Data and Computer Communications

Tenth Edition

by William Stallings

CHAPTER 15

Transport Protocols

"The foregoing observations should make us reconsider the widely held view that birds live only in the present. In fact, birds are aware of more than immediately present stimuli; they remember the past and anticipate the future."

—*The Minds of Birds,*
Alexander Skutch

Connection-Oriented Transport Mechanisms

- Two basic types of transport service:

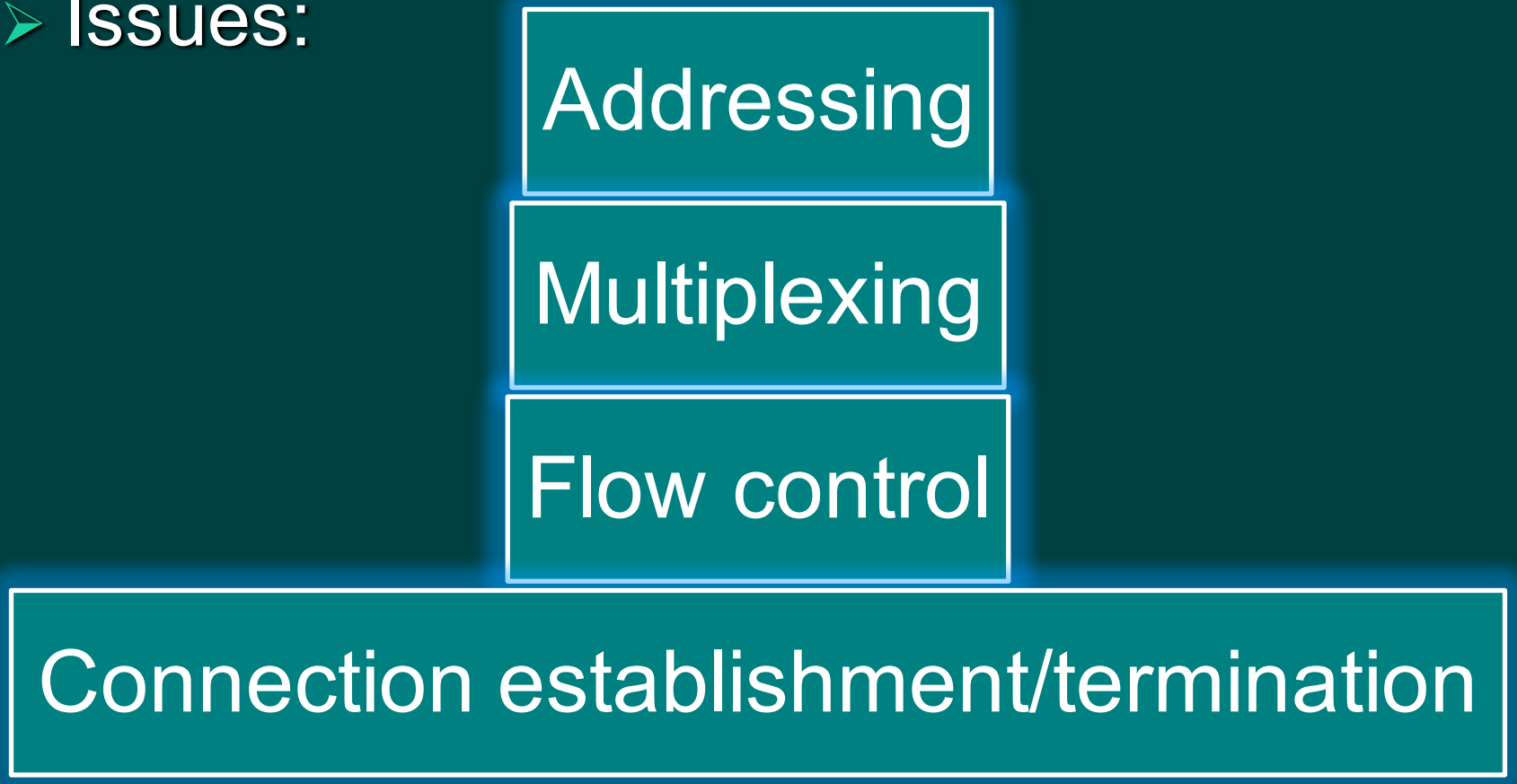
Connection-oriented

- 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

Reliable Sequencing Network Service

- Issues:



Addressing

- Transport protocol must be able to derive the following information from the TS user address:
  - User identification
  - Transport entity identification
  - Host address
  - Network number

Multiplexing

- Multiple users employ the same transport protocol and are distinguished by port numbers or service access points

Upward multiplexing

- Multiplexing of multiple connections on a single lower-level connection

Downward multiplexing

- Splitting of a single connection among multiple lower-level connections

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

Alternatives to Flow Control Requirements

Receiving transport entity can:

Do nothing

- Segments that overflow the buffer are discarded
- Sending transport entity will retransmit

Refuse to accept further segments from the network service

- Relies on network service to do the work

Use a fixed sliding window protocol

- With a reliable network service this works quite well

Use a credit scheme

- A more effective scheme to use with an unreliable network service

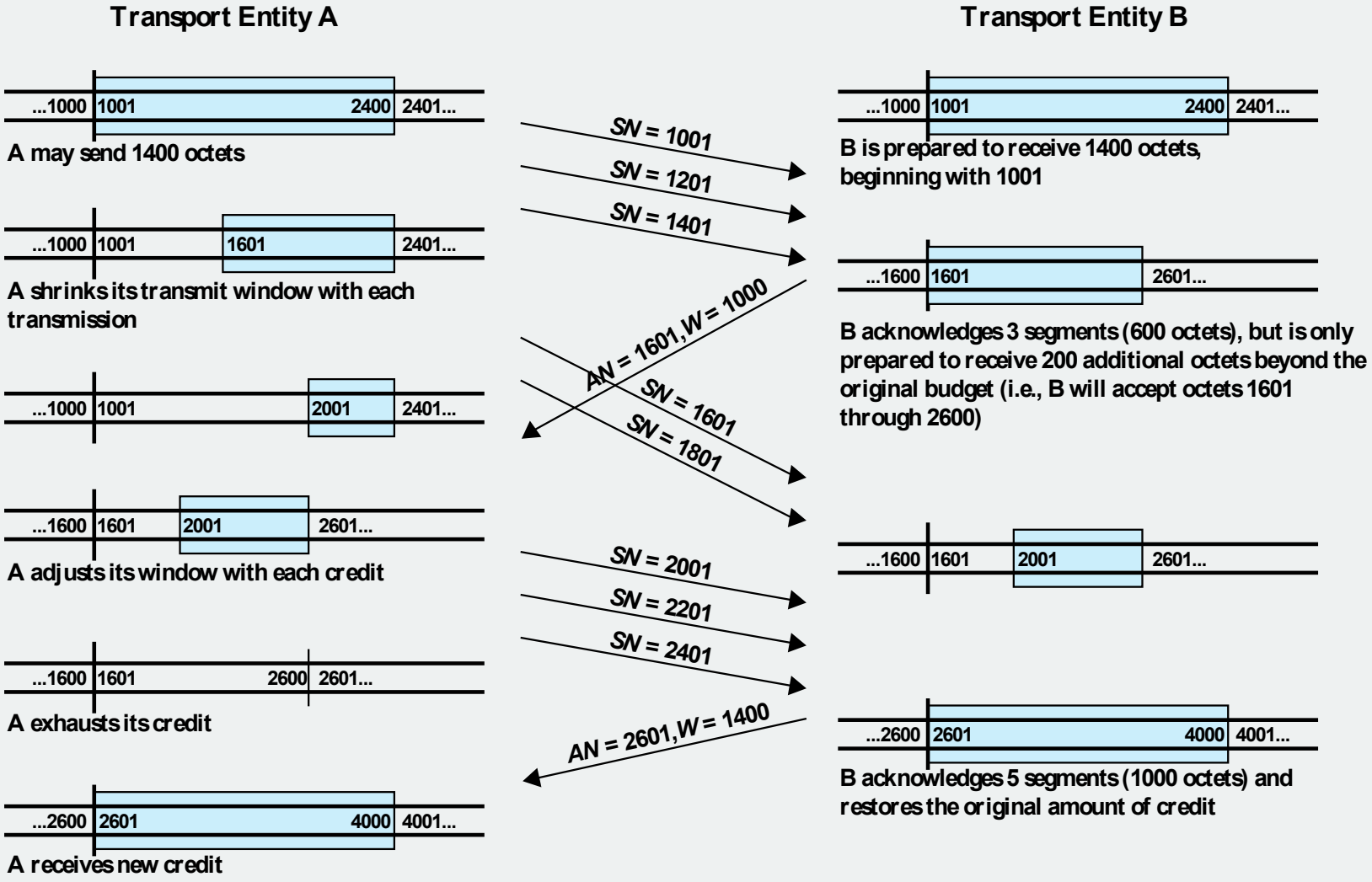
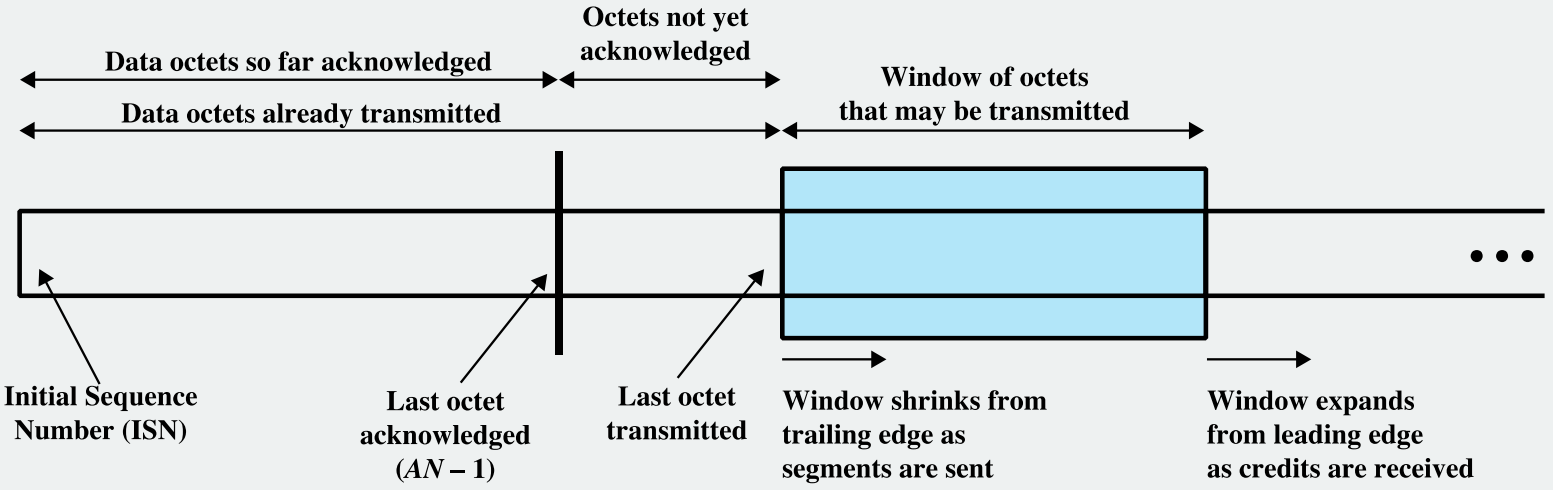
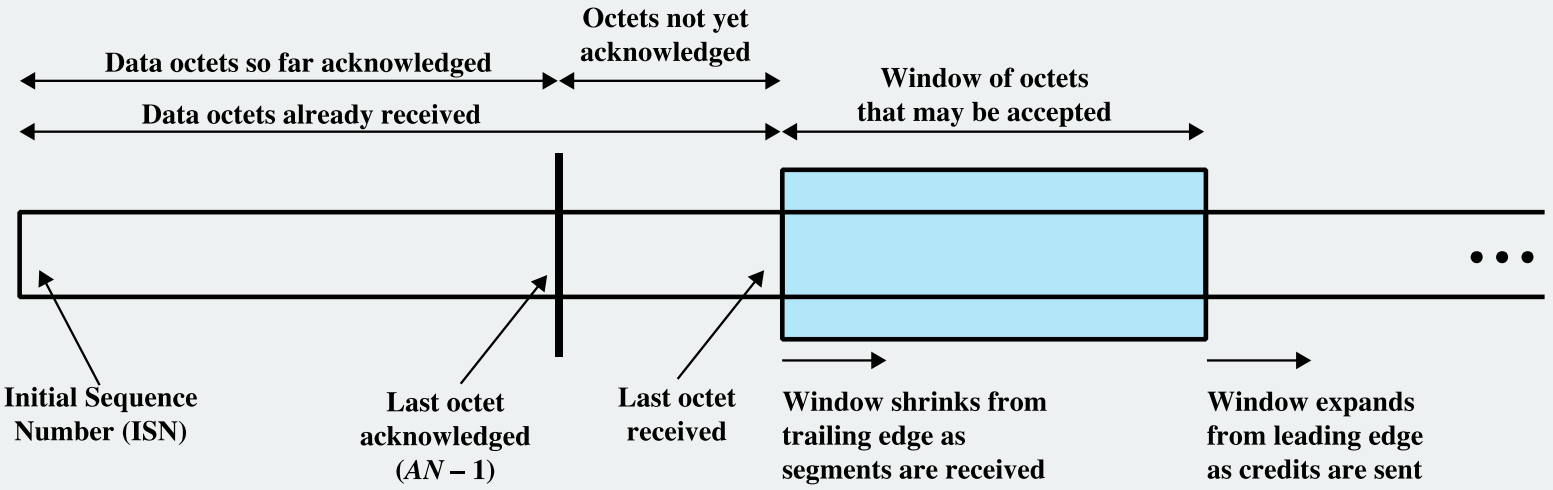


Figure 15.1 Example of TCP Credit Allocation Mechanism



(a) Send sequence space

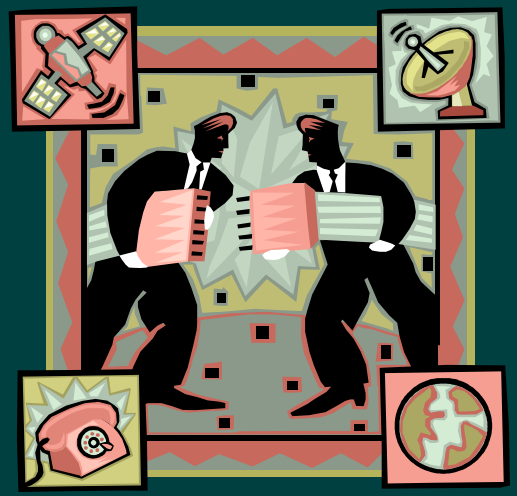


(b) Receive sequence space

Figure 15.2 Sending and Receiving Flow Control Perspectives

Connection Establishment and Termination

- Serves three main purposes:
  - Allows each end to assure that the other exists
  - Allows exchange or negotiation of optional parameters
  - Triggers allocation of transport entity resources
- Is by mutual agreement



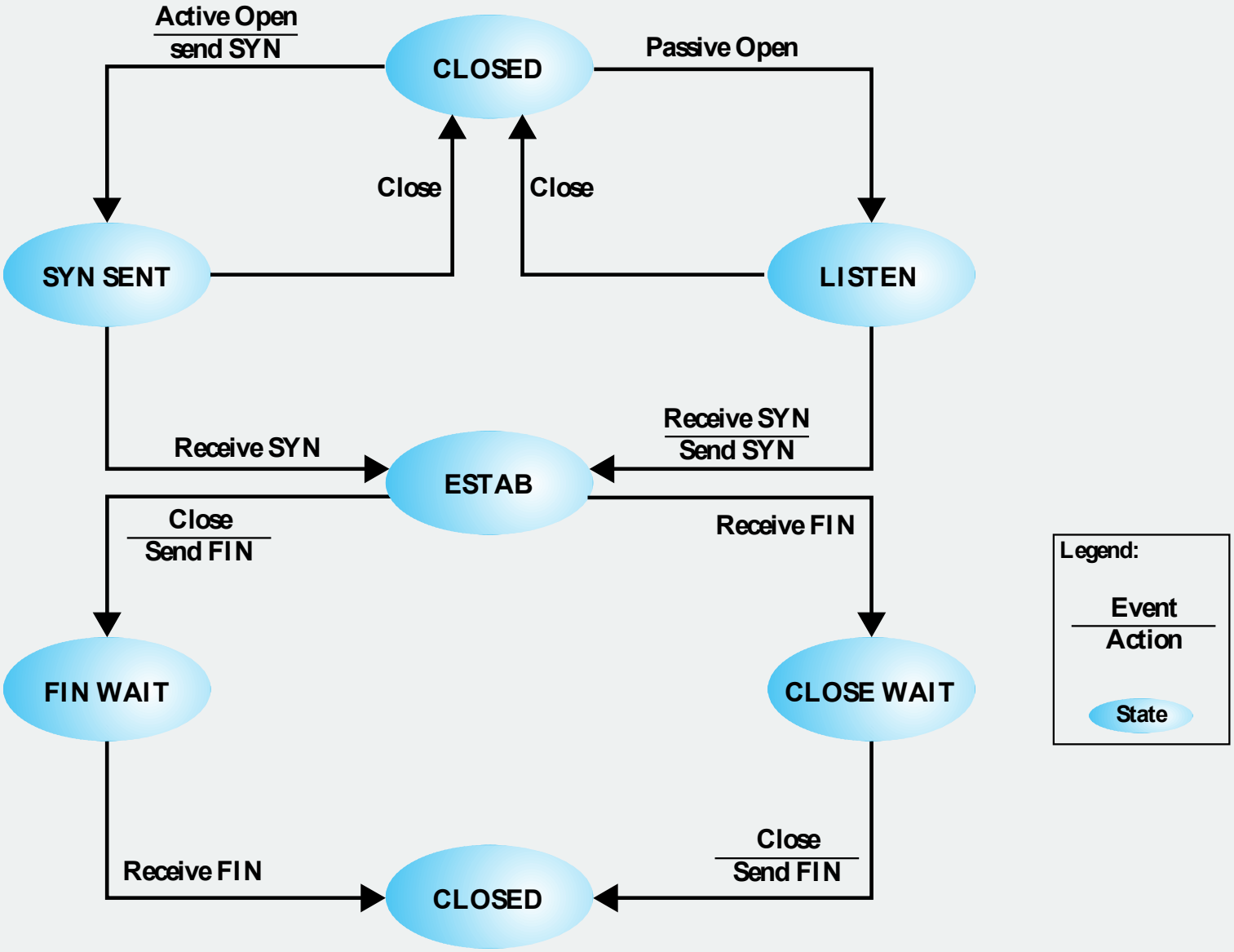


Figure 15.3 Simple Connection State Diagram

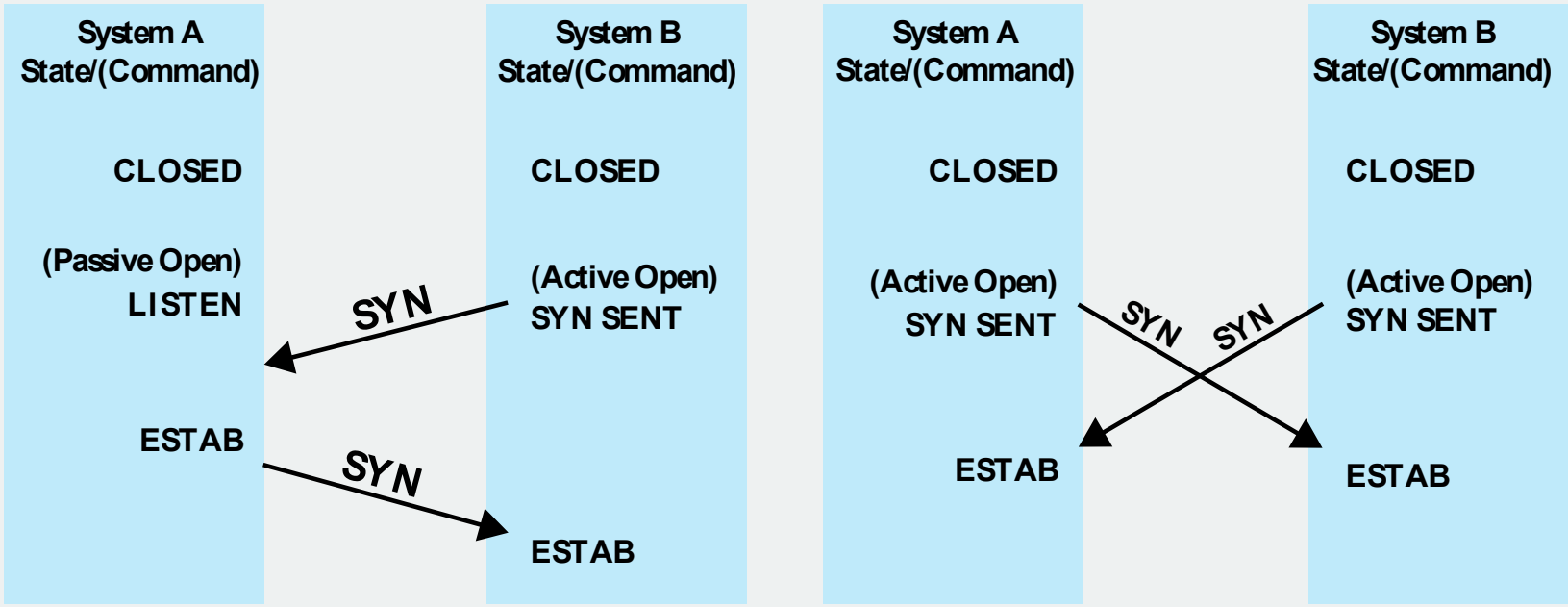


Figure 15.4 Connection Establishment Scenarios

Unreliable Network Service

Examples:

- Internetwork using IP
- Frame relay network using only the LAPF core protocol
- IEEE 802.3 LAN using the unacknowledged connectionless LLC service

- Segments are occasionally lost and may arrive out of sequence due to variable transit delays

Issues to Address

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

Ordered Delivery

- With an unreliable network service it is possible that segments may arrive out of order
- Solution to this problem is to number segments sequentially
  - TCP uses scheme where each data octet is implicitly numbered

Retransmission Strategy

- Events necessitating retransmission:

Segment may be damaged in transit but still arrives at its destination	Segment fails to arrive

- Sending transport does not know transmission was unsuccessful
- Receiver acknowledges successful receipt by returning a segment containing an acknowledgment number

Cont.

Retransmission Strategy

- No acknowledgment will be issued if a segment does not arrive successfully
  - Resulting in a retransmit
- Atimer needs to be associated with each segment as it is sent
- If timer expires before acknowledgment is received, sender must retransmit

Table 15.1
Transport Protocol Timers

Retransmission timer	Retransmit an unacknowledged segment
MSL (maximum segment lifetime) timer	Minimum time between closing one connection and opening another with the same destination address
Persist timer	Maximum time between ACK/CREDIT segments
Retransmit-SYN timer	Time between attempts to open a connection
`Keepalive timer`	Abort connection when no segments are received

Duplicate Detection

- Receiver must be able to recognize duplicates
- Segment sequence numbers help
- Complications arise if:
  - A duplicate is received prior to the close of the connection
    - Sender must not get confused if it receives multiple acknowledgments to the same segment
    - Sequence number space must be long enough
  - A duplicate is received after the close of the connection

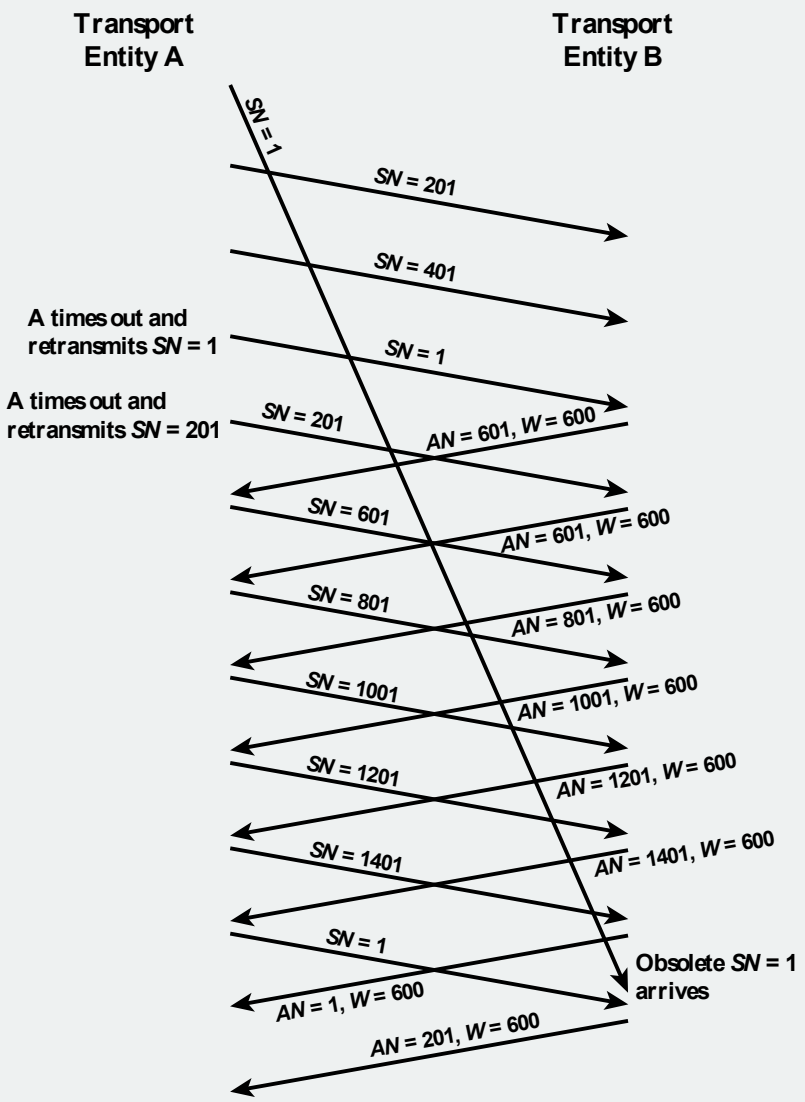


Figure 15.5 Example of Incorrect Duplicate Detection

Flow Control

- Future acknowledgments will resynchronize the protocol if an ACK/CREDIT segment is lost
- If no new acknowledgments are forthcoming the sender times out and retransmits a data segment which triggers a new acknowledgment
- Still possible for deadlock to occur

Connection Establishment

- Must take into account the unreliability of a network service
- Calls for the exchange of SYNs (two way handshake)
  - Could result in:
    - Duplicate SYNs
    - Duplicate data segments

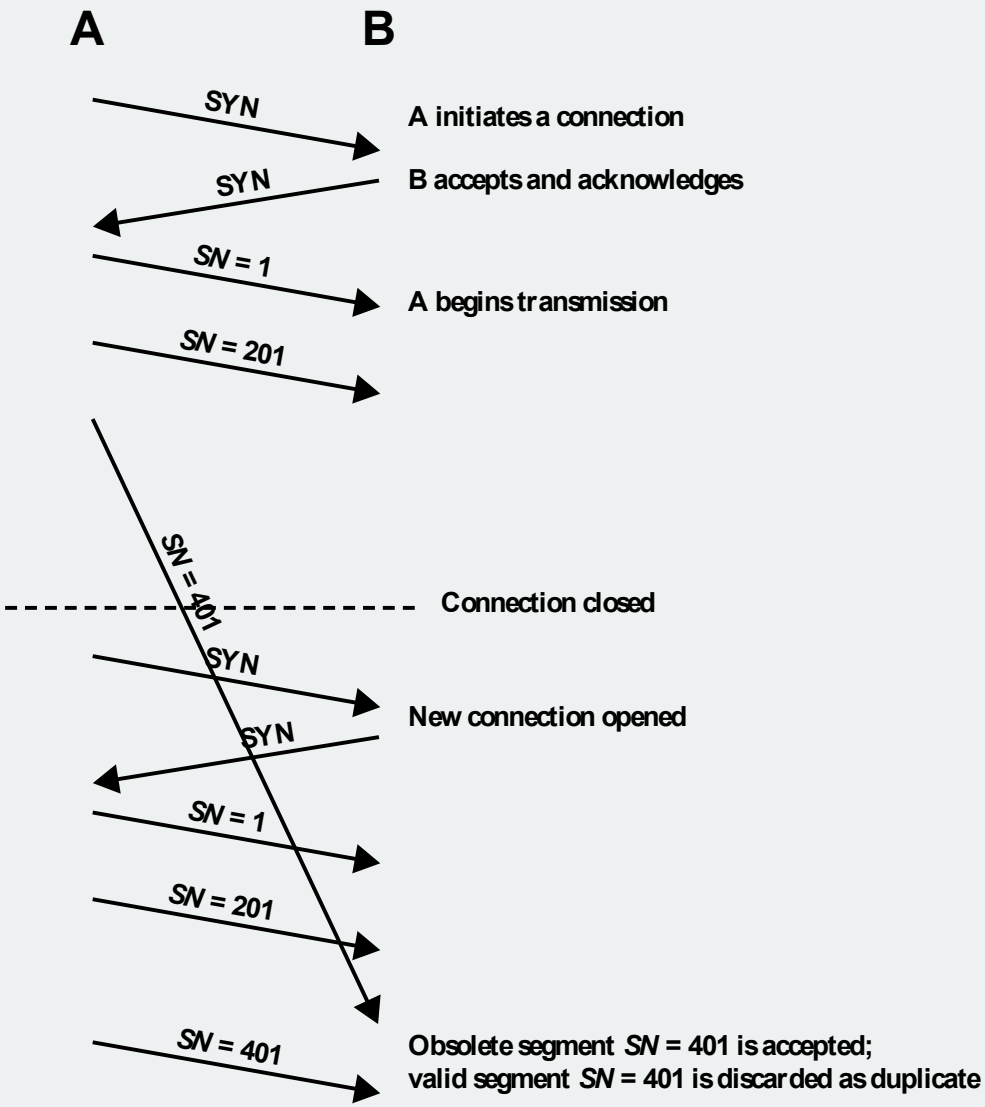


Figure 15.6 The Two-Way Handshake: Problem with Obsolete Data Segment

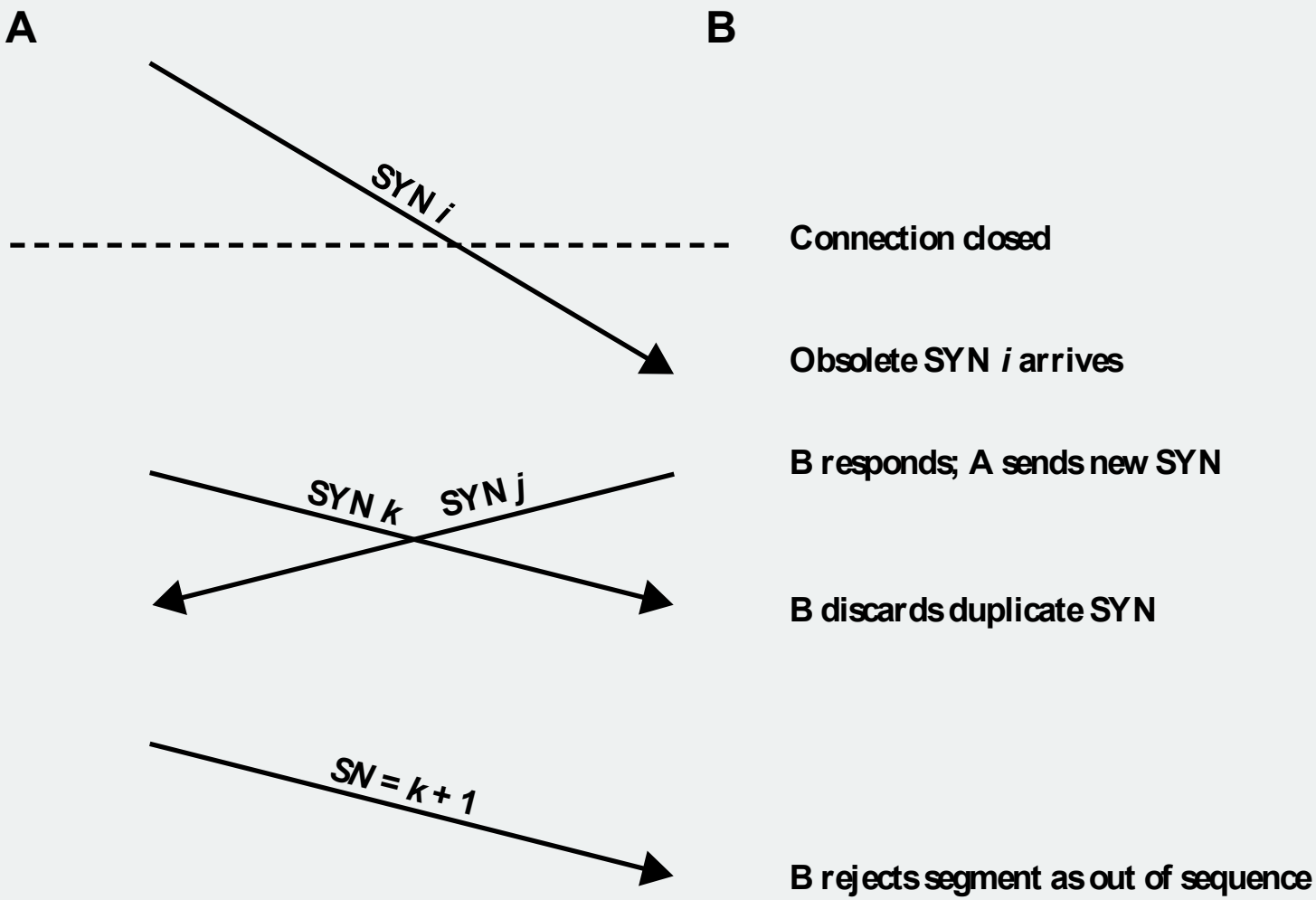
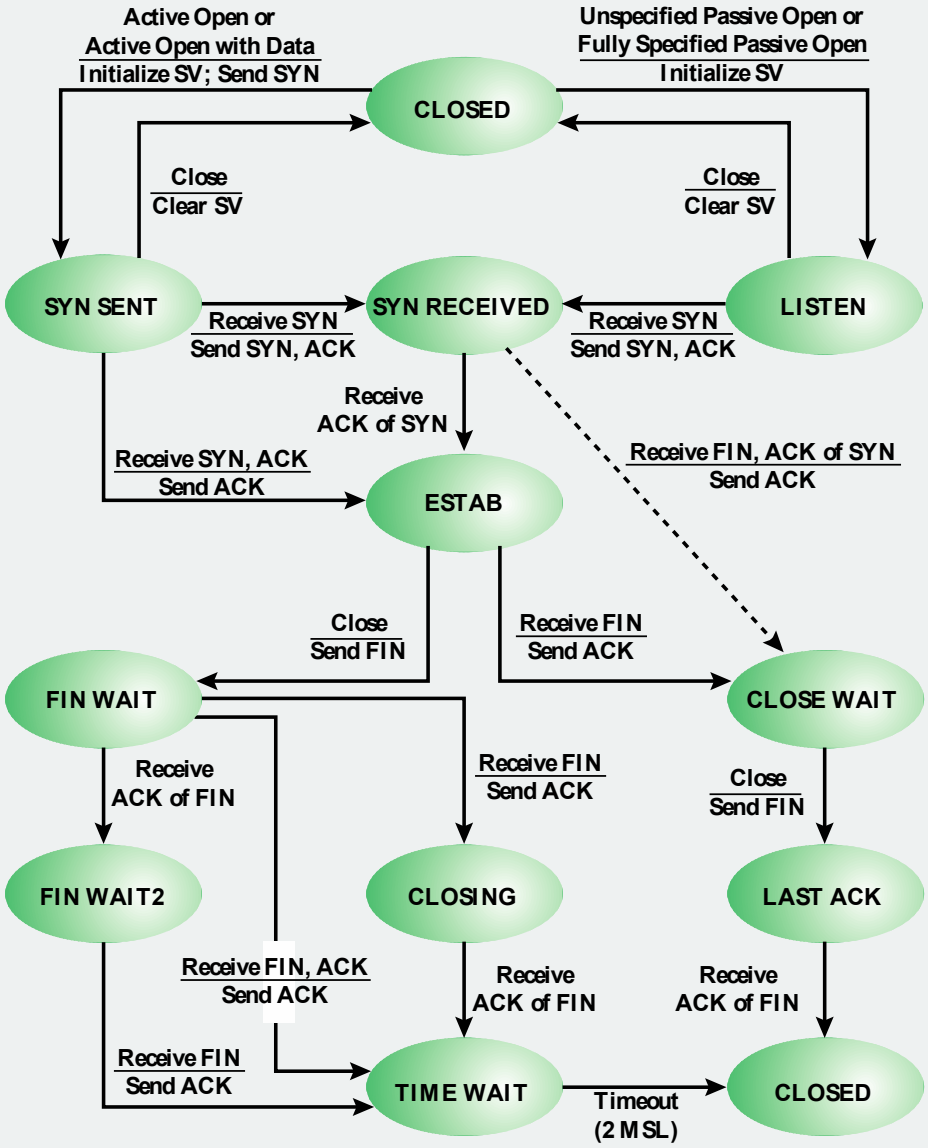


Figure 15.7 Two-Way Handshake: Problem with Obsolete SYN Segments



SV = state vector
MSL = maximum segment lifetime

Figure 15.8 TCP Entity State Diagram

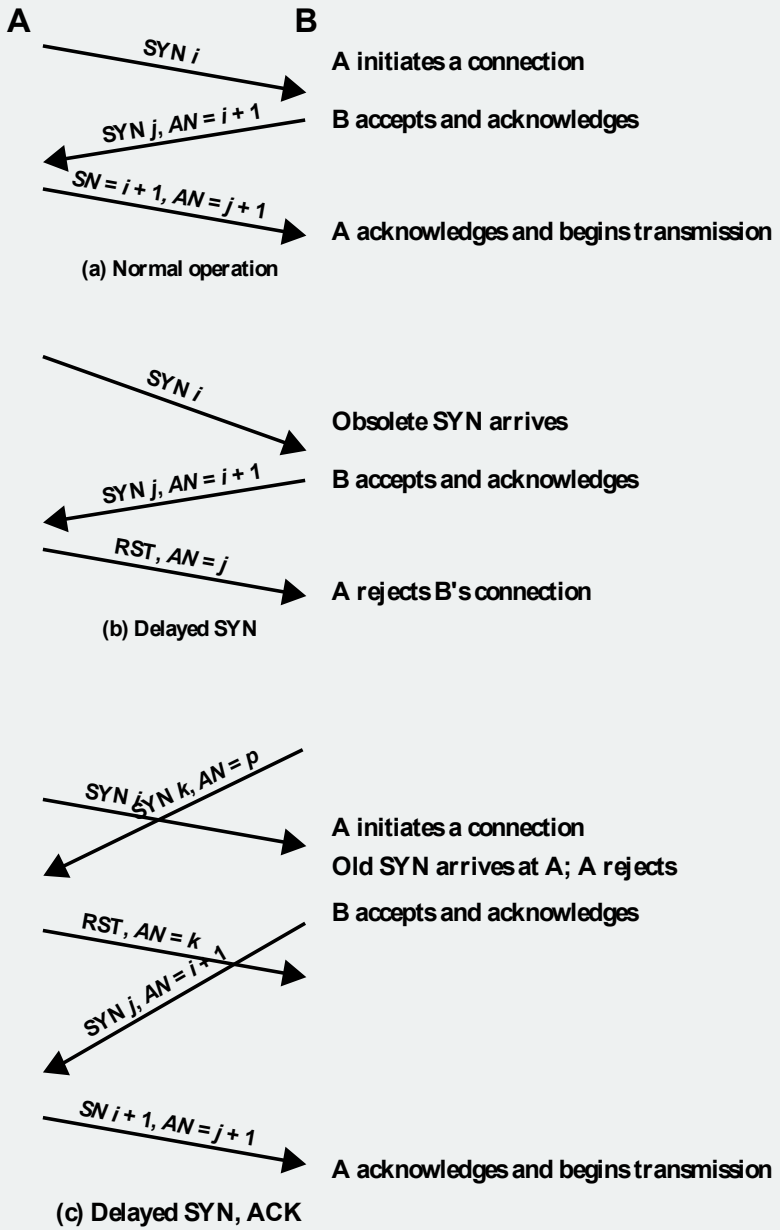


Figure 15.9 Examples of Three-Way Handshake

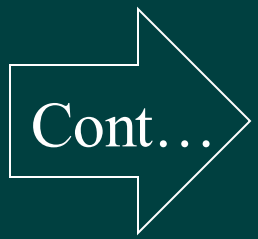
Connection Termination

- Two-way handshake was found to be inadequate for an unreliable network service

- Out of order segments could cause the FIN segment to arrive before the last data segment
  - To avoid this problem the next sequence number after the last octet of data can be assigned to FIN
  - Each side must explicitly acknowledge the FIN of the other using an ACK with the sequence number of the FIN to be acknowledged

Failure Recovery

- When the system that the transport entity is running on fails and subsequently restarts, the state information of all active connections is lost
  - Affected connections become half open because the side that did not fail does not realize the problem
    - Still active side of a half-open connection can close the connection using a ***keepalive*** timer



Failure Recovery

- In the event that a transport entity fails and quickly restarts, half-open connections can be terminated more quickly by the the use of the RST segment
  - Failed side returns an RST *i* to every segment *i* that it receives
  - RST *i* must be checked for validity on the other side
    - If valid an abnormal termination occurs
- There is still the chance that some user data will be lost or duplicated

TCP Services

- RFC 793

TCP labels data as:
• Data stream Push • Urgent data signaling

- Defined in terms of primitives and parameters

Table 15.2
TCP Service Request Primitives

Primitive	Parameters	Description
Unspecified Passive Open	source-port, [timeout], [timeout-action], [precedence], [security-range]	Listen for connection attempt at specified security and precedence from any remote destination.
Fully Specified Passive Open	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security-range]	Listen for connection attempt at specified security and precedence from specified destination.
Active Open	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security]	Request connection at a particular security and precedence to a specified destination.
Active Open with Data	source-port, destination-port, destination-address, [timeout], [timeout-action], [precedence], [security], data, data-length, PUSH-flag, URGENT-flag	Request connection at a particular security and precedence to a specified destination and transmit data with the request.
Send	local-connection-name, data, data-length, PUSH-flag, URGENT-flag, [timeout], [timeout-action]	Transfer data across named connection.
Allocate	local-connection-name, data-length	Issue incremental allocation for receive data to TCP.
Close	local-connection-name	Close connection gracefully.
Abort	local-connection-name	Close connection abruptly.
Status	local-connection-name	Query connection status.

(Table is on page 512 in textbook)

Table 15.3
TCP Service Response Primitives

Primitive	Parameters	Description
Open ID	local-connection-name, source-port, destination-port*, destination-address*,	Informs TCP user of connection name assigned to pending connection requested in an Open primitive
Open Failure	local-connection-name	Reports failure of an Active Open request
Open Success	local-connection-name	Reports completion of pending Open request
Deliver	local-connection-name, data, data-length, URGENT-flag	Reports arrival of data
Closing	local-connection-name	Reports that remote TCP user has issued a Close and that all data sent by remote user has been delivered
Terminate	local-connection-name, description	Reports that the connection has been terminated; a description of the reason for termination is provided
Status Response	local-connection-name, source-port, source-address, destination-port, destination-address, connection-state, receive-window, send-window, amount-awaiting-ACK, amount-awaiting-receipt, urgent-state, precedence, security, timeout	Reports current status of connection
Error	local-connection-name, description	Reports service-request or internal error

* = Not used for Unspecified Passive Open

(Table is on page 513 in textbook)

Table 15.4

TCP Service Parameters

Source Port	Local TCP user
Timeout	Longest delay allowed for data delivery before automatic connection termination or error report; user specified
Timeout-action	Indicates whether the connection is terminated or an error is reported to the TCP user in the event of a timeout
Precedence	Precedence level for a connection. Takes on values zero (lowest) through seven (highest); same parameter as defined for IP
Security-range	Allowed ranges in compartment, handling restrictions, transmission control codes, and security levels
Destination Port	Remote TCP user
Destination Address	Internet address of remote host
Security	Security information for a connection, including security level, compartment, handling restrictions, and transmission control code; same parameter as defined for IP
Data	Block of data sent by TCP user or delivered to a TCP user
Data Length	Length of block of data sent or delivered
PUSH flag	If set, indicates that the associated data are to be provided with the data stream push service
URGENT flag	If set, indicates that the associated data are to be provided with the urgent data signaling service
Local Connection Name	Identifier of a connection defined by a (local socket, remote socket) pair; provided by TCP
Description	Supplementary information in a Terminate or Error primitive
Source Address	Internet address of the local host
Connection State	State of referenced connection (CLOSED, ACTIVE OPEN, PASSIVE OPEN, ESTABLISHED, CLOSING)
Receive Window	Amount of data in octets the local TCP entity is willing to receive
Send Window	Amount of data in octets permitted to be sent to remote TCP entity
Amount Awaiting ACK	Amount of previously transmitted data awaiting acknowledgment
Amount Awaiting Receipt	Amount of data in octets buffered at local TCP entity pending receipt by local TCP user
Urgent State	Indicates to the receiving TCP user whether there are urgent data available or whether all urgent data, if any, have been delivered to the user

(Table is on Page 514 in textbook)

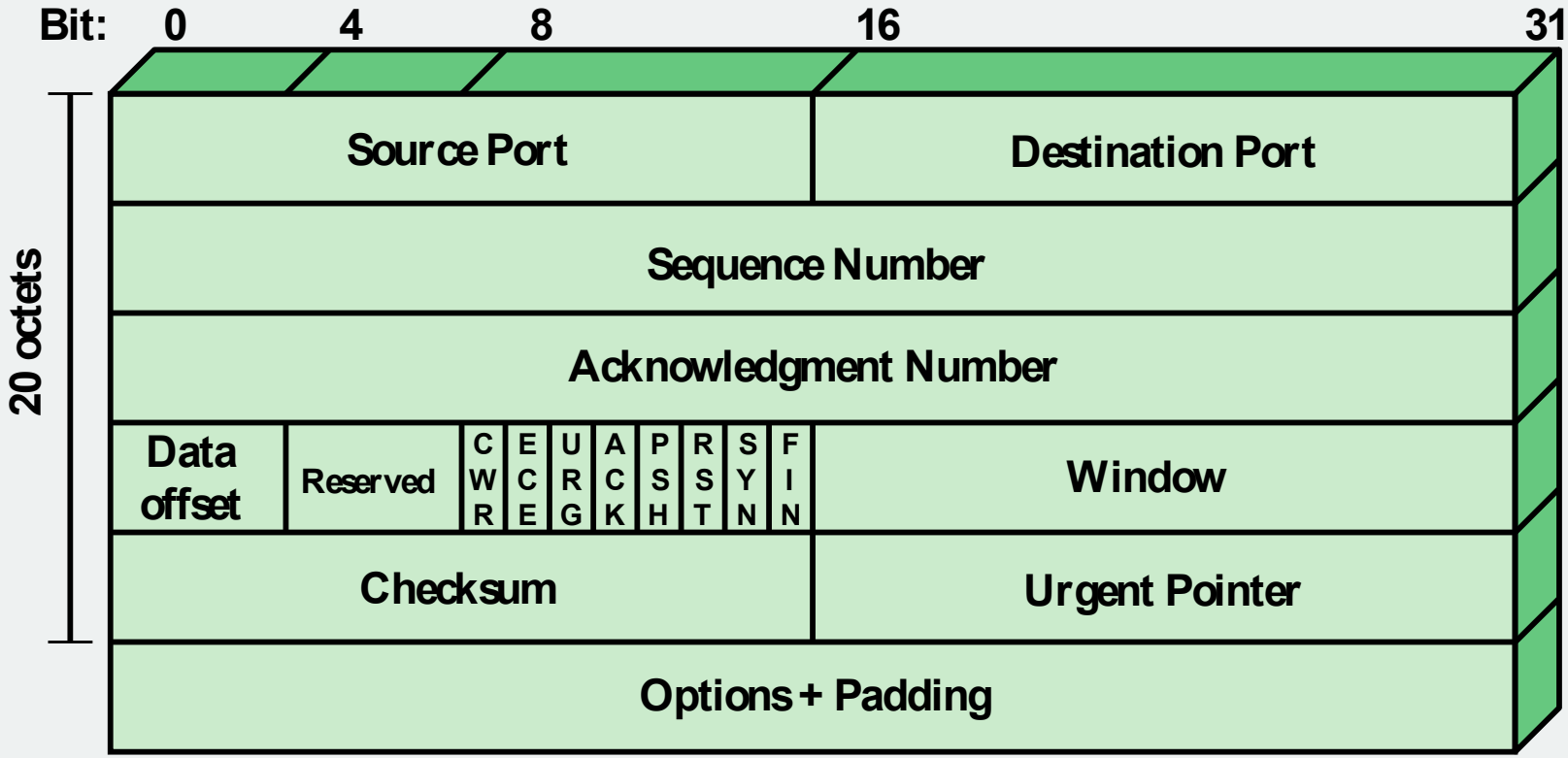


Figure 15.10 TCP Header

TCP Mechanisms

- Can be grouped into:

Connection establishment

- Always uses a three-way handshake
- Connection is determined by host and port

Data transfer

- Viewed logically as consisting of a stream of octets
- Flow control is exercised using credit allocation

Connection termination

- Each TCP user must issue a CLOSE primitive
- An abrupt termination occurs if the user issues an ABORT primitive

TCP Implementation Policy Options

- Implementation opportunities:

Send policy

Deliver policy

Accept policy

Retransmit policy

Acknowledge policy

Send Policy

- In the absence of both pushed data and a closed transmission window a sending TCP entity is free to transmit data at its own convenience
- TCP may construct a segment for each batch of data provided or it may wait until a certain amount of data accumulates before constructing and sending a segment
- Infrequent and large transmissions have low overhead in terms of segment generation and processing
- If transmissions are frequent and small, the system is providing quick response

Deliver Policy

- In the absence of a Push, a receiving TCP entity is free to deliver data to the user at its own convenience
- May deliver as each in-order segment is received, or may buffer data before delivery
- If deliveries are infrequent and large, the user is not receiving data as promptly as may be desirable
- If deliveries are frequent and small, there may be unnecessary processing, as well as operating system interrupts

Accept Policy

- If segments arrive out of order the receiving TCP entity has two options:

In-order

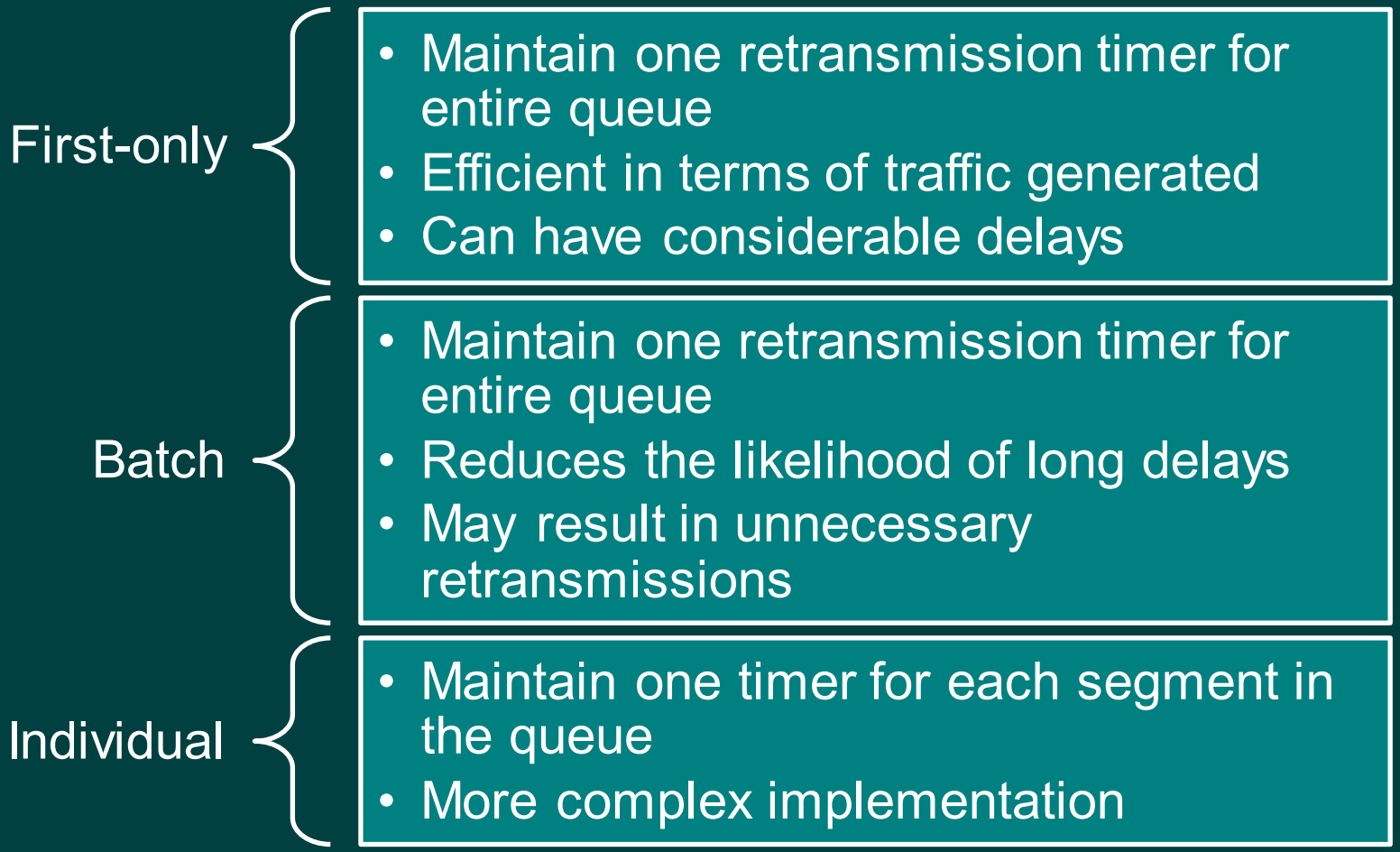
- Accepts only segments that arrive in order; any segment that arrives out of order is discarded
- Makes for simple implementation but places a burden on the networking facility
- If a single segment is lost in transit, then all subsequent segments must be retransmitted

In-window

- Accepts all segments that are within the receive window
- Requires a more complex acceptance test and a more sophisticated data storage scheme

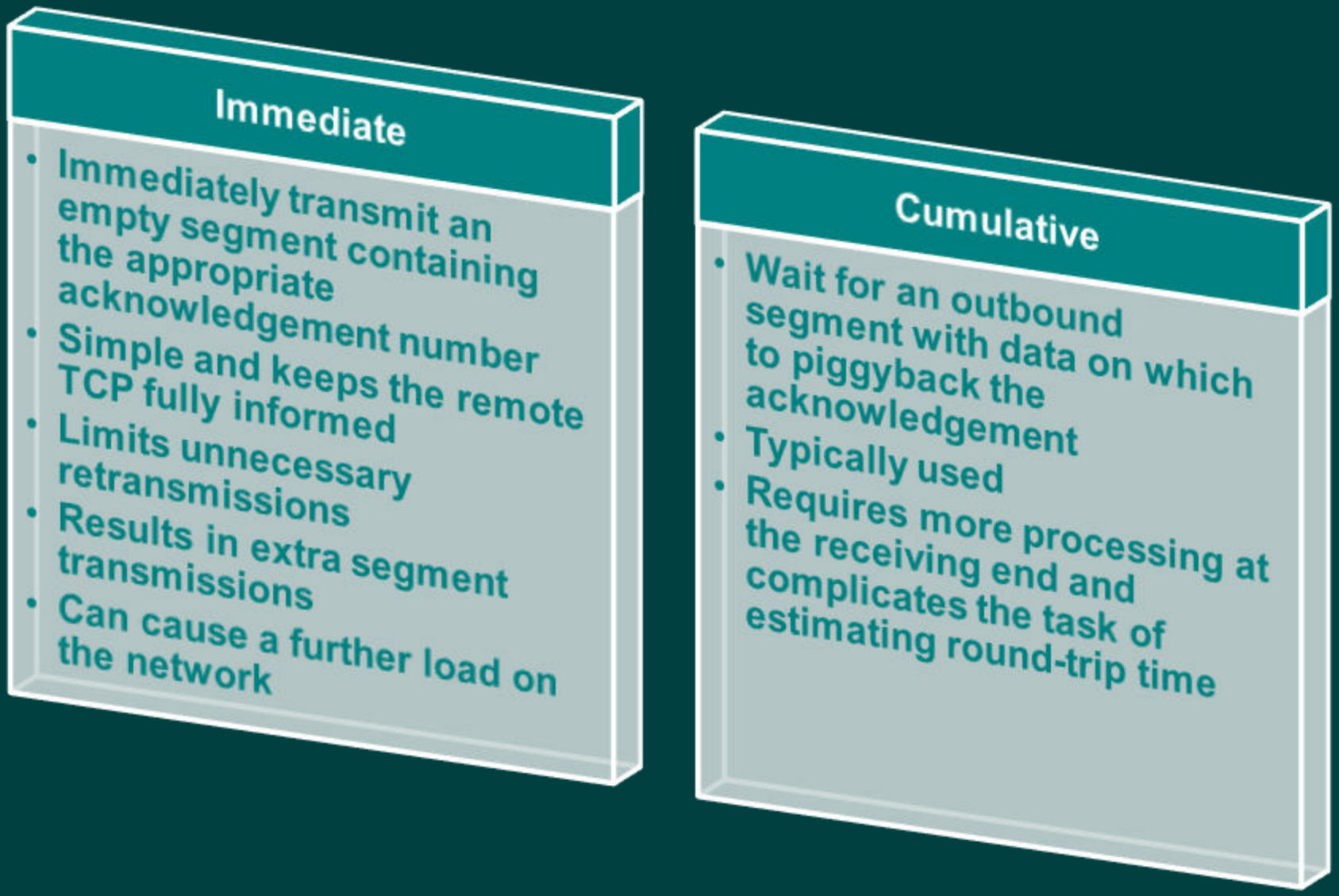
Retransmit Policy

- Retransmission strategies:



Acknowledge Policy

- Timing of acknowledgment:



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

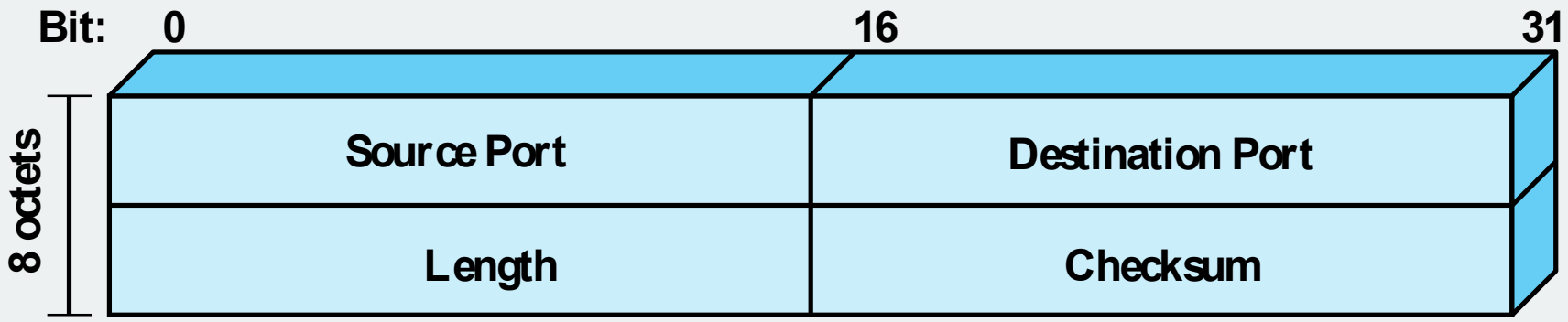


Figure 15.11 UDP Header

Summary

- Connection-oriented transport protocol mechanisms
  - Reliable sequencing network service
  - Unreliable network service
- UDP
- TCP
  - Services
  - Header format
  - Mechanisms
  - Implementation policy options