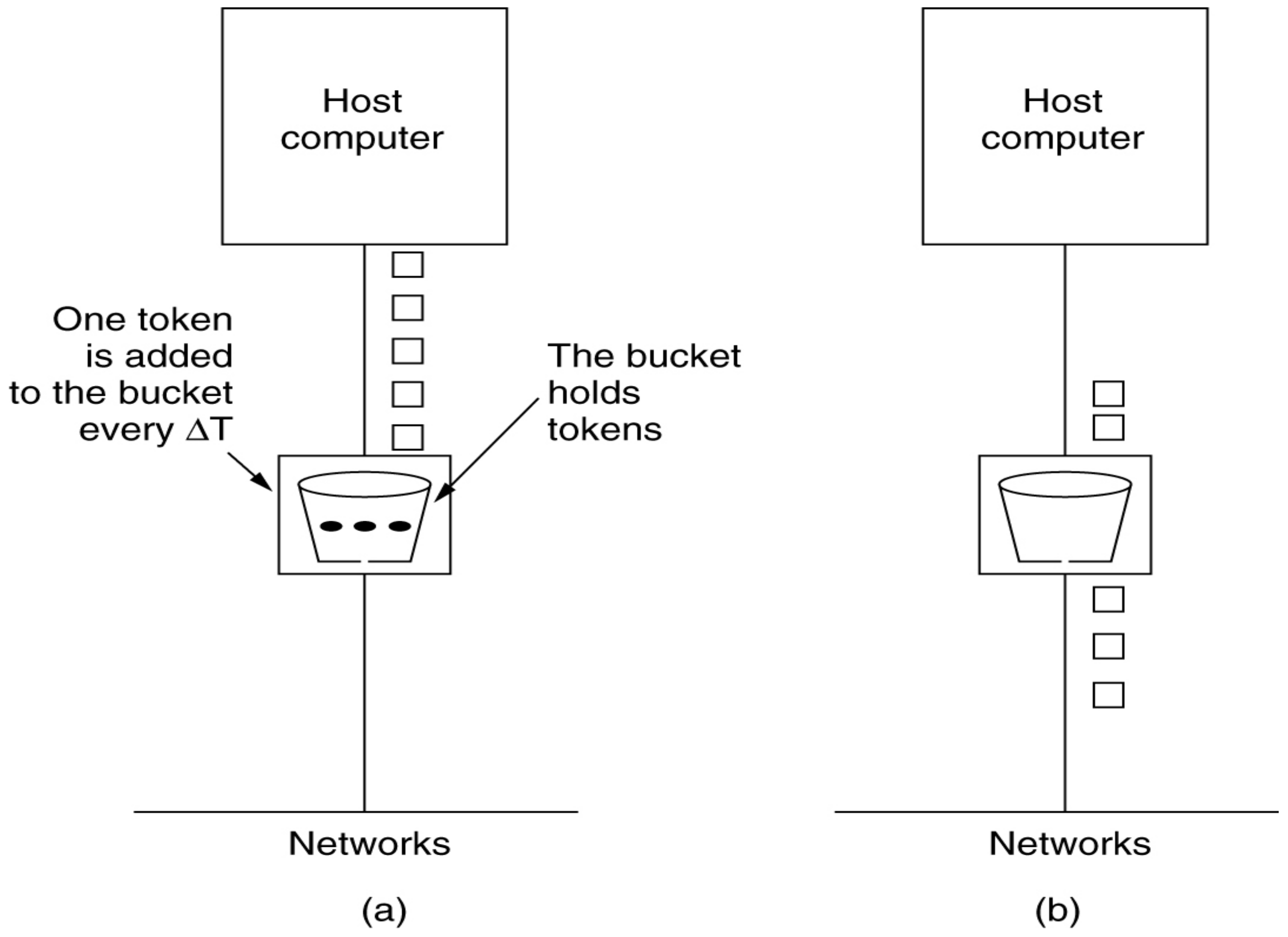
- In contrast to the LB, the Token Bucket (TB) algorithm, allows the output rate to vary, depending on the size of the burst.

- In the TB algorithm, the bucket holds tokens. To transmit a packet, the host must capture and destroy one token.

- Tokens are generated by a clock at the rate of one token every Δt sec.

- Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.

Token Bucket Algorithm (contd.)



(a) Before (b) After

Token bucket operation

- TB accumulates fixed size tokens in a token bucket
- Transmits a packet (from data buffer, if any are there) or arriving packet if the sum of the token sizes in the bucket add up to packet size
- More tokens are periodically added to the bucket (at rate Δt). If tokens are to be added when the bucket is full, they are discarded

Token bucket properties

- Does not bound the peak rate of small bursts, because bucket may contain enough token to cover a complete burst size
- Performance depends only on the sum of the data buffer size and the token bucket size

Token bucket - example

- 2 tokens of size 100 bytes added each second to the token bucket of capacity 500 bytes
  - Avg. rate = 200 bytes/sec, burst size = 500 bytes
  - Packets bigger than 500 bytes will never be sent
  - Peak rate is unbounded – i.e., 500 bytes of burst can be transmitted arbitrarily fast

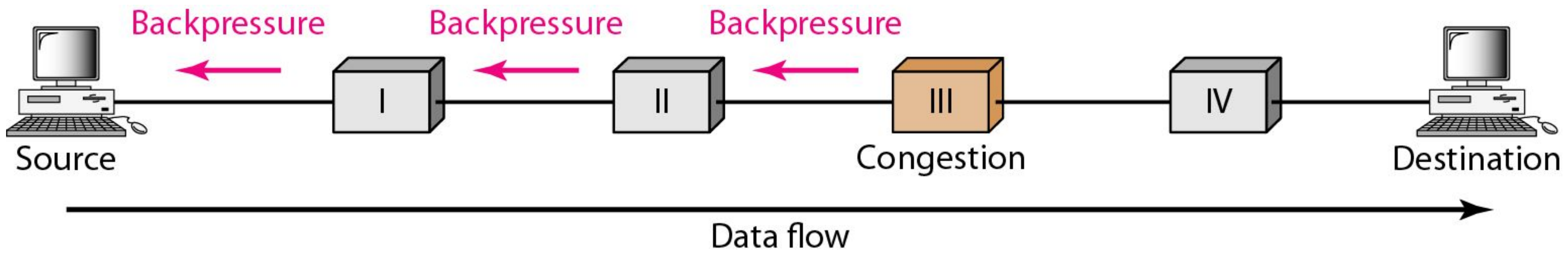
Leaky Bucket vs Token Bucket

- LB discards packets; TB does not. TB discards tokens.
- With TB, a packet can only be transmitted if there are enough tokens to cover its length in bytes.
- LB sends packets at an average rate. TB allows for large bursts to be sent faster by speeding up the output.
- TB allows saving up tokens (permissions) to send large bursts. LB does not allow saving.

Closed-Loop Congestion Control

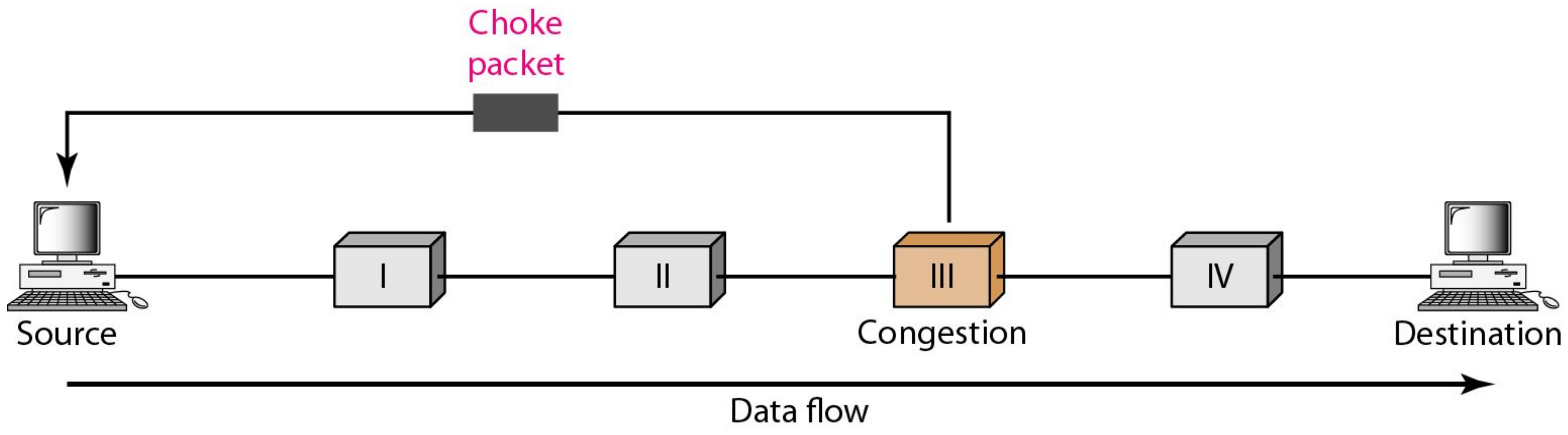
- Closed-loop congestion control mechanisms try to alleviate congestion after it happens

- Backpressure
  - The technique of backpressure refers to a congestion control mechanism in which a congested node stops receiving data from the immediate upstream node or nodes. This may cause the upstream node or nodes to become congested, and they, in turn, reject data from their upstream nodes or nodes. And so on



Choke Packet

- A choke packet is a packet sent by a node to the source to inform it of congestion



- Implicit Signaling

  - In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is a congestion somewhere in the network from other symptoms
  - For example when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested

- Explicit Signaling

  - The node that experiences congestion can explicitly send a signal to the source or destination
  - The signal is included in the packets that carry data.

- Backward Signaling
  - A bit can be set in a packet moving in the direction opposite to the congestion. This bit can warn the source that there is congestion and that it needs to slow down to avoid the discarding of packets
- Forward Signaling
  - A bit can be set in a packet moving in the direction of the congestion. This bit can warn the destination that there is congestion. The receiver in this case can use policies, such as slowing down the acknowledgments, to alleviate the congestion