Textbook: William Stallings, Data and Computer Communications

Data Communications and Networking

Chapter 17

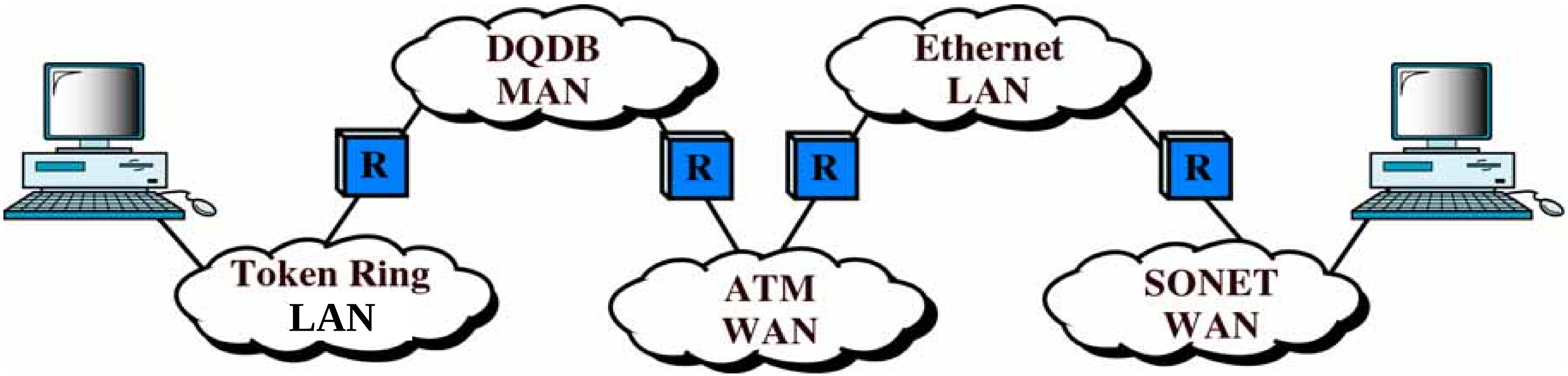
Transport Protocols

http://cs.nju.edu.cn/yangxc
xcy@nju.eu.cn

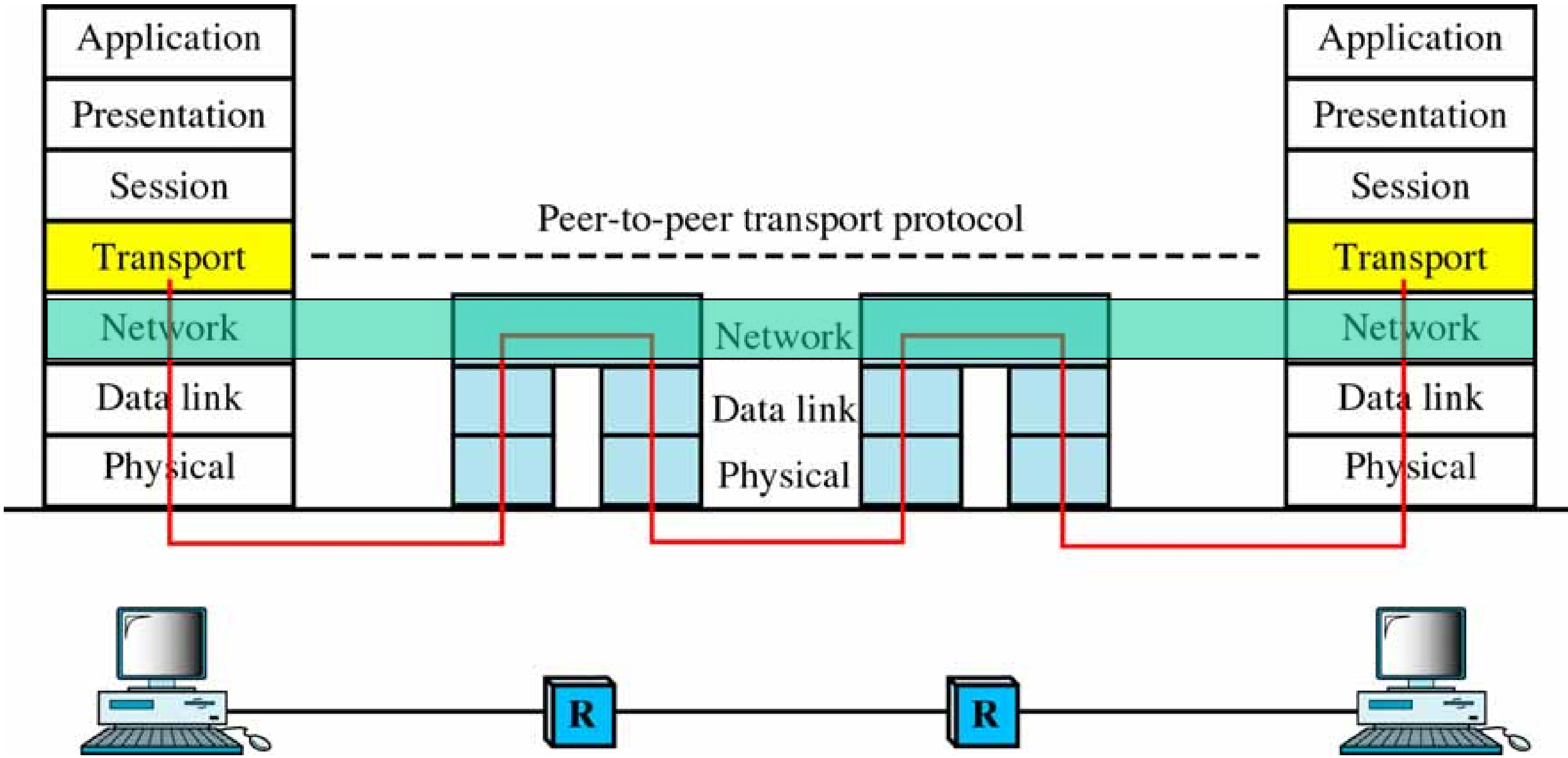
Transport Protocol

- Sits above a network or internetwork, which provides network-related services
- Provides transport services (TS) to upper layer users such as FTP, SMTP, and TELNET
- Uses some lower layer services such as IP protocol to support communication between local and remote transport entities

An Internetwork

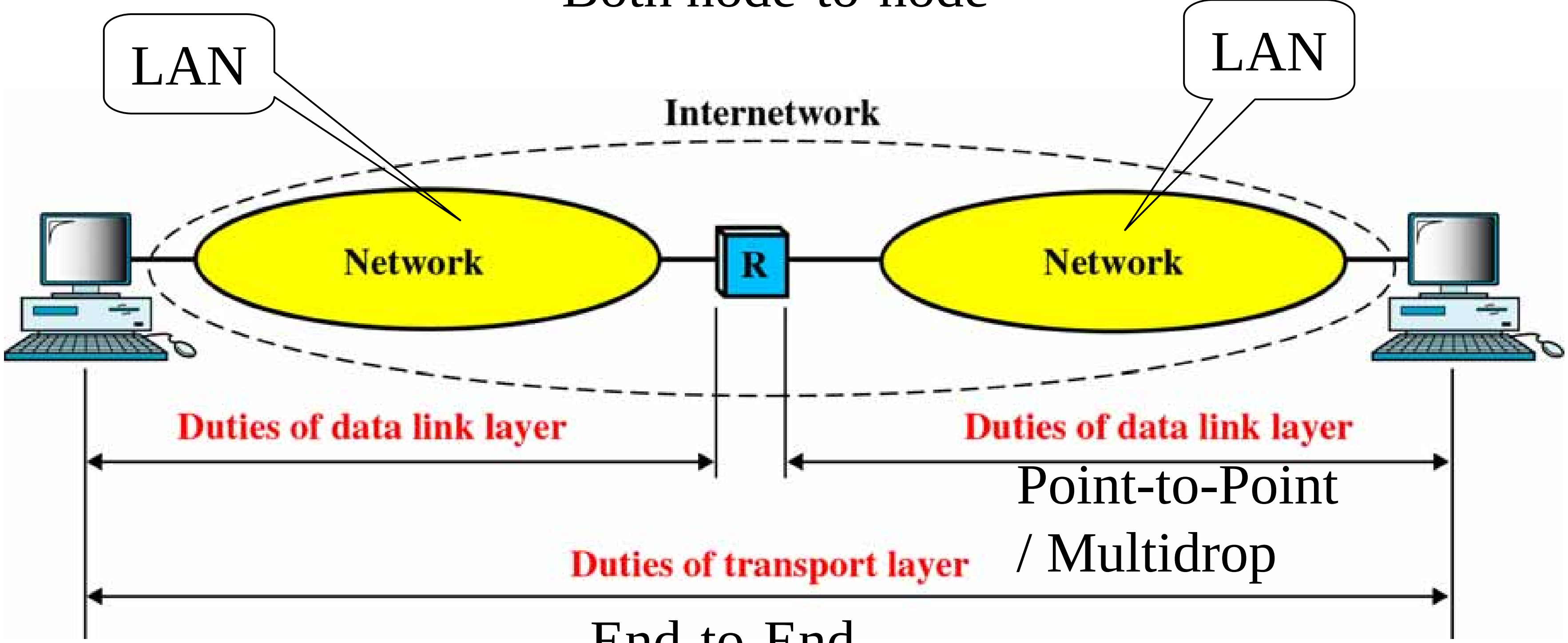


Transport Layer Concept



Transport Layer Compared with Data Link Layer

Both node-to-node



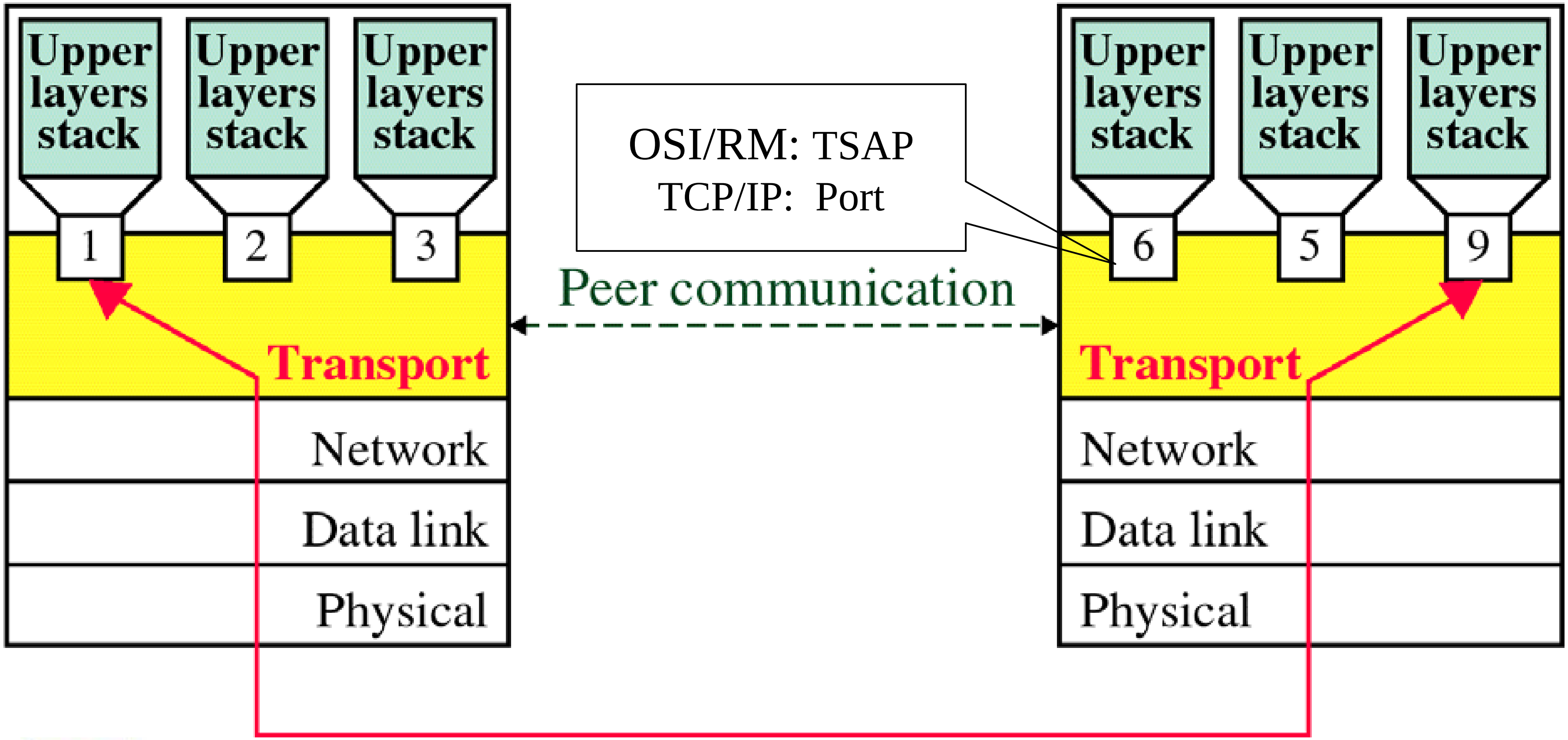
Point-to-Point / Multidrop

End-to-End (host-to-host)

Transport Layer Compared with Network or Internetwork Layer

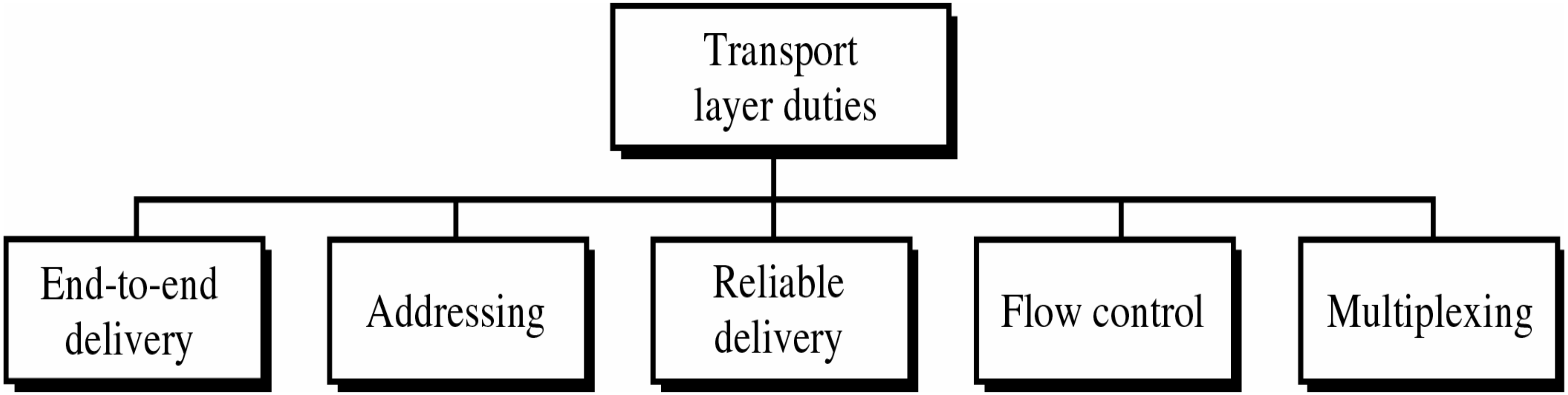
- End-to-end Delivery
  - Network or internetwork layer
    - Oversees end-to-end delivery of individual packets
    - No see any relationship between packets
    - Deliver packets to host
  - Transport layer
    - Make sure entire message (not just a single packet) arrive intact
    - Oversee end-to-end delivery of an entire massage
    - Deliver messages to application processes in host

Service Points



TSAP - Transport Service Access Point

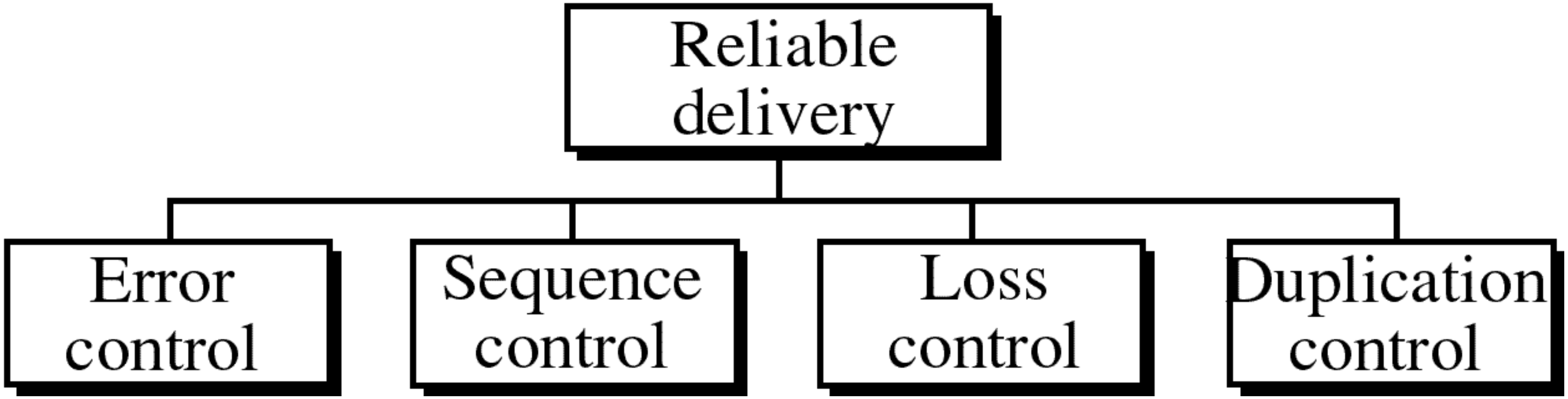
Transport Layer Duties



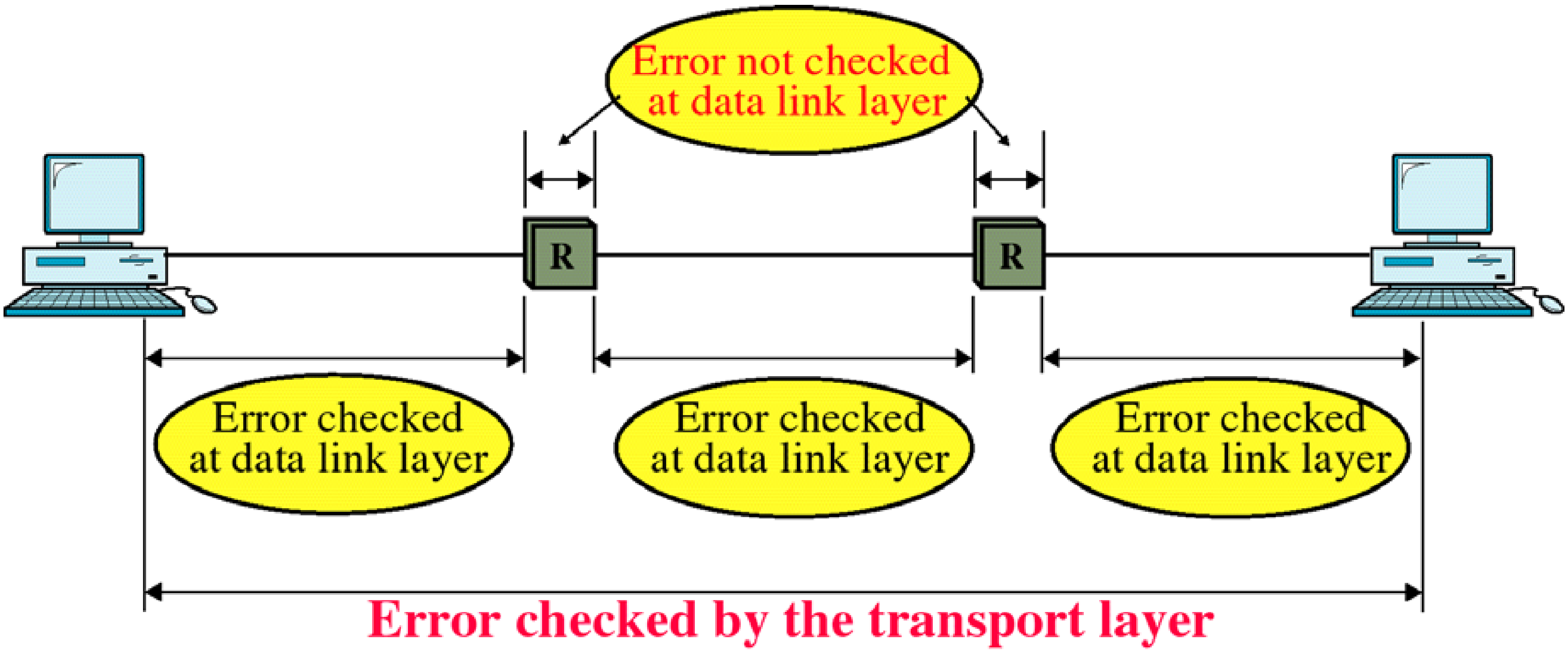
Addressing

- To ensure accurate delivery from service point to service point, another level of addressing will be needed in addition to those at data link and network levels
  - **Data link level protocols** need to know which two computers within a network are communicating
  - **Network level protocols** need to know which two computers within an internetwork are communicating
  - **Transport level protocol** needs to know which upper layer protocols are communicating

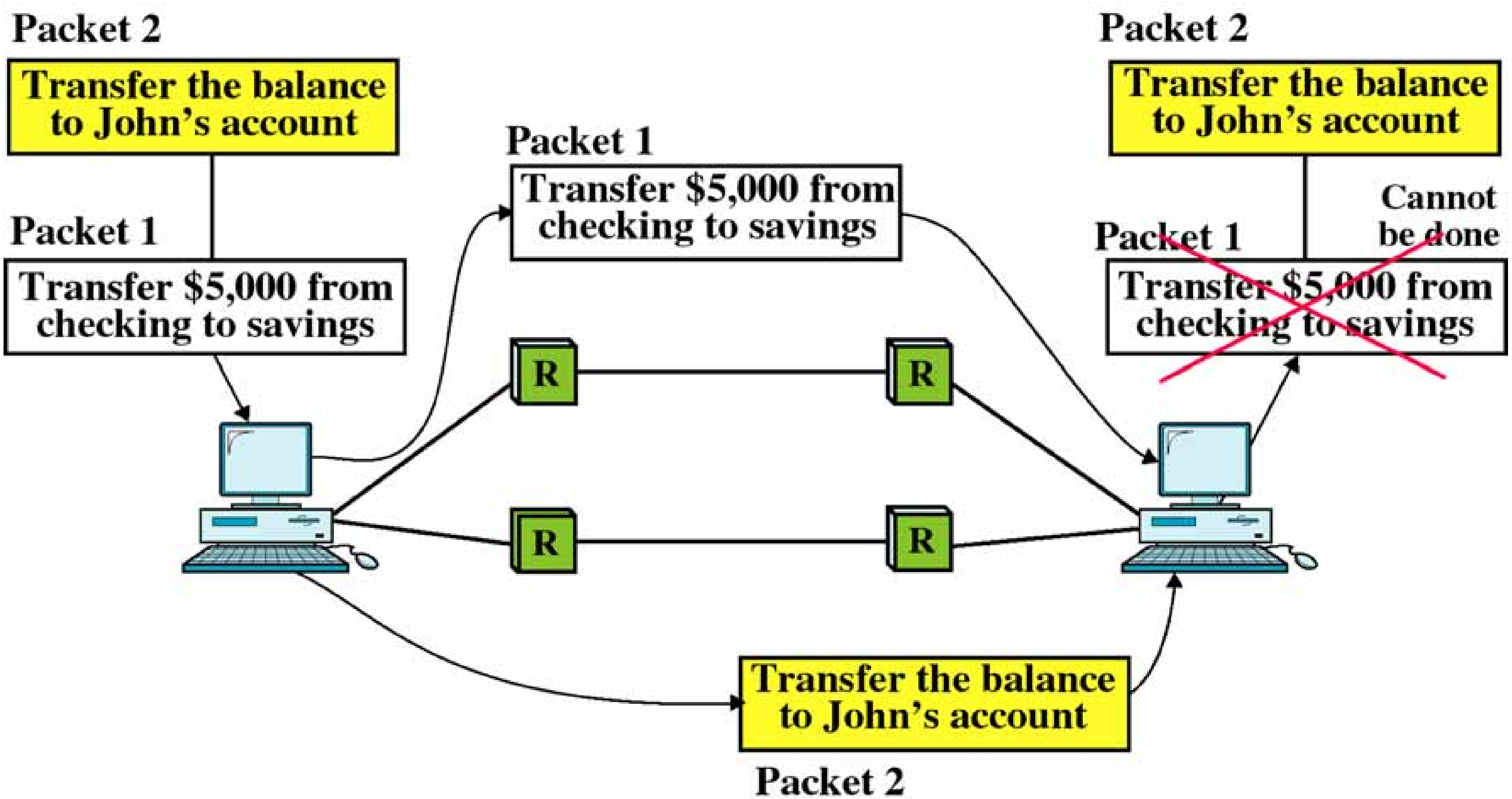
Reliable Delivery

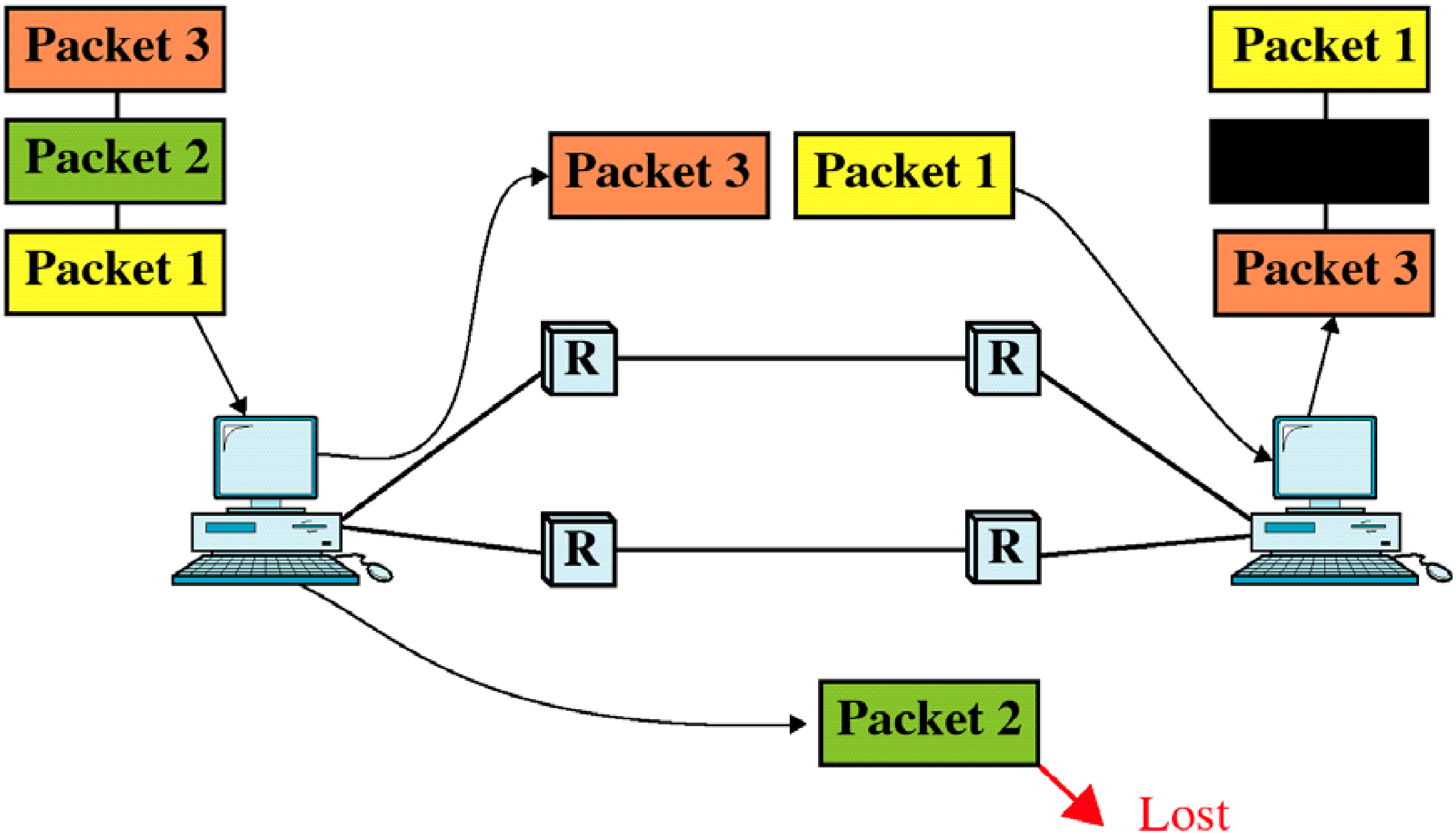


Transport and Data Link Error Control

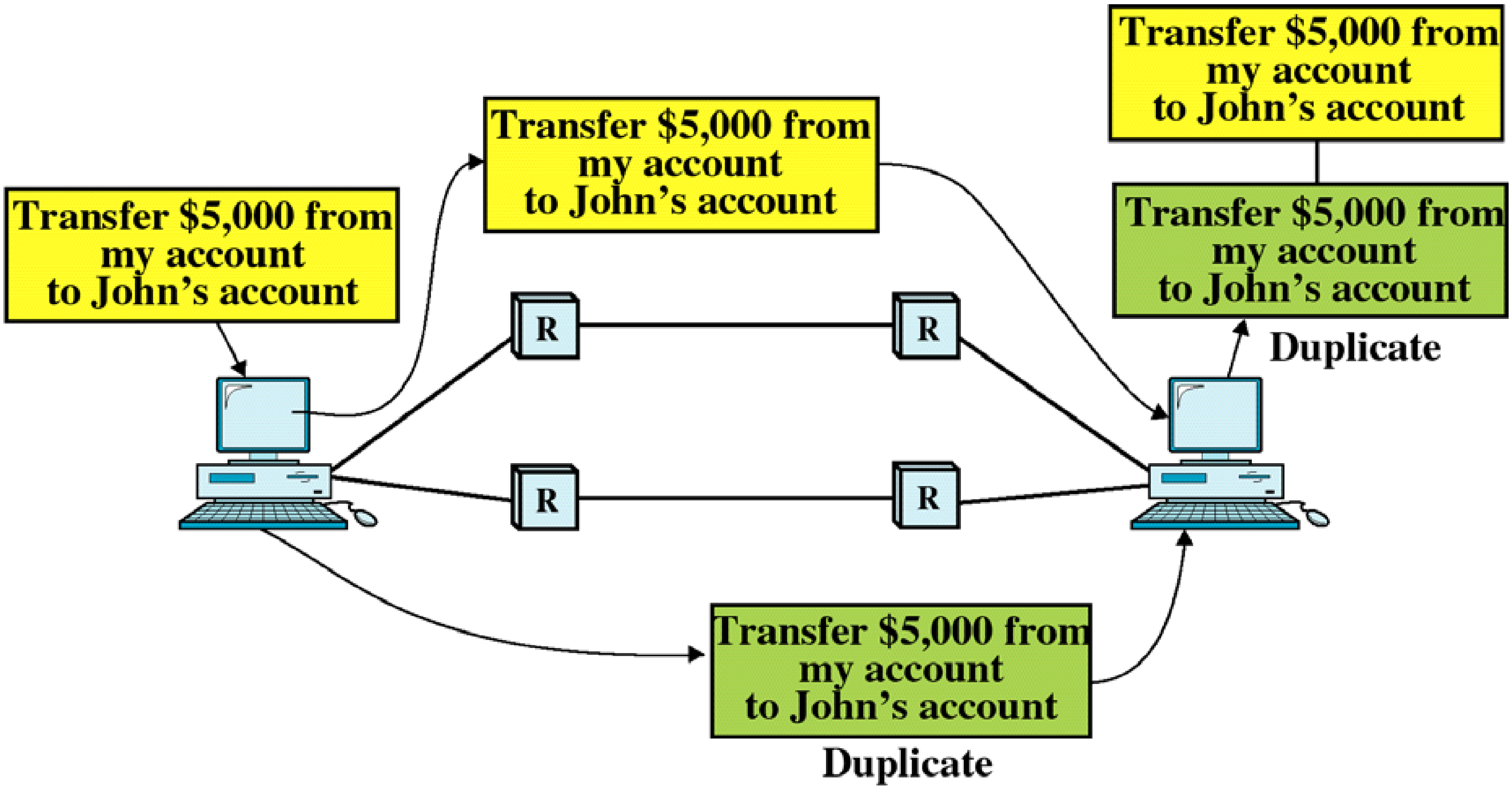


Sequence Control

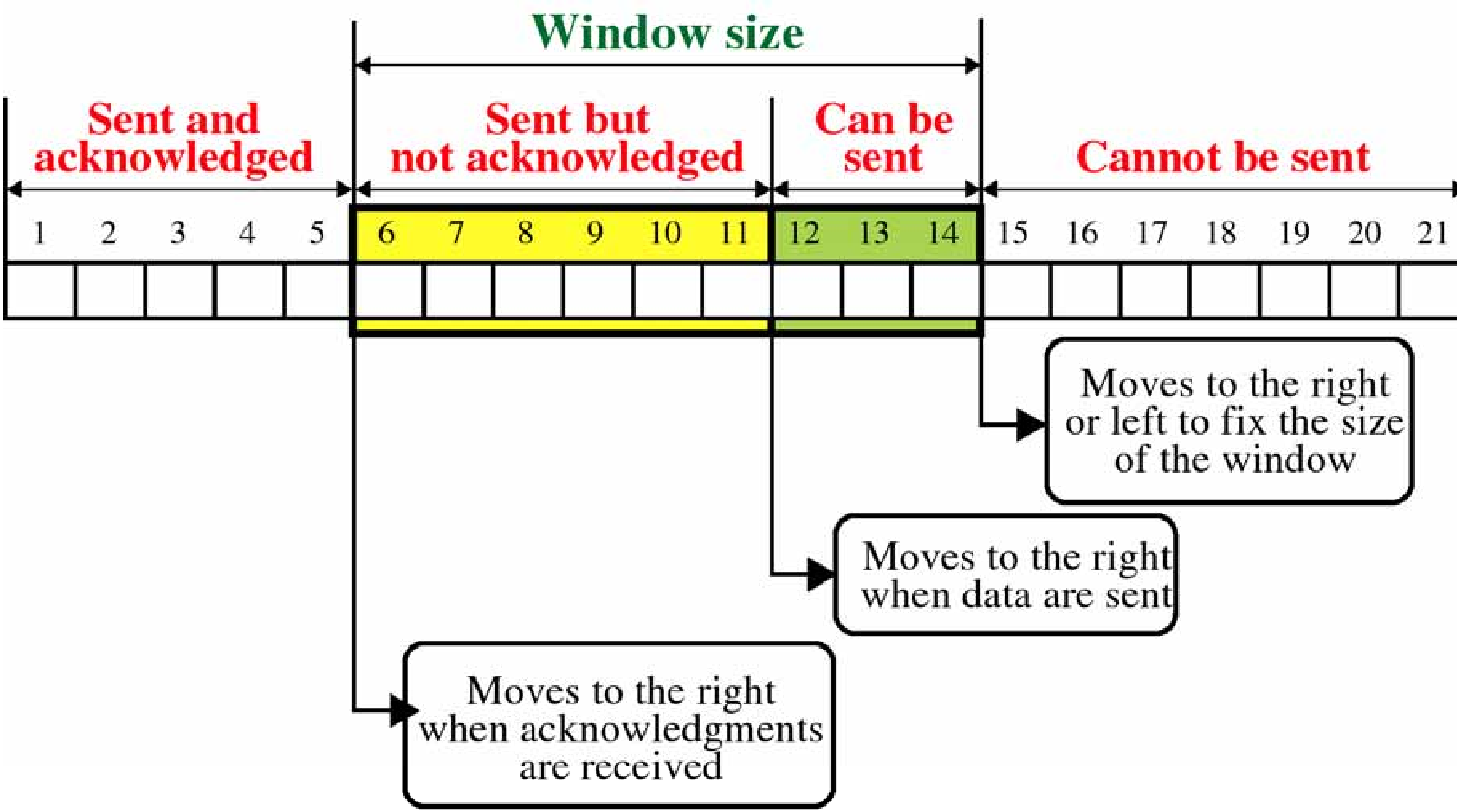




Loss Control

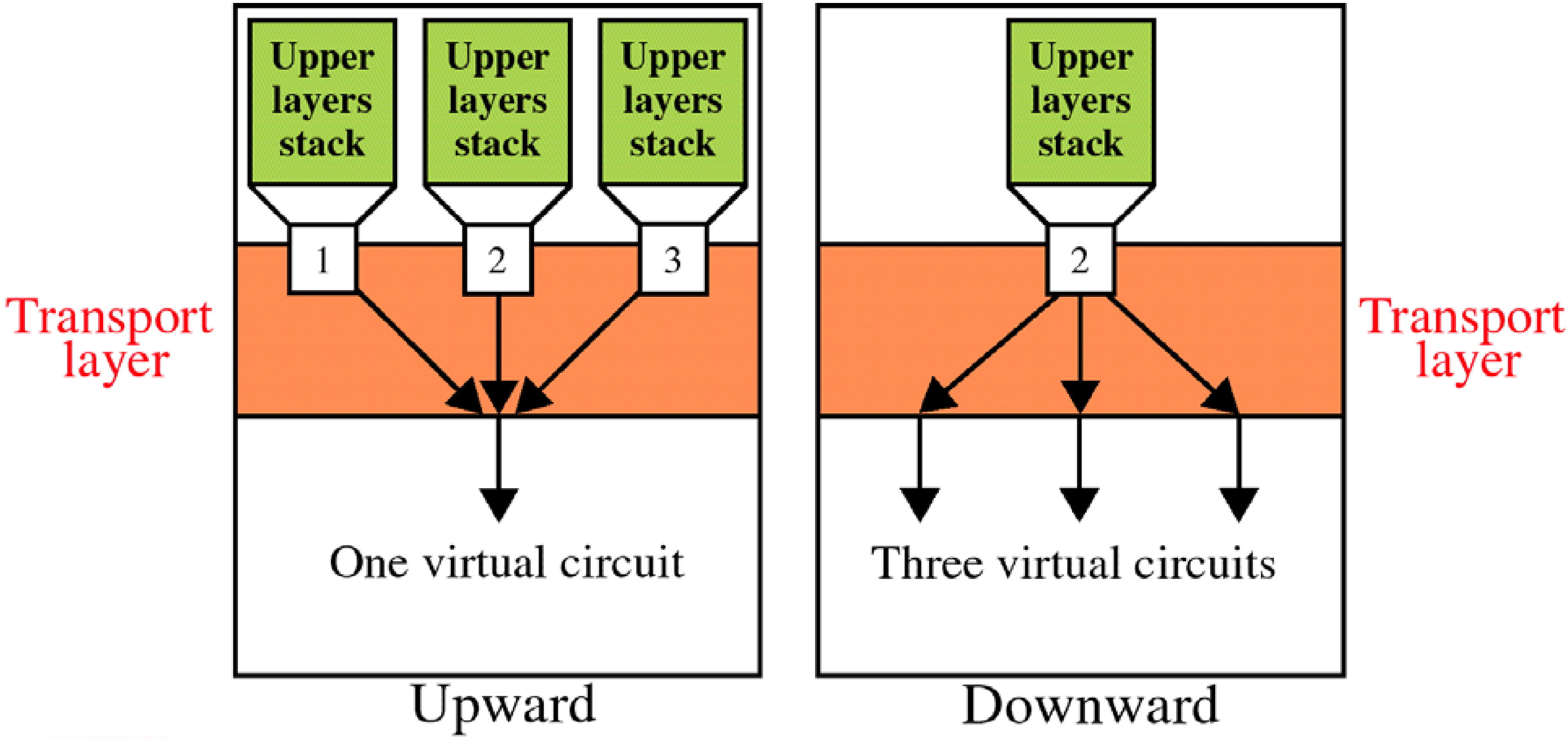


Duplication Control



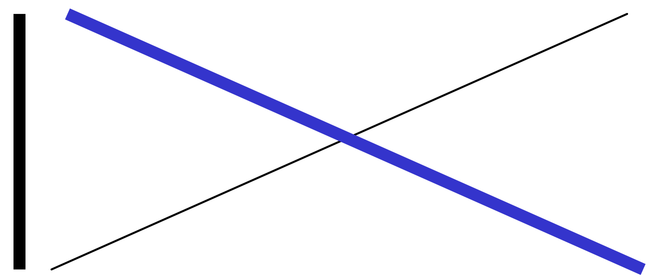
Flow Control

Multiplexing



Services of Transport Layer and Network Layer

Layer 4 (TS) Connection-oriented Conectionless



Layer 3 (NS) Reliable service Unreliable service

Connection Oriented Transport Protocol Mechanisms

- Logical connection
  - Establishment
  - Maintenance
  - Termination
- Reliable
- e.g. TCP

Reliable Sequencing Network Service

- Assume arbitrary length message
- Assume virtually 100% reliable delivery by network service
- Examples
  - reliable packet switched network using X.25
  - frame relay using LAPF control protocol
  - IEEE 802.3 using connection oriented LLC service
- Transport service is end to end protocol between two systems on same network

Issues in a Simple Transport Protocol

- Addressing
- Multiplexing
- Flow Control
- Connection establishment and termination

Addressing

- Target user specified by:
  - **User identification**
    - Usually (host, port)
      - Called a socket in TCP
    - Port represents a particular transport service (TS) user

    48-bit socket =
    32-bit IP address + 16-bit port number

  - **Transport entity identification**
    - Generally only one per host
    - If more than one, then usually one of each type
      - Specify transport protocol (TCP, UDP)
  - **Host address**
    - An attached network device
    - In an internet, a global internet address
  - **Network number**

Finding Addresses

- Four methods (2 static and 2 dynamic)
  - Know address ahead of time
    - e.g. collection of network device statistics
  - Well known addresses
    - e.g. Time sharing and word procssing
  - Name server
  - Sending process request to well known address

Multiplexing

- Multiple users employ same transport protocol
- User identified by port number or service access point (SAP)
- Upward and downward multiplexing
- May also multiplex with respect to network services used
  - e.g. multiplexing a single virtual X.25 circuit to a number of transport service user
    - X.25 charges per virtual circuit connection time

Flow Control

- Longer transmission delay between transport entities compared with actual transmission time
  - Delay in communication of flow control info
- Variable transmission delay
  - Difficult to use timeouts
- Flow may be controlled because:
  - The receiving user can not keep up
  - The receiving transport entity can not keep up
- Results in buffer filling up

Coping with Flow Control Requirements (1)

- Do nothing
  - Segments that overflow are discarded
  - Sending transport entity will fail to get ACK and will retransmit
    - Thus further adding to incoming data
- Refuse further segments
  - Backpressure
  - Clumsy
  - Multiplexed connections are controlled on aggregate flow

Coping with Flow Control Requirements (2)

- Use fixed sliding window protocol
  - See chapter 7 for operational details
  - Works well on reliable network
    - Failure to receive ACK is taken as flow control indication
  - Does not work well on unreliable network
    - Can not distinguish between lost segment and flow control
- Use credit scheme

Credit Scheme

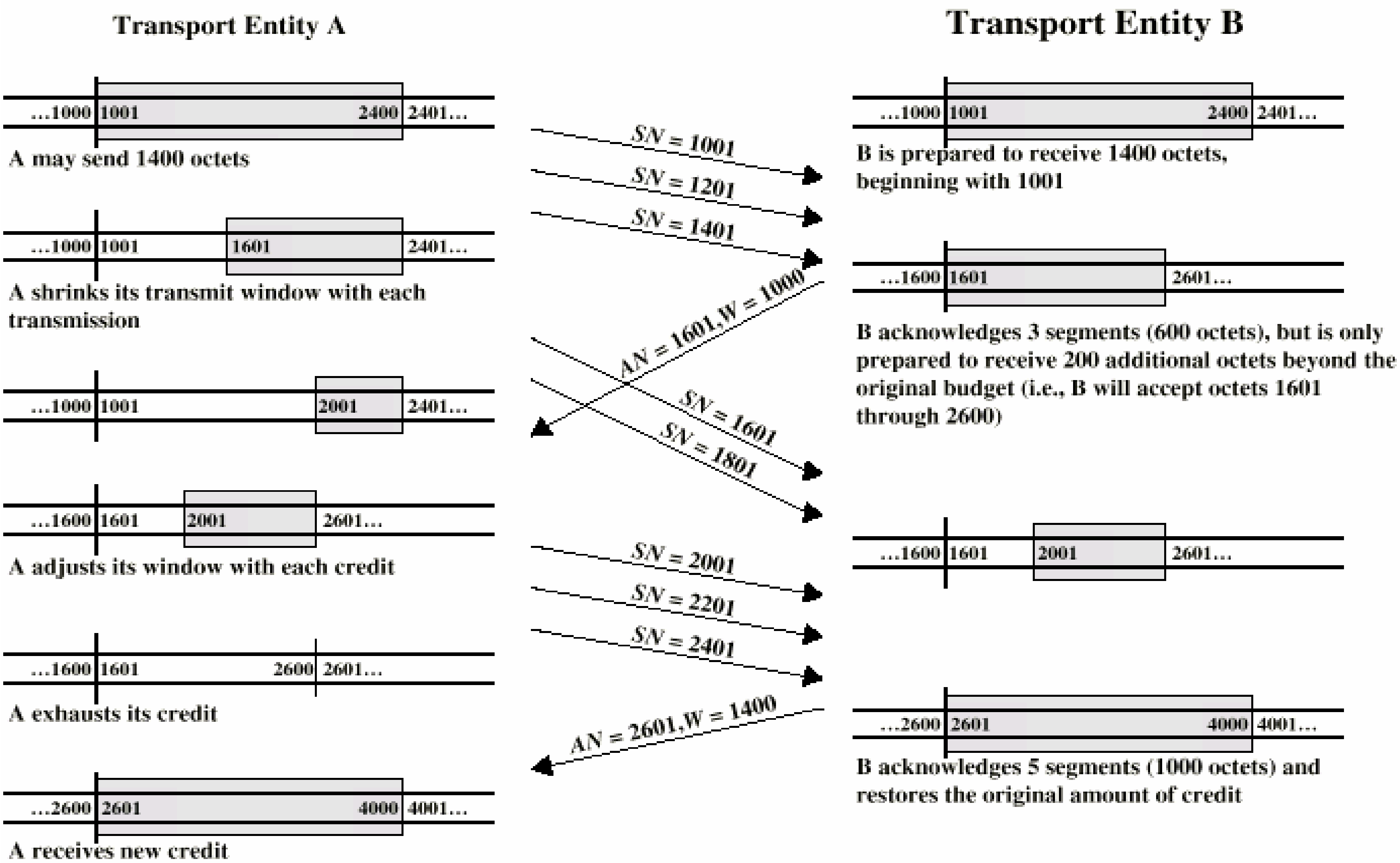
- Greater control on reliable network
- More effective on unreliable network
- Decouples flow control from ACK
  - May ACK without granting credit and vice versa
- Each octet has sequence number
- Each transport segment has seq number, ack number and window size in header

Use of Header Fields

- When sending, seq number is that of first octet in segment
- ACK includes AN=i, W=j
- All octets through SN=i-1 acknowledged
  - Next expected octet is i
- Permission to send additional window of W=j octets
  - i.e. octets through i+j-1

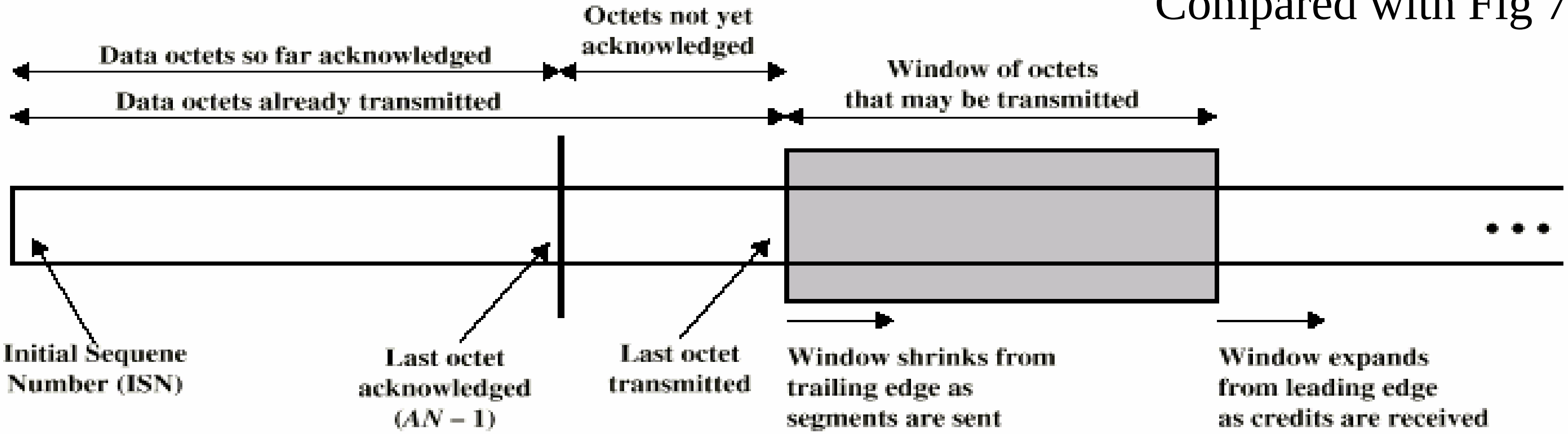
Compared with Fig 7.4

Credit Allocation

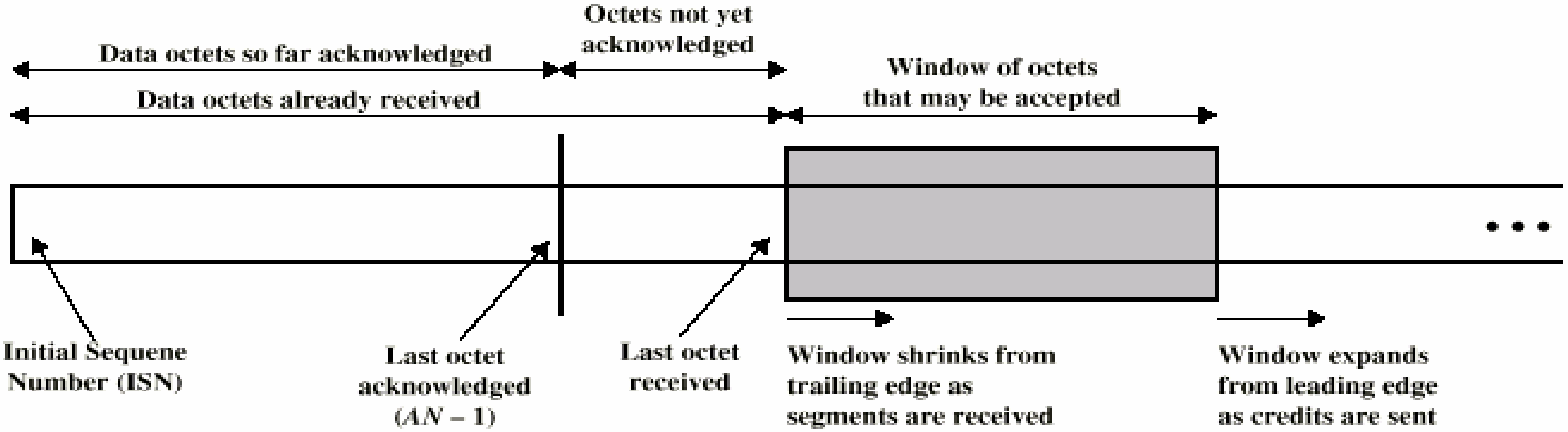


Sending and Receiving Perspectives

Compared with Fig 7.3



(a) Send sequence space

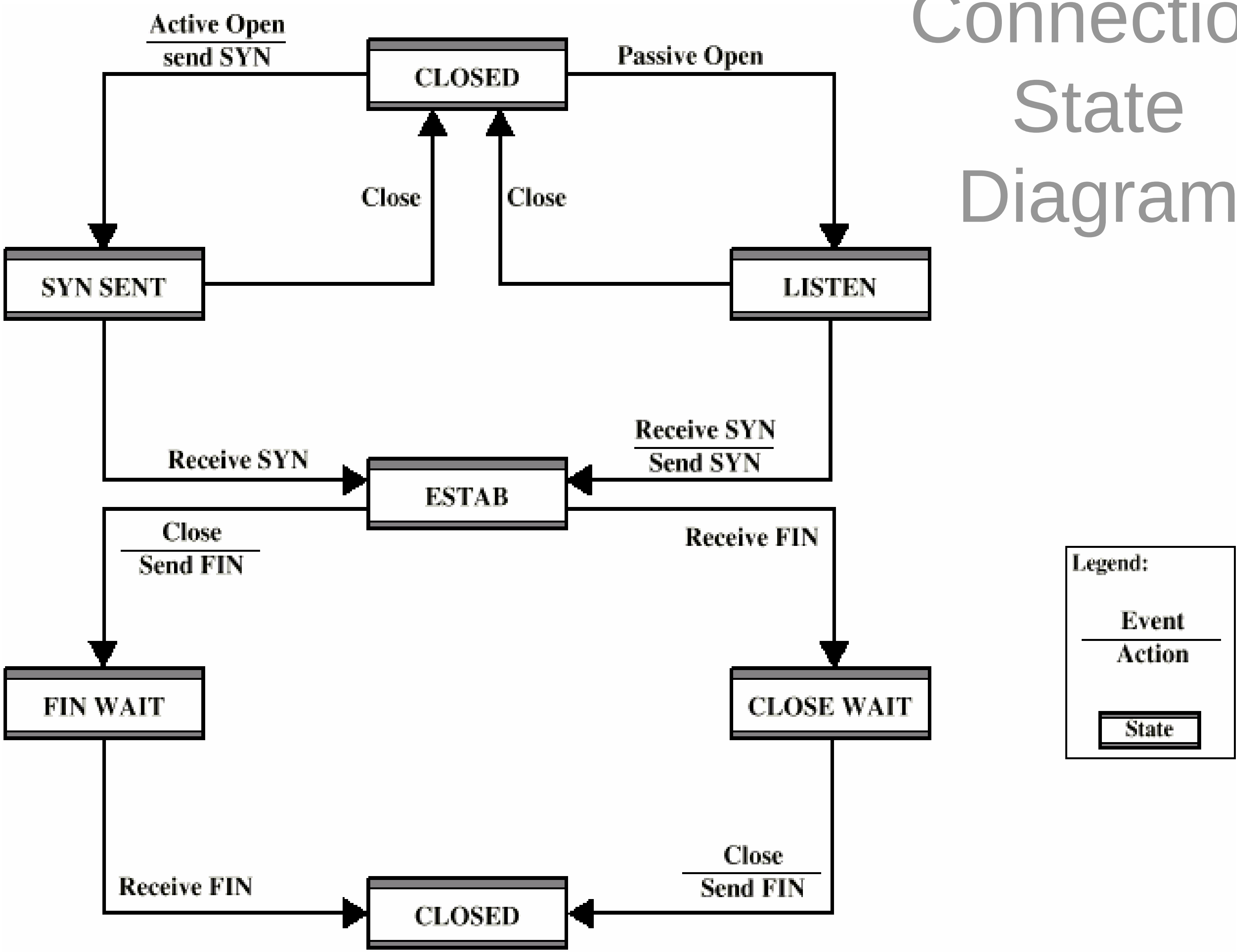


(b) Receive sequence space

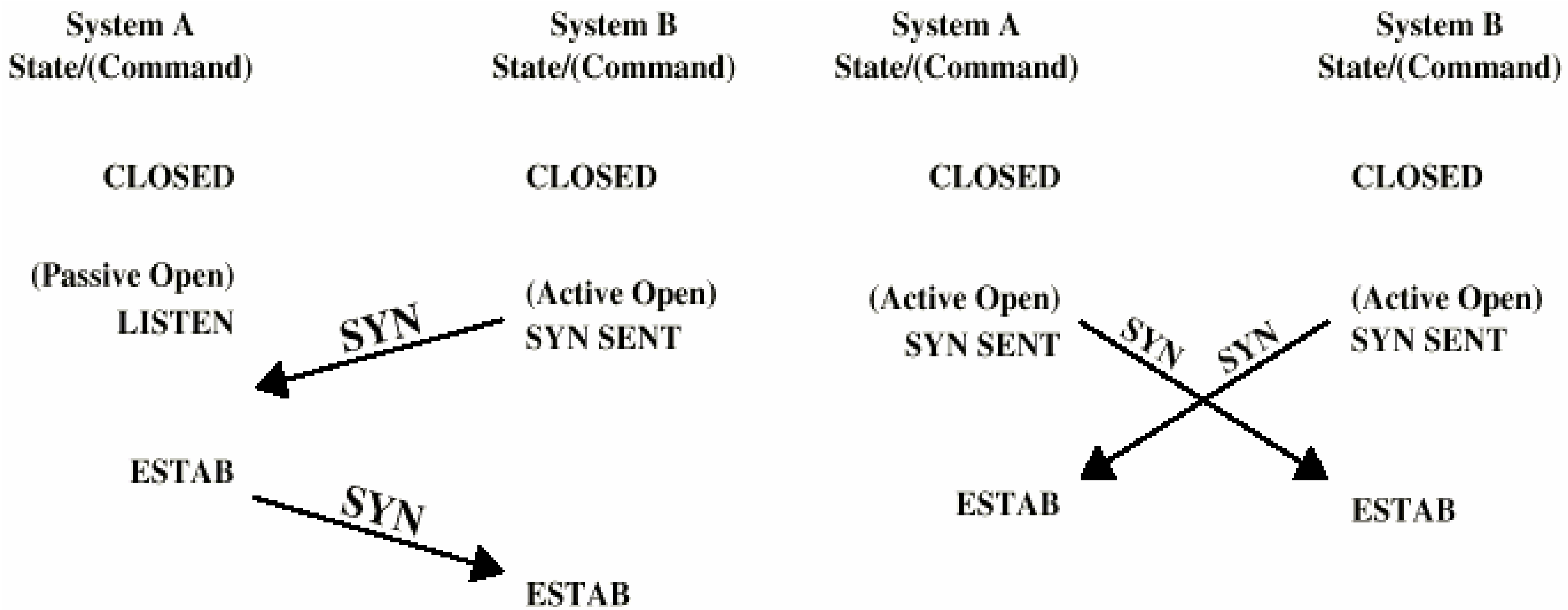
Establishment and Termination

- Allow each end to know the other exists
- Negotiation of optional parameters
  - Max. segment size, Max. window size, QoS
- Triggers allocation of transport entity resources
  - Buffer space, entry in connection table
- By mutual agreement

Connection State Diagram



Connection Establishment



(a) Active/Passive Open

(b) Active/Active Open

Not Listening

- Reject with RST (Reset)
- Queue request until matching open issued
- Signal TS user to notify of pending request
  – May replace passive open with accept

Termination

- Either or both sides
- By mutual agreement
- Abrupt termination
- Or graceful termination
  - Close wait state must accept incoming data until FIN received

Side Initiating Termination

- TS user Close request
- Transport entity sends FIN, requesting termination
- Connection placed in FIN WAIT state
  - Continue to accept data and deliver data to user
  - Not send any more data
- When FIN received, inform user and close connection

Side Not Initiating Termination

- FIN received
- Inform TS user Place connection in CLOSE WAIT state
  - Continue to accept data from TS user and transmit it
- TS user issues CLOSE primitive
- Transport entity sends FIN
- Connection closed

- All outstanding data is transmitted from both sides
- Both sides agree to terminate

Unreliable Network Service

- Examples
  - internet using IP
  - frame relay using LAPF
  - IEEE 802.3 using unacknowledged connectionless LLC
- Segments may get lost
- Segments may arrive out of order

Problems

- Ordered Delivery
- Retransmission strategy
- Duplication detection
- Flow control
- Connection establishment
- Connection termination
- Crash recovery

Ordered Delivery

- Segments may arrive out of order
- Number segments sequentially
- TCP numbers each octet sequentially
- Segments are numbered by the first octet number in the segment

Retransmission Strategy

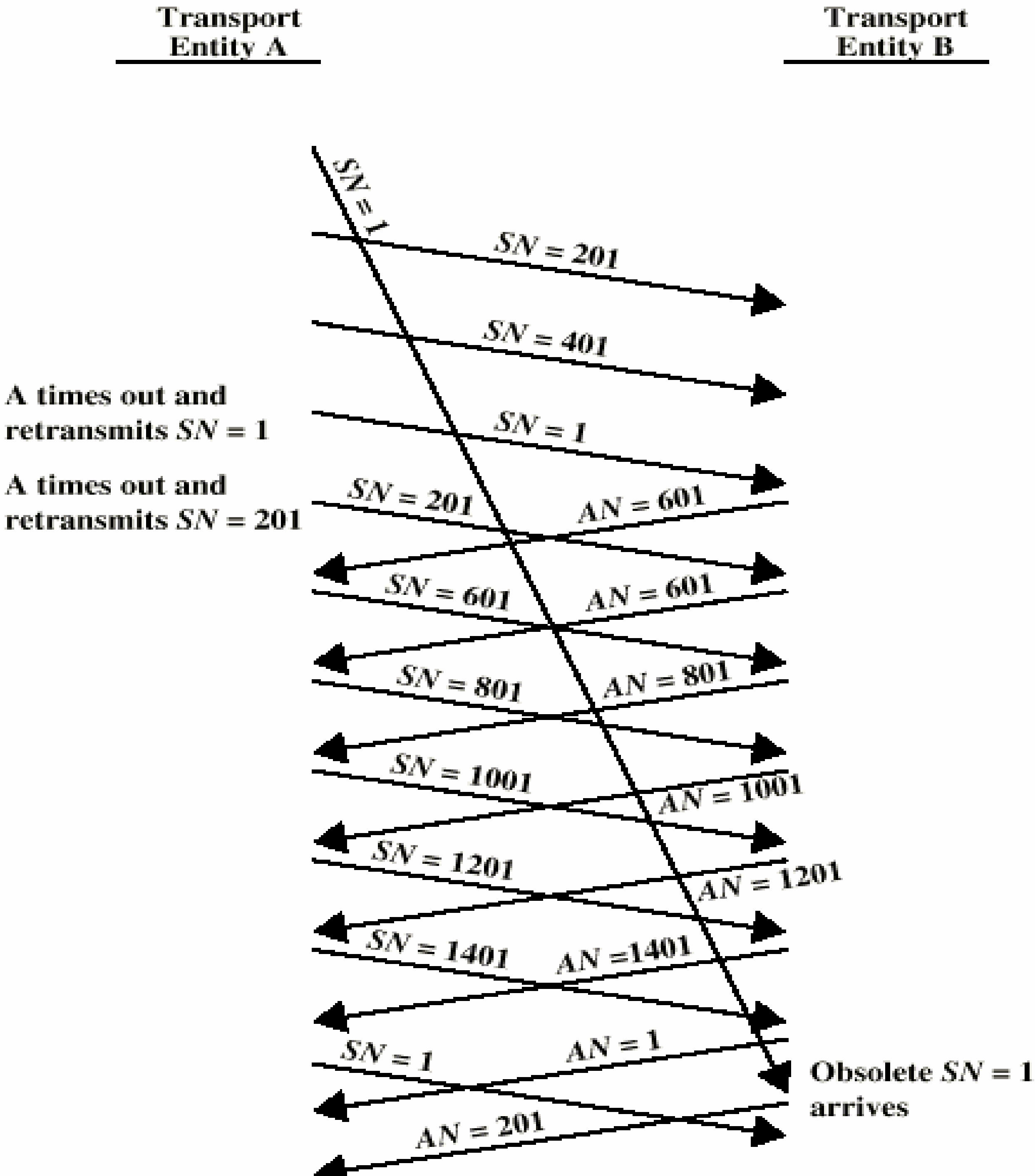
- Segment damaged in transit
- Segment fails to arrive
- Transmitter does not know of failure
- Receiver must acknowledge successful receipt
- Use cumulative acknowledgement
- Time out waiting for ACK triggers re-transmission

Timer Value

- Fixed timer
  - Based on understanding of network behavior
  - Can not adapt to changing network conditions
  - Too small leads to unnecessary re-transmissions
  - Too large and response to lost segments is slow
  - Should be a bit longer than round trip time
- Adaptive scheme
  - May not ACK immediately
  - Can not distinguish between ACK of original segment and re-transmitted segment
  - Conditions may change suddenly

Duplication Detection

- If ACK lost, segment is re-transmitted
- Receiver must recognize duplicates
- Duplicate received prior to closing connection
  - Receiver assumes ACK lost and ACKs duplicate
  - Sender must not get confused with multiple ACKs
  - Sequence number space large enough to not cycle within maximum life of segment
- Duplicate received after closing connection


Transport Entity A
Transport Entity B
SN = 1
SN = 201
SN = 401
A times out and retransmits SN = 1
SN = 1
A times out and retransmits SN = 201
SN = 201
AN = 601
SN = 601
AN = 601
SN = 801
AN = 801
SN = 1001
AN = 1001
SN = 1201
AN = 1201
SN = 1401
AN =1401
SN = 1
AN = 1
Obsolete SN = 1 arrives
AN = 201

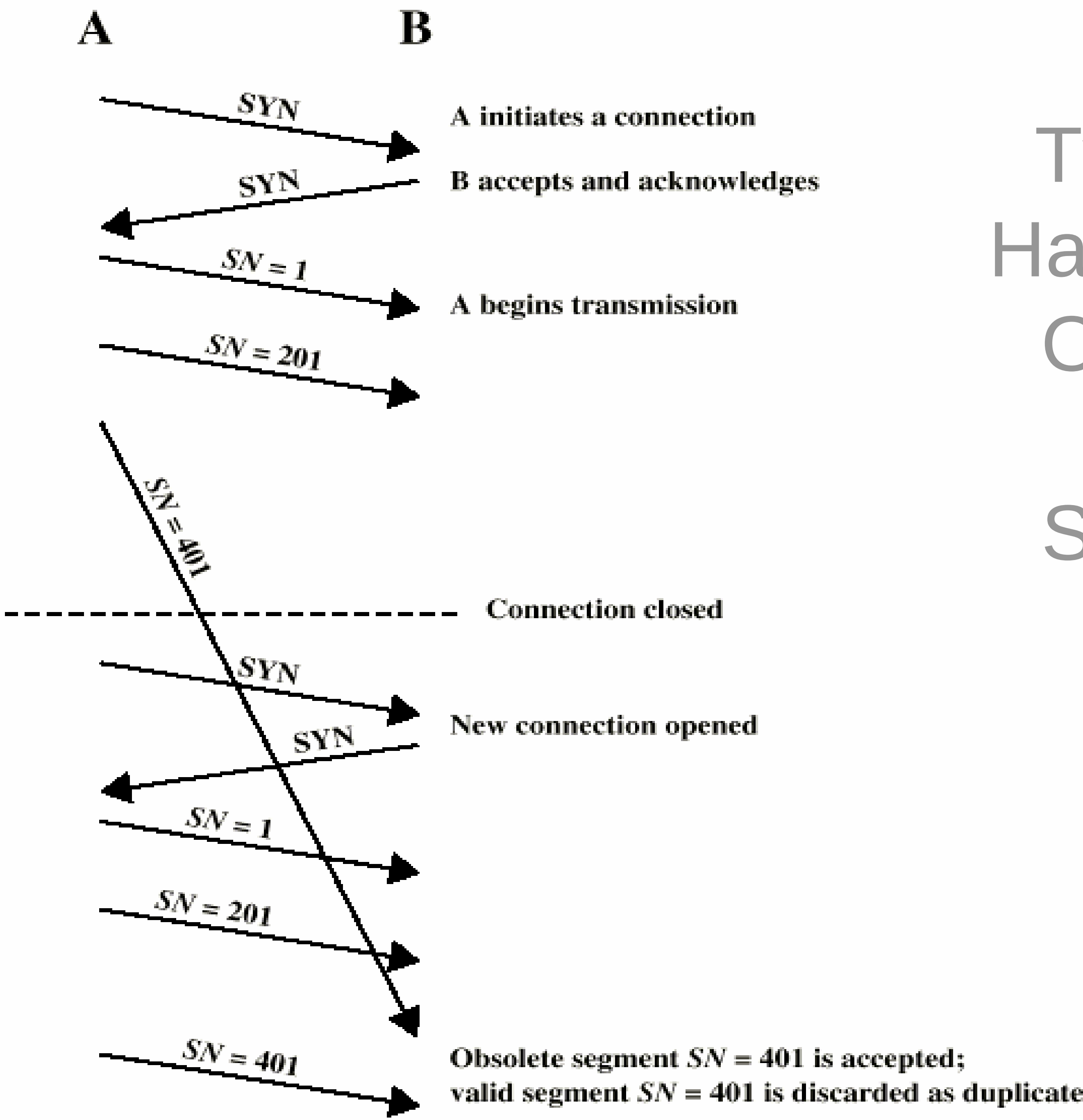
Flow Control

- Credit allocation
- Problem if AN=i, W=0 closing window
- Send AN=i, W=j to reopen, but this is lost
- Sender thinks window is closed, receiver thinks it is open
- Use window timer
- If timer expires, send something
  - Could be re-transmission of previous segment

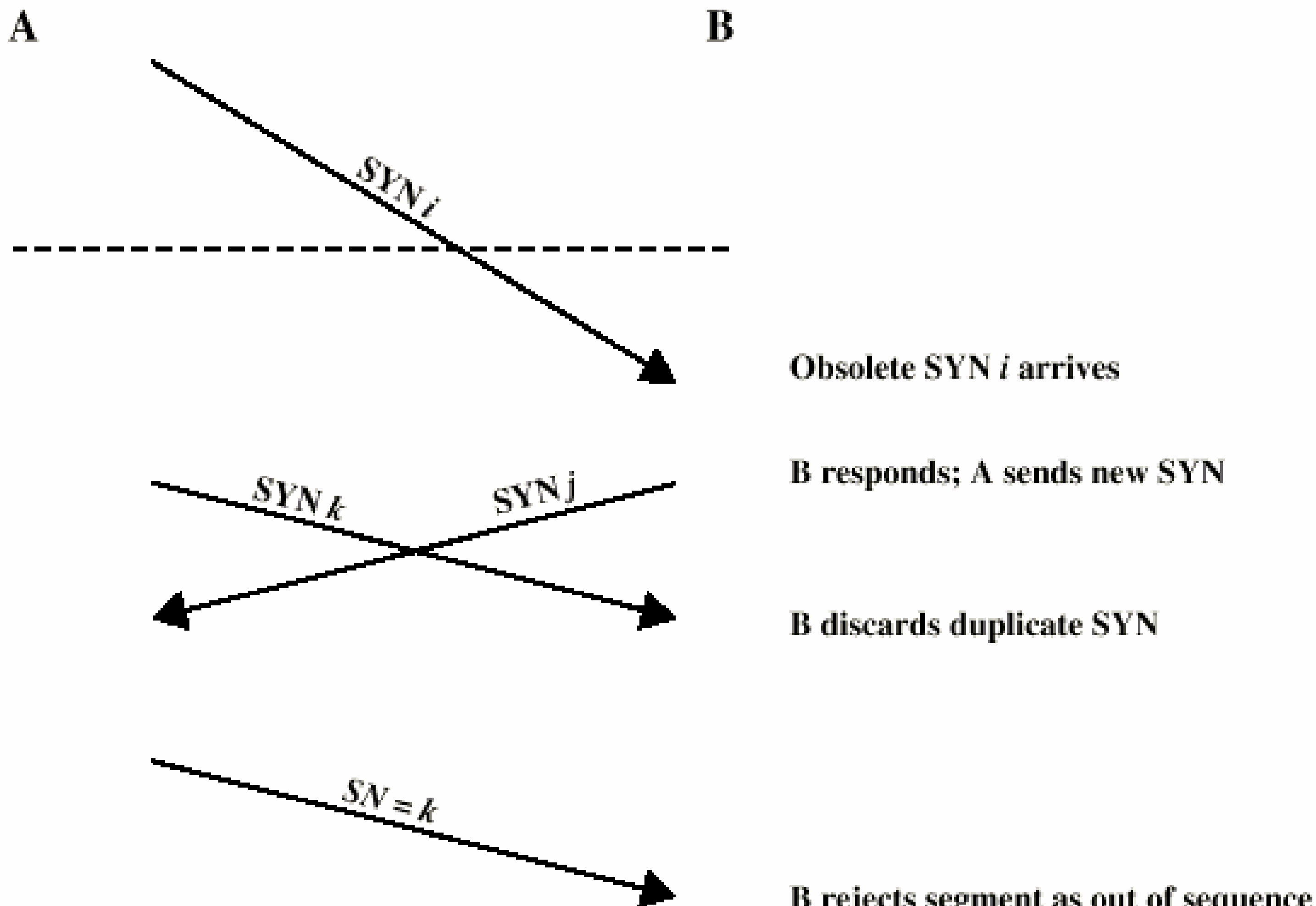
Connection Establishment

- Two way handshake
  - A send SYN, B replies with SYN
  - Lost SYN handled by re-transmission
    - Can lead to duplicate SYNs
  - Ignore duplicate SYNs once connected
- Lost or delayed data segments can cause connection problems
  - Segment from old connections
  - Start segment numbers fare removed from previous connection
    - Use SYN i
    - Need ACK to include i
    - Three Way Handshake

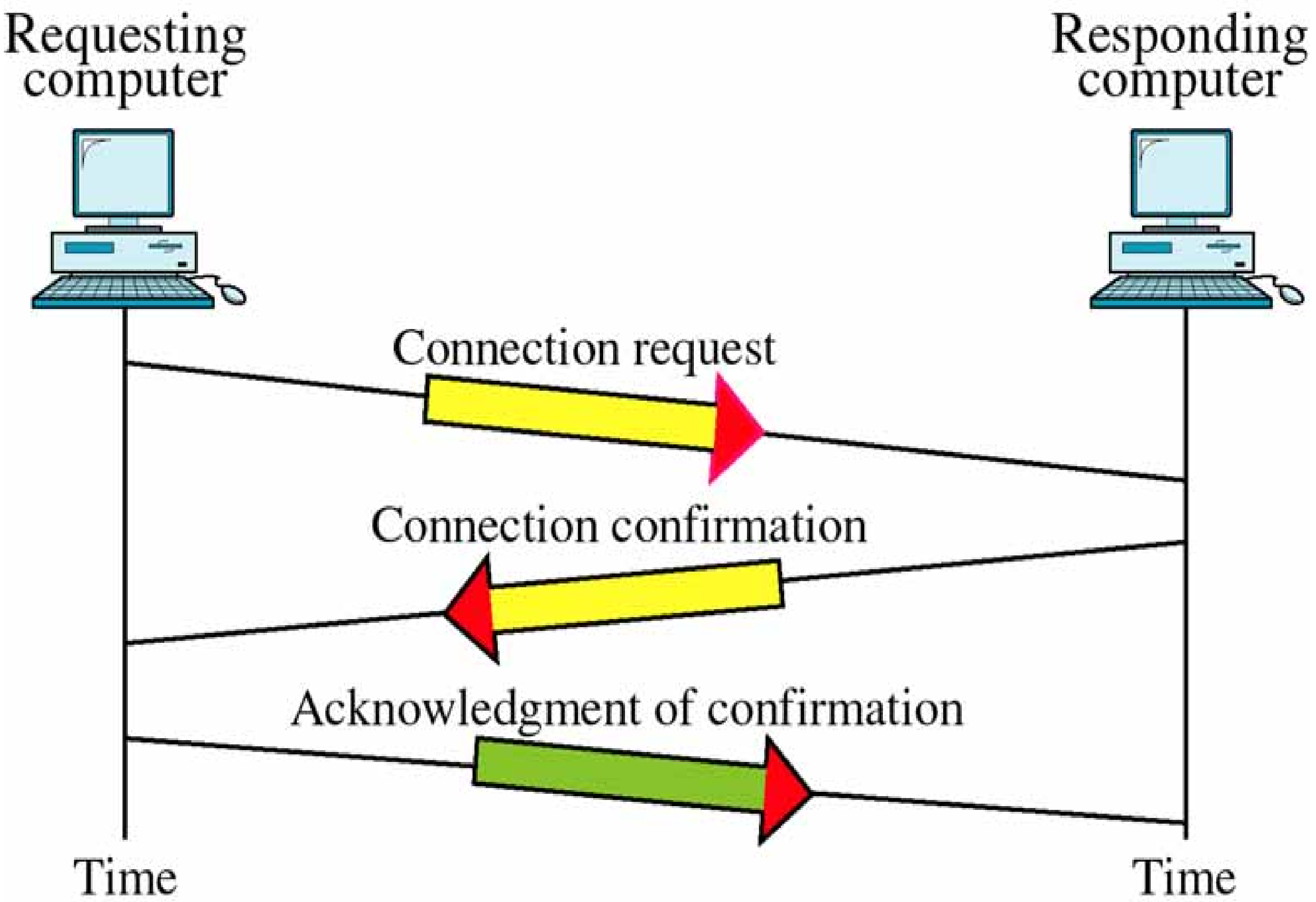
Two Way Handshake: Obsolete Data Segment



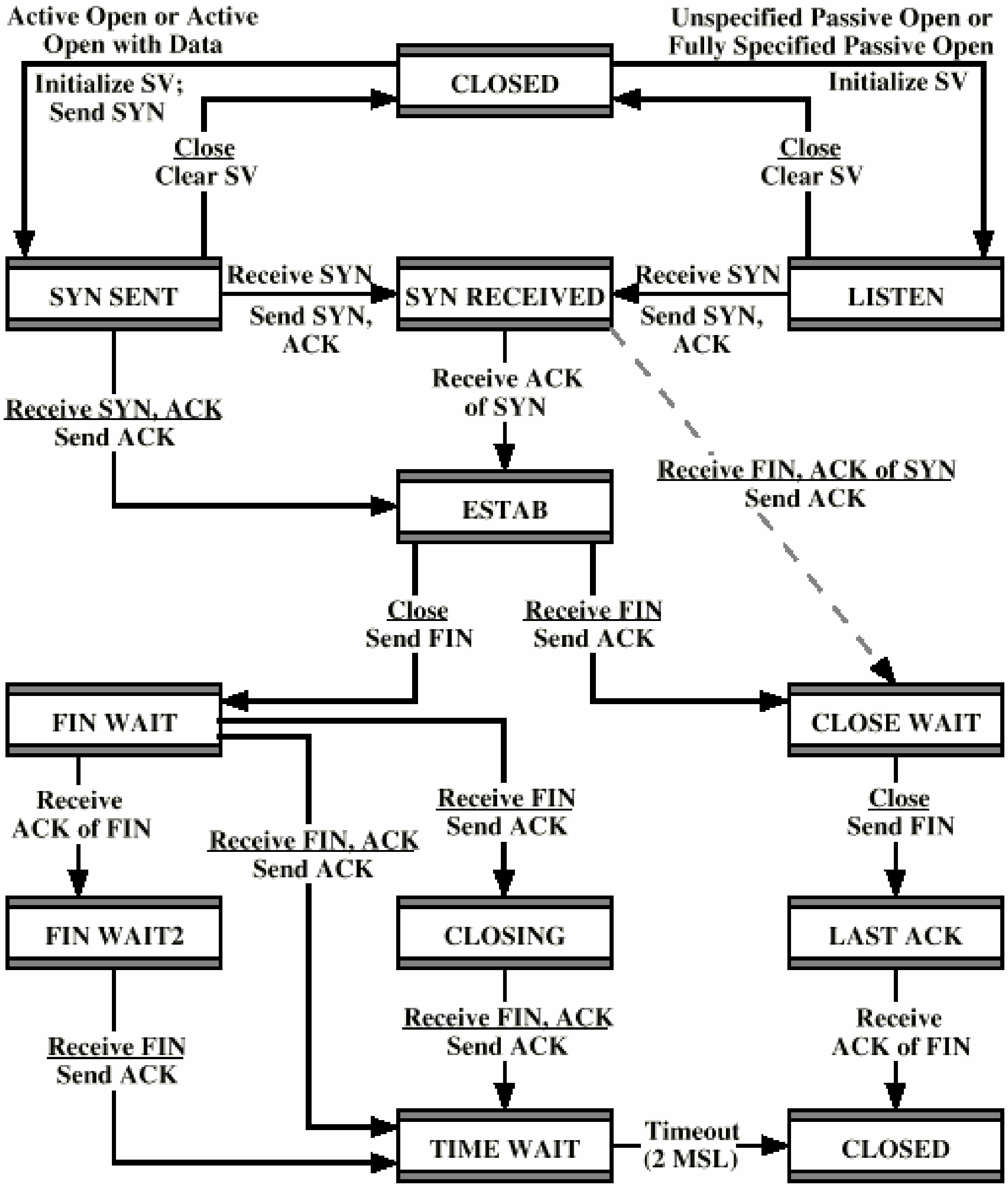
Two Way Handshake: Obsolete SYN Segment



Three Way Handshake: Connection Establishment

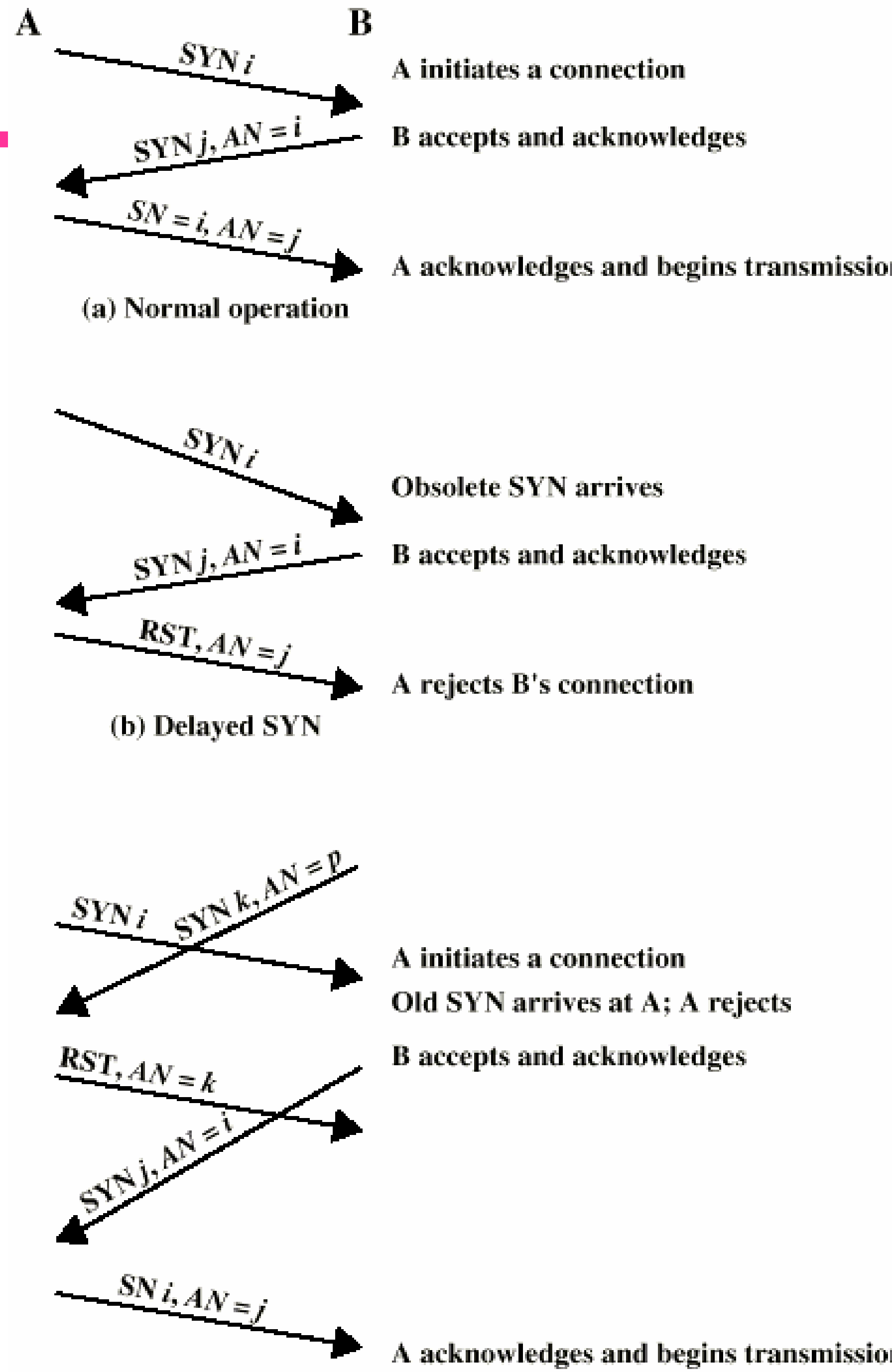


Three Way Handshake: State Diagram



SV = state vector
MSL = maximum segment lifetime

Three Way Handshake: Examples



(a) Normal operation

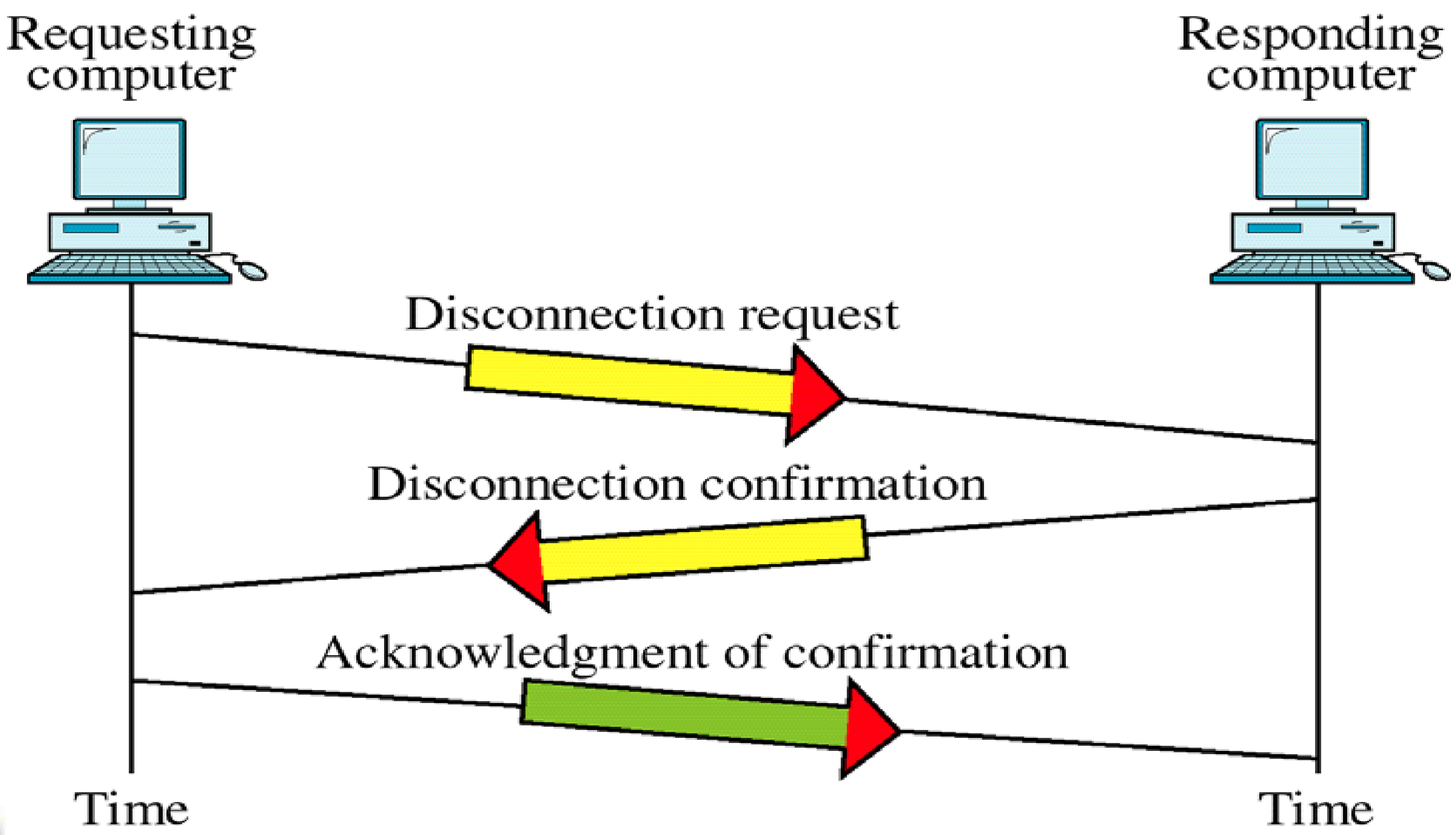
(b) Delayed SYN

(c) Delayed SYN, ACK

Connection Termination

- Entity in CLOSE WAIT state sends last data segment, followed by FIN
- FIN arrives before last data segment
- Receiver accepts FIN
  - Closes connection
  - Loses last data segment
- Associate sequence number with FIN
- Receiver waits for all segments before FIN sequence number
- Loss of segments and obsolete segments
  - Must explicitly ACK FIN

Connection Termination



Graceful Close

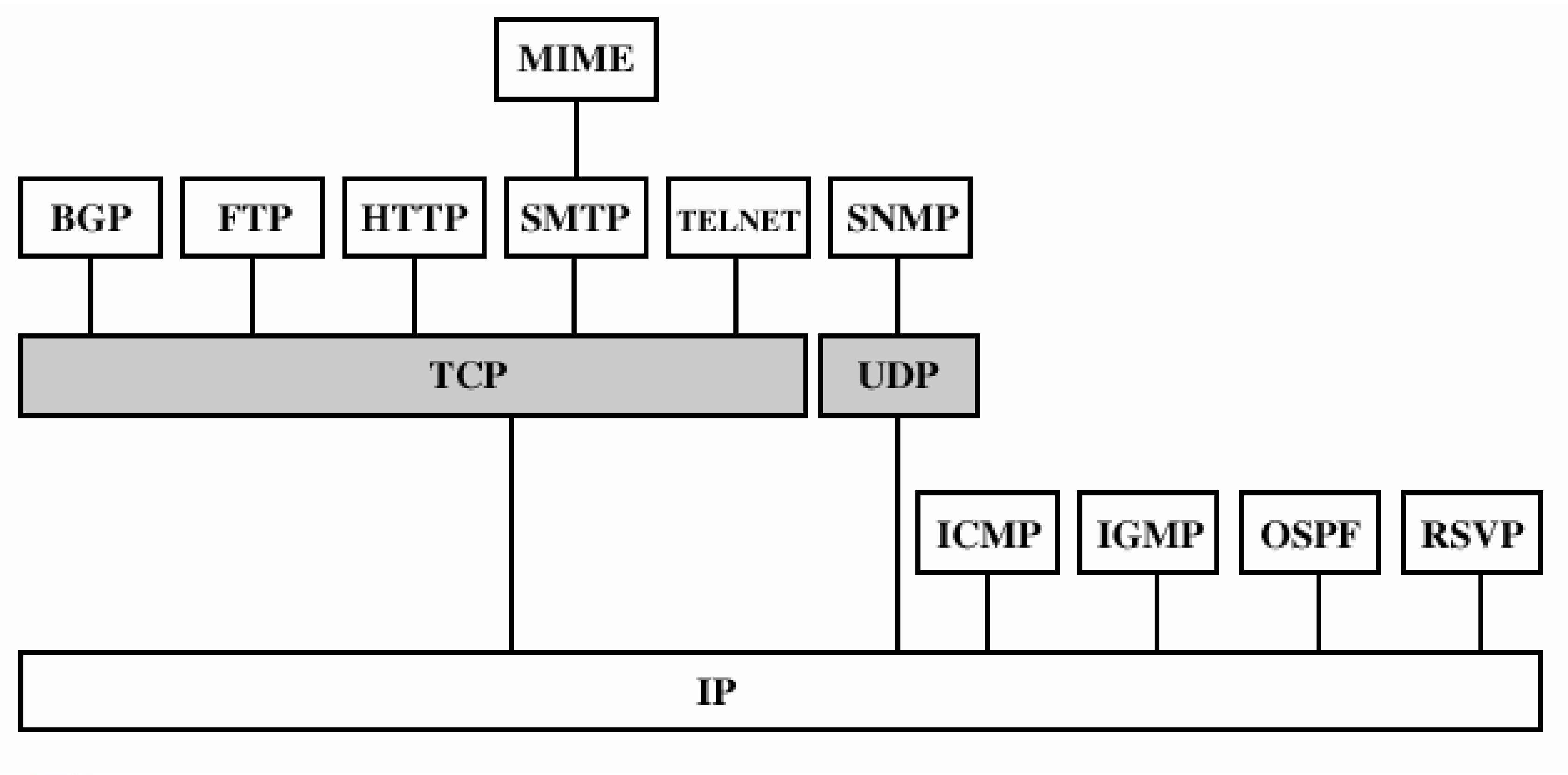
- Send FIN i and receive AN i
- Receive FIN j and send AN j
- Wait twice maximum expected segment lifetime

Crash Recovery

“坚持”而非“放弃”

- After restart all state info is lost
- Connection is half open
  - Side that did not crash still thinks it is connected
- Close connection using persistence timer
  - Wait for ACK for (time out) * (number of retries)
  - When expired, close connection and inform user
- Send RST i in response to any i segment arriving
- User must decide whether to reconnect
  - Problems with lost or duplicate data

Transport-Level Protocols in TCP/IP



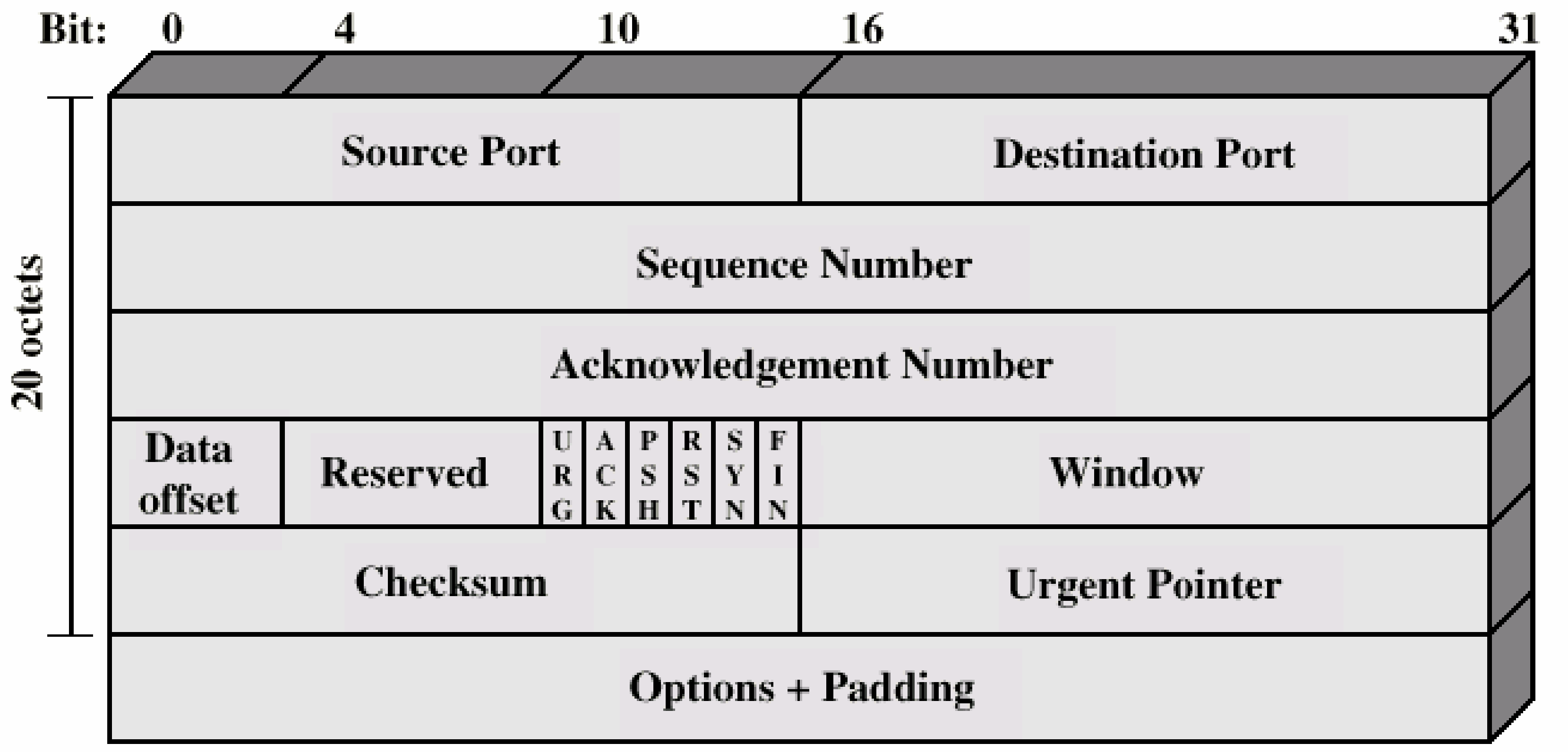
TCP & UDP

- Transmission Control Protocol
  - Connection oriented
  - RFC 793
- User Datagram Protocol (UDP)
  - Connectionless
  - RFC 768

TCP Services

- Reliable communication between pairs of processes
- Across variety of reliable and unreliable networks and internets
- Two labeling facilities
  - Data stream push
    - TCP user can require transmission of all data up to push flag
    - Receiver will deliver in same manner
    - Avoids waiting for full buffers
  - Urgent data signal
    - Indicates urgent data is upcoming in stream
    - User decides how to handle it

TCP Header


Bit: 0 4 10 16 31
20 octets
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Data offset
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window
Checksum
Urgent Pointer
Options + Padding

Items Passed to IP

- TCP passes some parameters down to IP
  - Precedence
  - Normal delay/low delay
  - Normal throughput/high throughput
  - Normal reliability/high reliability
  - Security

TCP Mechanisms (1)

- Connection establishment
  - Three way handshake
  - Between pairs of ports
  - One port can connect to multiple destinations

TCP Mechanisms (2)

- Data transfer
  - Logical stream of octets
  - Octets numbered modulo 2^{32}
  - Flow control by credit allocation of number of octets
  - Data buffered at transmitter and receiver

TCP Mechanisms (3)

- Connection termination
  - Graceful close
  - TCP users issues CLOSE primitive
  - Transport entity sets FIN flag on last segment sent
  - Abrupt termination by ABORT primitive
    - Entity abandons all attempts to send or receive data
    - RST segment transmitted

Implementation Policy Options

- Send
- Deliver
- Accept
- Retransmit
- Acknowledge

Send

- If no push or close TCP entity transmits at its own convenience
- Data buffered at transmit buffer
  - May construct segment per data batch
  - May wait for certain amount of data

Deliver

- In absence of push, deliver data at own convenience
  - May deliver as each in order segment received
  - May buffer data from more than one segment

Accept

- Segments may arrive out of order
  - In order
    - Only accept segments in order
    - Discard out of order segments
  - In windows
    - Accept all segments within receive window

Retransmit

- TCP maintains queue of segments transmitted but not acknowledged
- TCP will retransmit if not ACKed in given time
  - First only
  - Batch
  - Individual

Acknowledgement

- Immediate
- Cumulative

Congestion Control

- RFC 1122, Requirements for Internet hosts
- Retransmission timer management
  - Estimate round trip delay by observing pattern of delay
  - Set time to value somewhat greater than estimate
  - Simple average
  - Exponential average
  - RTT Variance Estimation (Jacobson's algorithm)

Simple Average

$$ARTT(k+1) = \frac{1}{k+1}\sum_{i=1}^{k+1} RTT(i)$$

- *RTT(i)* is the round-trip time observed for *i*th transmission segment
- *ARTT(k)* is the average round-trip time for the first *k* segment

$$ARTT(k+1) = \frac{k}{k+1} ARTT(k) + \frac{1}{k+1} RTT(k+1)$$

- With this formulation, it is not necessary to recalculate the entire summation each time

Exponential Average

- *Let* α *replace* $\frac{k}{k+1}$, $(0<\alpha<1)$
- *Smoothed Round Trip Time*

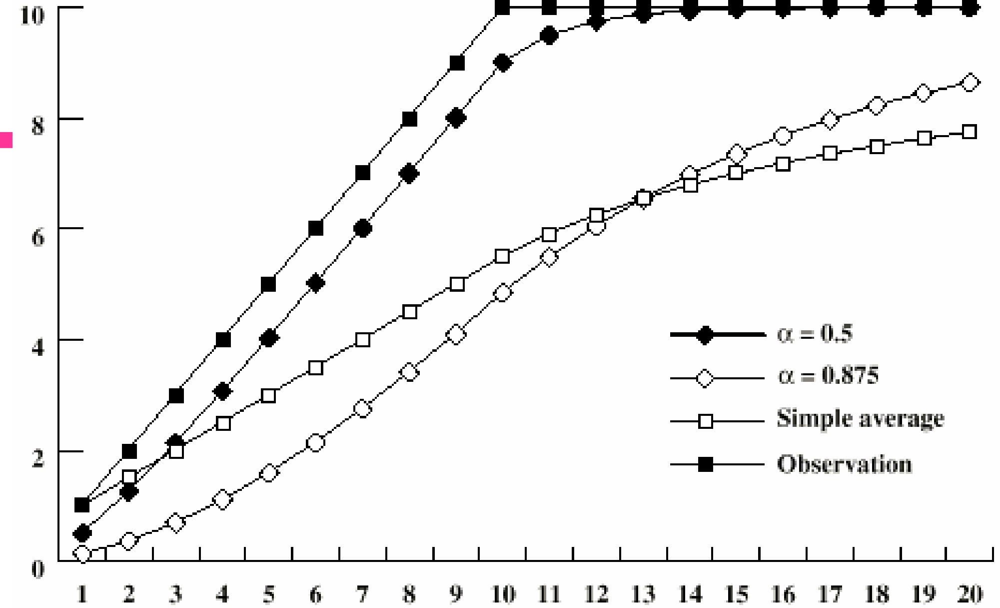
$$SRTT(k+1) = \alpha \times SRTT(k) + (1-\alpha) \times RTT(k+1)$$

- Consider the following expansion of above equation:

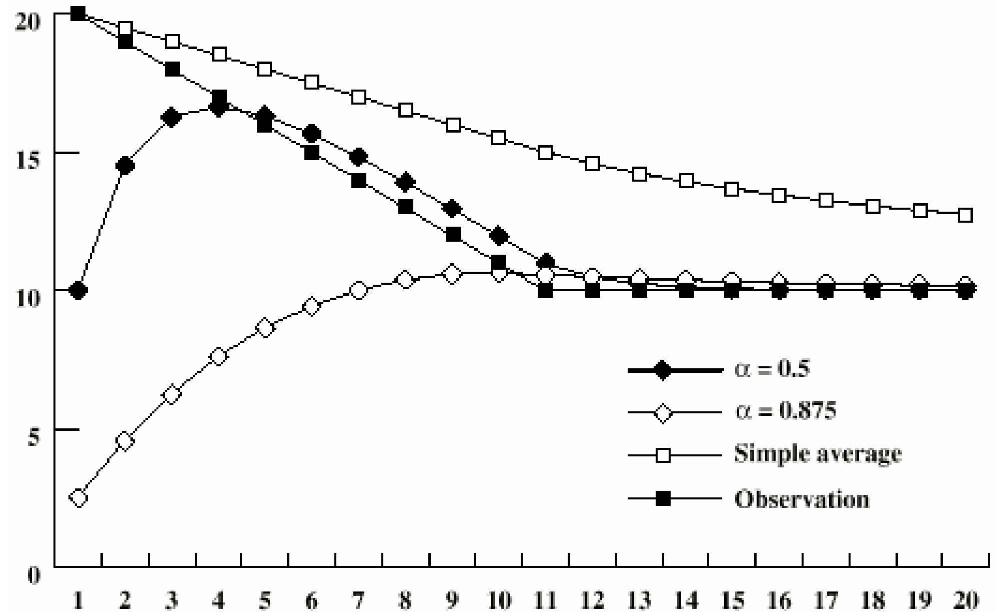
$$SRTT(k+1) = (1-\alpha)RTT(k+1) + \alpha(1-\alpha)RTT(k) + \alpha^2(1-\alpha)RTT(k-1) + \cdots + \alpha^k(1-\alpha)RTT(1)$$

- The older the observation, the less it is counted in the average

Use of Exponential Averaging



(a) Increasing function



(b) Decreasing function

RTO - Retransmission TimeOut

- RTO is transmission timer
  - $RTO(k+1) = SRTT(k+1) + \Delta$
  - $RTO(k+1) = MIN(MAX(\beta \times SRTT(k+1), LBOUND), UBOUND)$
  - $RTO(k+1) = SRTT(k+1) + f \times SDEV(k+1)$

RTT Variance Estimation

- Jacobson's algorithm

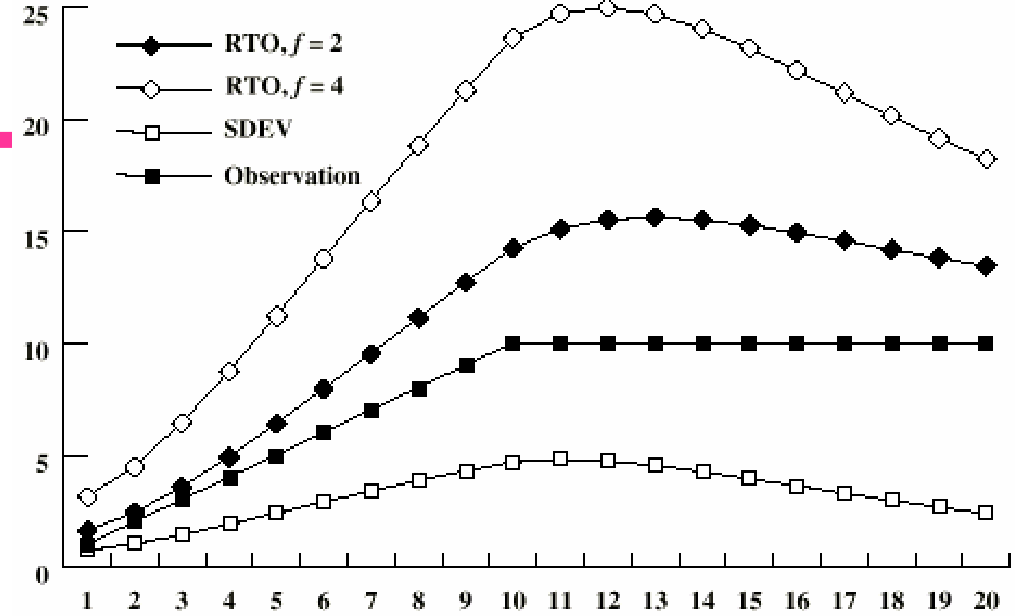
$$SRTT(k+1) = (1-g)\times SRTT(k) + g\times RTT(k+1) \qquad (g = 1-\alpha)$$

$$SERR(k+1) = RTT(k+1) - SRTT(k)$$

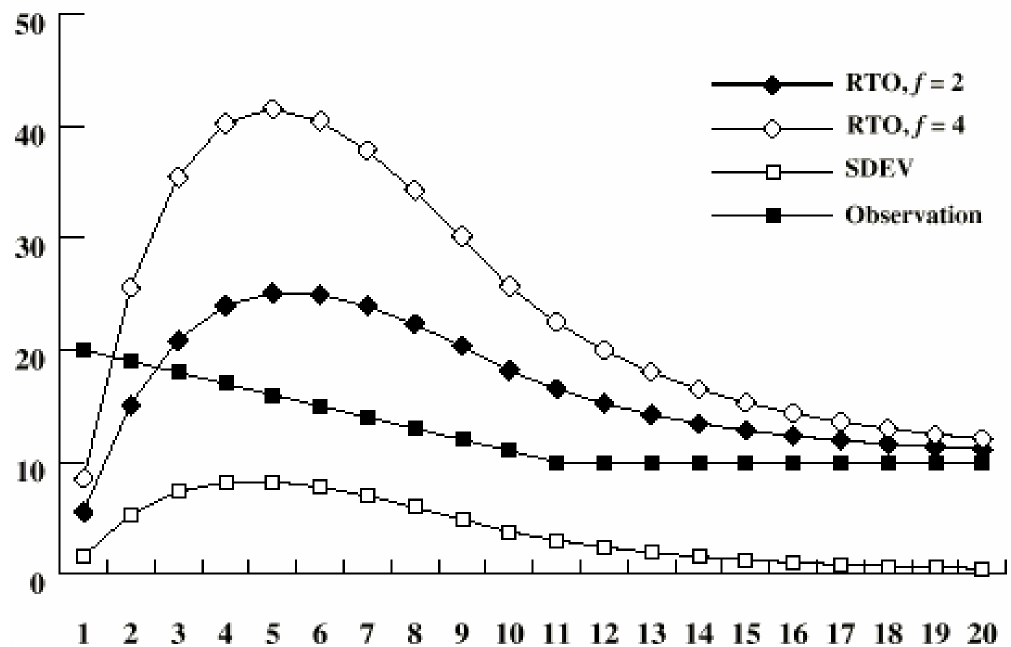
$$SDEV(k+1) = (1-h)\times SDEV(k) + h\times \left|SERR(k+1)\right|$$

$$g = 1/8 = 0.125$$
$$h = 1/4 = 0.25$$
$$f = 4$$

Jacobson's RTO Calculation



(a) Increasing function



(b) Decreasing function

Exponential RTO Backoff

- Since timeout is probably due to congestion (dropped packet or long round trip), maintaining RTO is not good idea
- RTO increased each time a segment is re-transmitted
- RTO = q*RTO
- Commonly q=2
  - Binary exponential backoff

Karn's Algorithm

- If a segment is re-transmitted, the ACK arriving may be:
  - For the first copy of the segment
    - RTT longer than expected
  - For second copy
    - RTT will be much too small
- No way to tell
- Do not measure RTT for re-transmitted segments
- Calculate backoff when re-transmission occurs
- Use backoff RTO until ACK arrives for segment that has not been re-transmitted

Window Management

- Slow start
  - awnd = MIN[credit, cwnd]
  - Start connection with cwnd=1
  - Increment cwnd at each ACK, to some max
- Dynamic windows sizing on congestion
  - When a timeout occurs
  - Set slow start threshold to half current congestion window
    - ssthresh=cwnd/2
  - Set cwnd = 1 and slow start until cwnd=ssthresh
    - Increasing cwnd by 1 for every ACK
  - For cwnd >=ssthresh, increase cwnd by 1 for each RTT

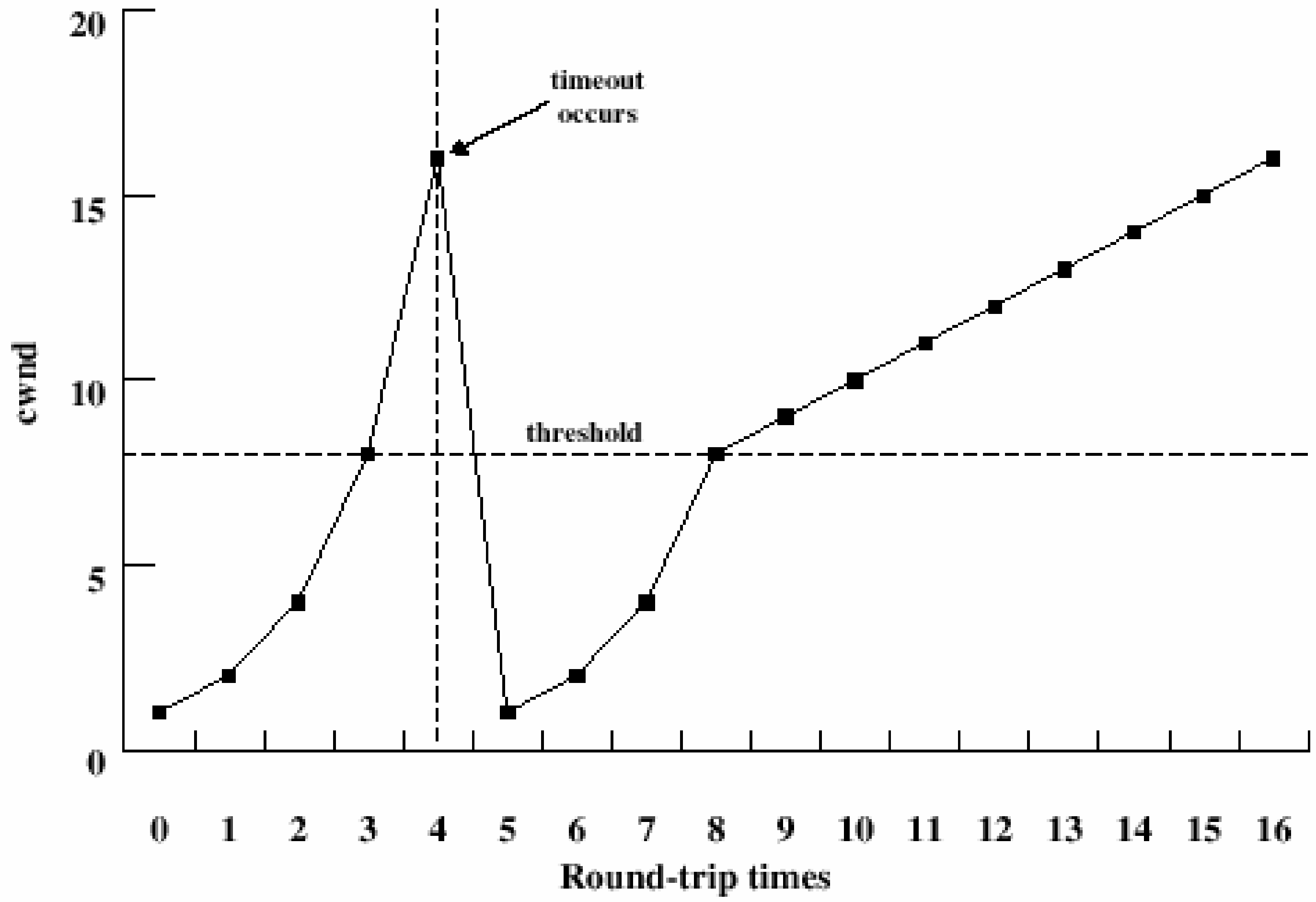


Figure 17.14 Illustration of Slow Start and Congestion Avoidance

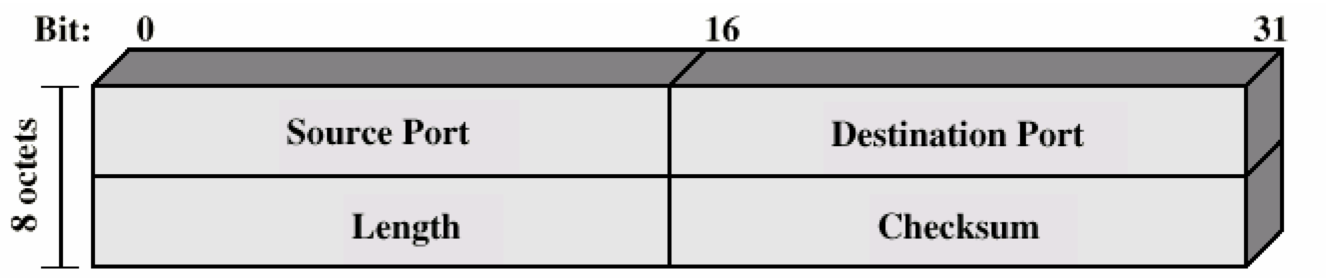
UDP

- User datagram protocol
- RFC 768
- Connectionless service for application level procedures
  - Unreliable
  - Delivery and duplication control not guaranteed
- Reduced overhead
- e.g. network management (Chapter 19)

UDP Uses

- Inward data collection
- Outward data dissemination
- Request-Response
- Real time application

UDP Header


Bit:
0
16
31
8 octets
Source Port
Destination Port
Length
Checksum

Required Reading

- Stallings, D&CC, 6e, Chapter 17
- Forouzan, Introduction to DC&N, 2e, Chapter 22
- 谢希仁，计算机网络，第三版，第八章
- RFC 793 ,768, 1122, 2001

Problems

- 17.3, 17.9,17.15, 17.20