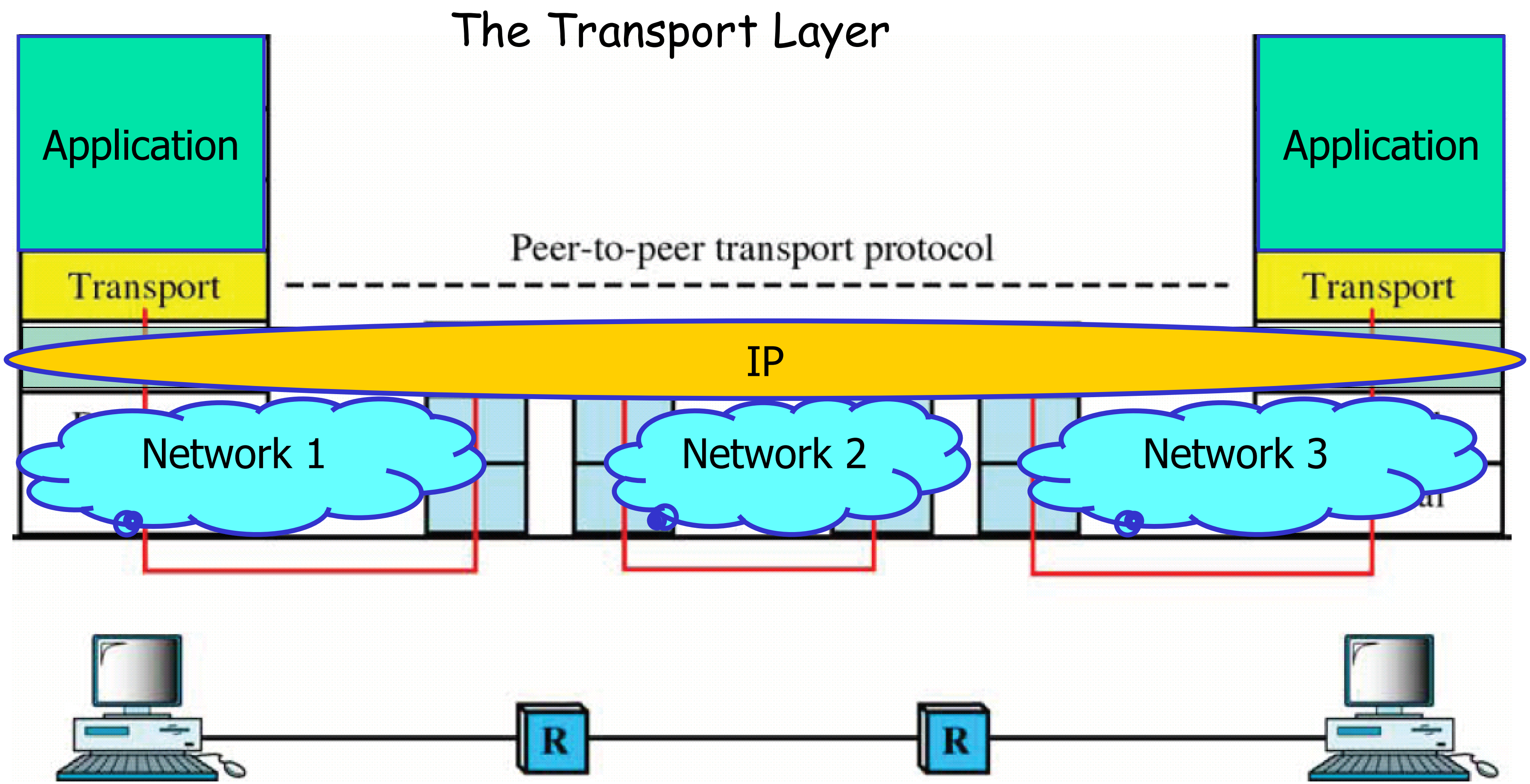
Computer Networks

Wenzhong Li
Nanjing University
Fall 2014

Chapter 5. End-to-End Protocols

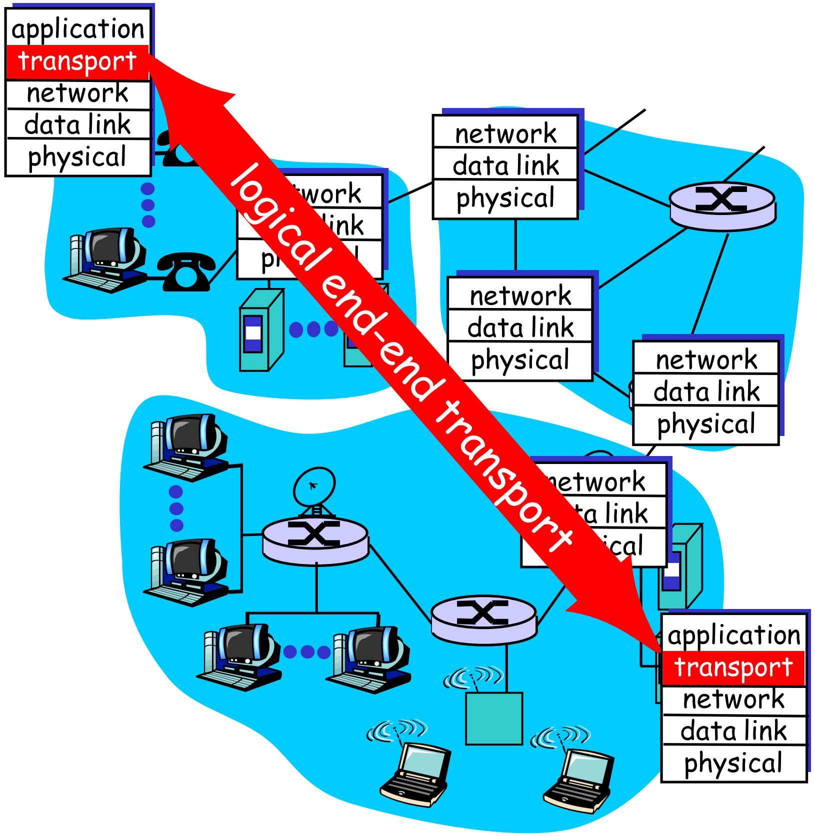
- Transport Services and Mechanisms
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)
- TCP Congestion Control
- Real-time Transport Protocol (RTP)
- Session Initiation Protocol (SIP)
- Real Time Streaming Protocol (RTSP)

Transport Services and Mechanisms



Internet Transport Services

- Provide logical communication between app processes running on different hosts
- Transport protocols run in end systems
  - Send side: breaks app messages into segments, passes to network layer
  - Receive side: reassembles segments into messages, passes to app layer
- More than one transport protocol available to apps
  - Internet: TCP and UDP

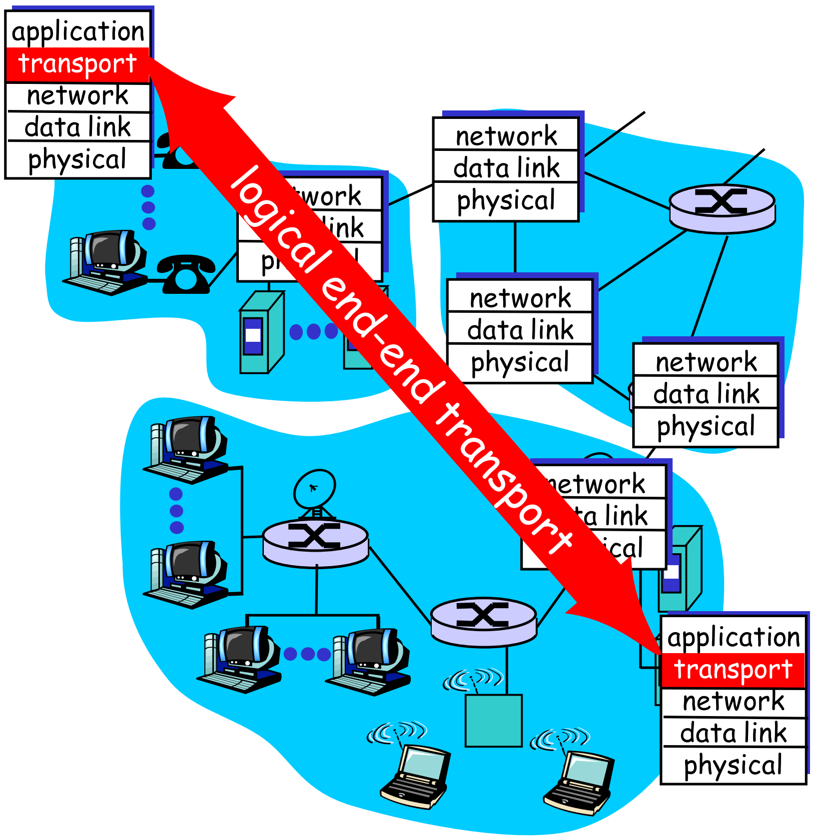


Transport vs. network layer

- Network layer:
  - logical communication between hosts
- Transport layer:
  - logical communication between processes
    - Relies on, enhances, network layer services
  - Multiplexing (复用) & Demultiplexing (分用)

Internet Transport-Layer Protocols

- Reliable, in-order delivery (TCP)
  - Connection oriented
  - Congestion control
  - Flow control
- Unreliable, unordered delivery (UDP)
  - No-frills extension of "best-effort" IP
- Services not available:
  - delay guarantees
  - bandwidth guarantees



Internet Transport Mechanisms

- Addressing and multiplexing
- Connection-oriented mechanisms
  - Flow control
  - Connection establishment and termination
  - Reliable sequencing communication

Addressing

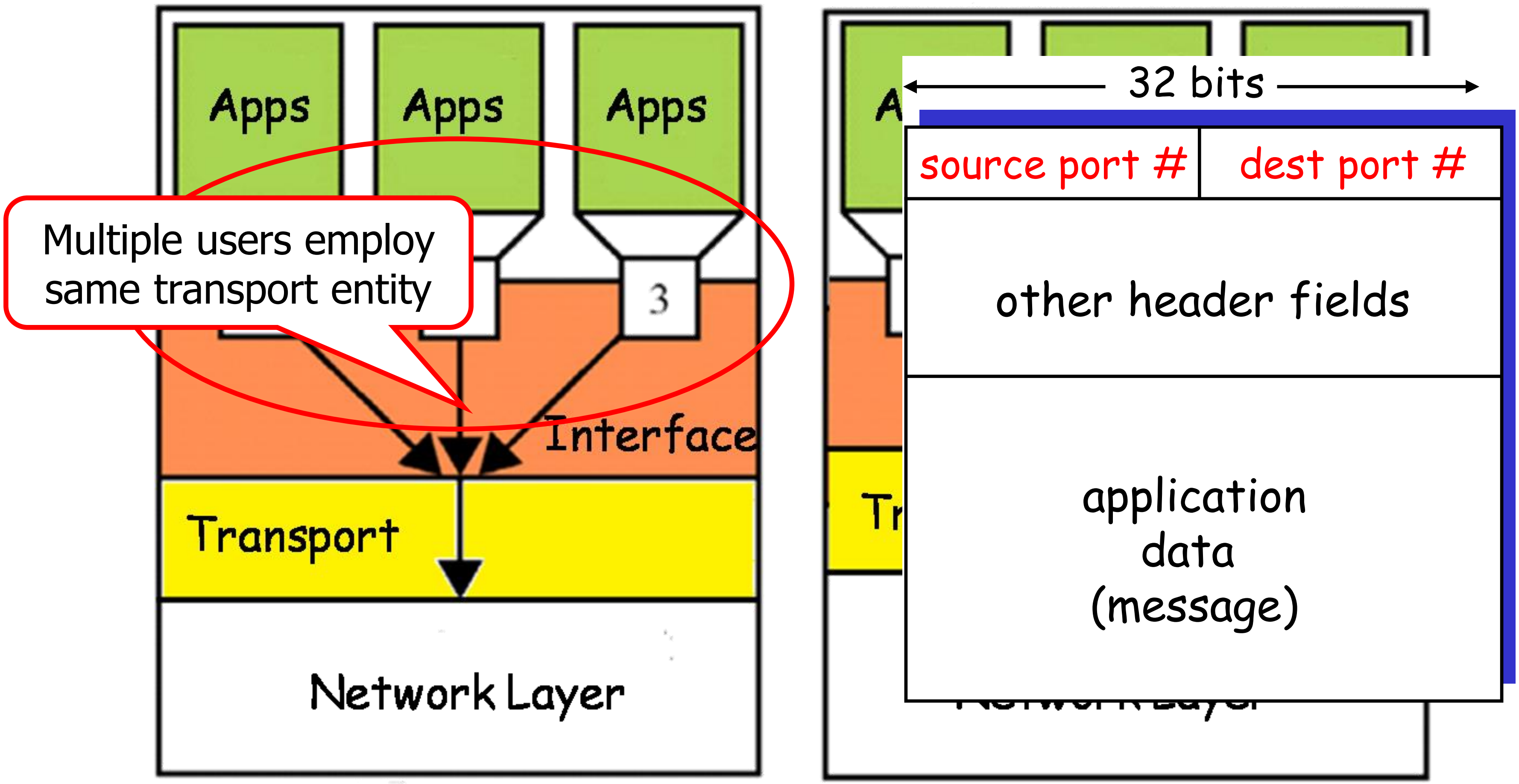
3-level address for application processes on hosts

- Process identification
  - *SAP* on transport entity, represents a particular transport service (TS) user
  - Port number on TCP/UDP
- Transport entity identification
  - Generally only one of each type per host
  - Transport protocol identity (TCP, UDP)
- Host address
  - A global Internet address for attached hosts
- Internet TCP/UDP addressing
  - <HostIP, Port>, called a socket

Finding Addresses

- How TS user get to know each other
  - Target port: use conventions
- 4 methods can be used for target host
  - Know address ahead of time
  - Well known addresses
  - Name server, directory service
  - Sending process request to known address
    - Create a TS user on target host

Multiplexing



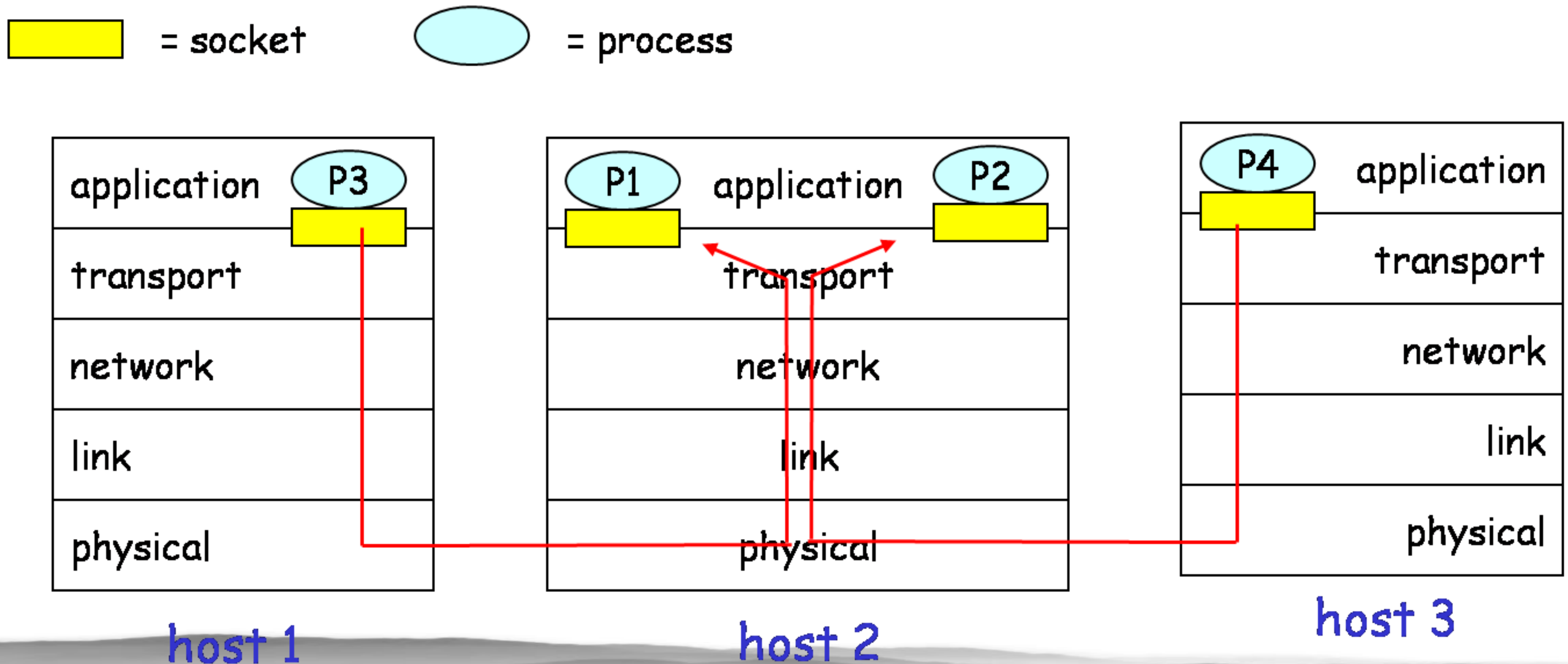
Multiplexing / Demultiplexing

Multiplexing at send host:

Gathering data from multiple sockets, enveloping data with header (for demultiplexing)

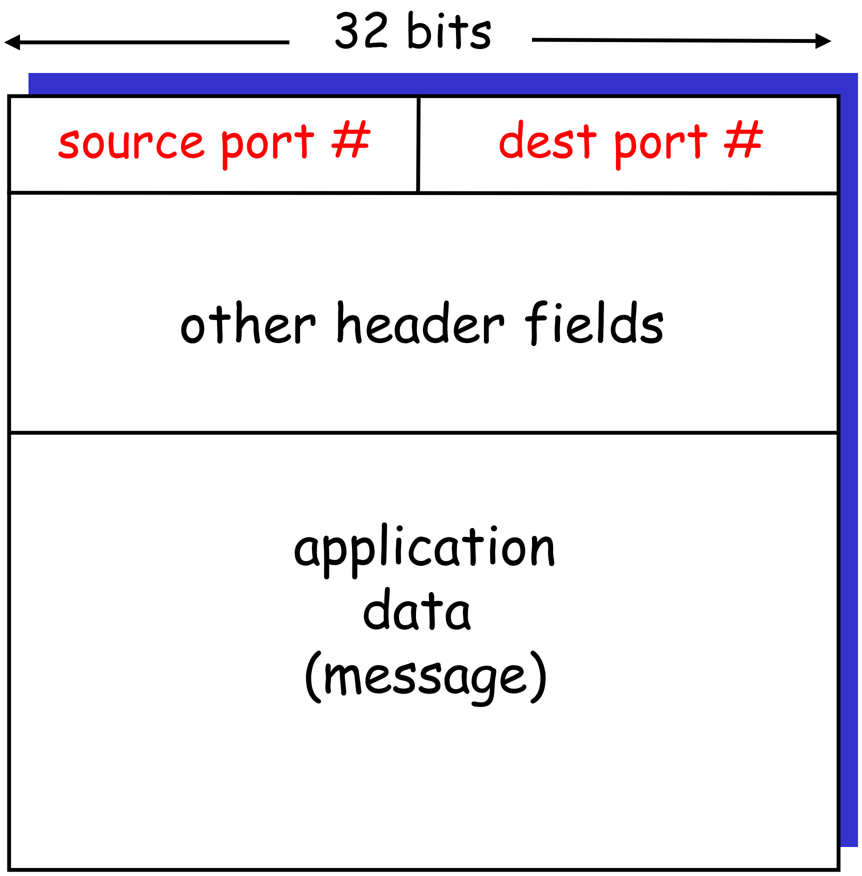
Demultiplexing at rcv host:

Delivering received segments to correct socket



Demultiplexing

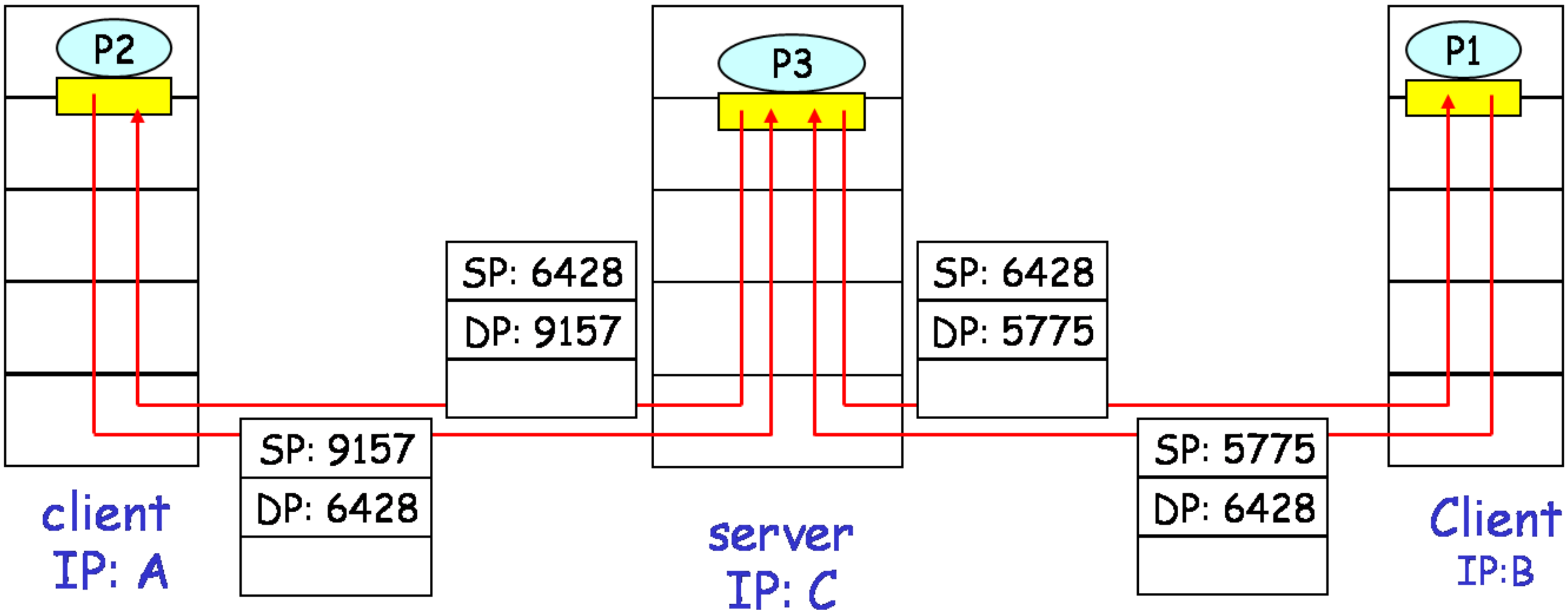
- TE receives IP datagrams
- Each datagram carries 1 transport-layer segment
- Each datagram has source IP address, destination IP address
- Each segment has source, destination port number
- TE uses IP addresses & port numbers to direct segment to appropriate socket



TCP/UDP segment format

Demultiplexing in UDP

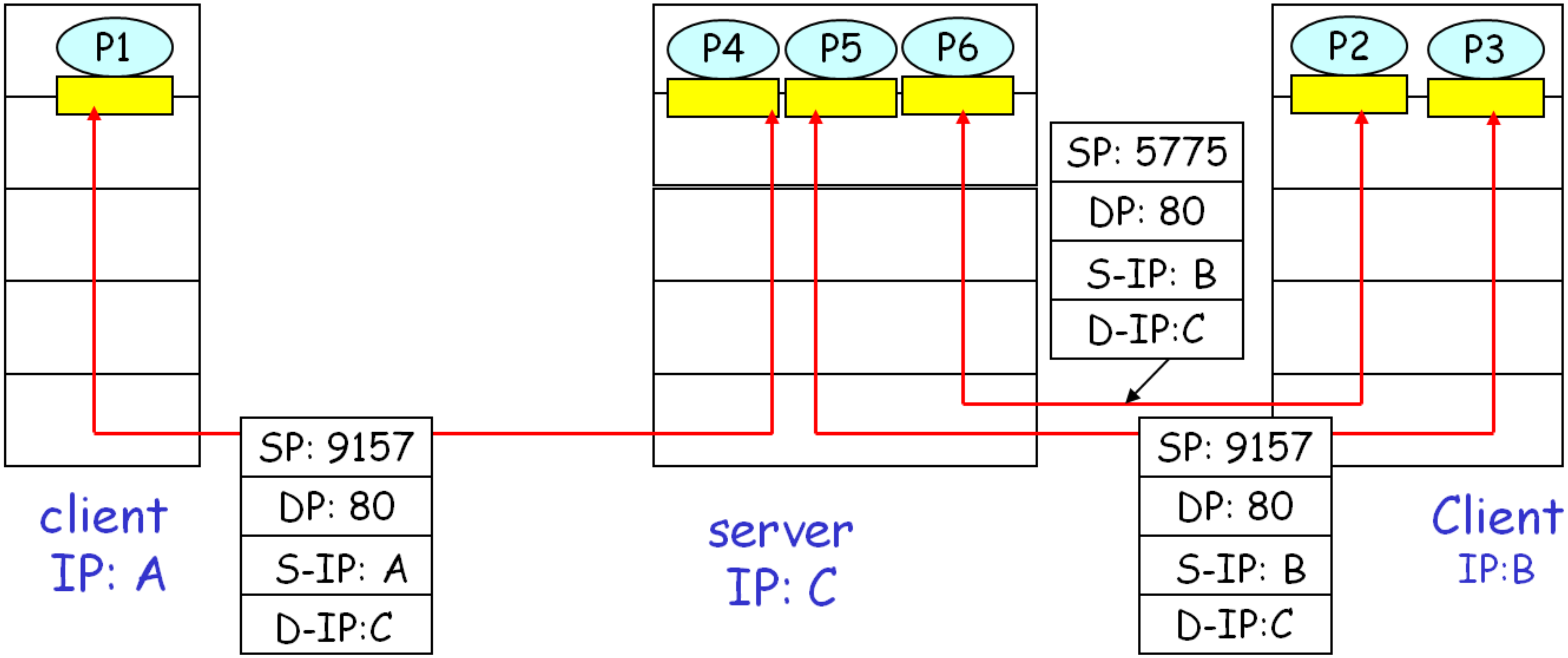
```
DatagramSocket serverSocket = new DatagramSocket(6428);
```



SP provides “return address”

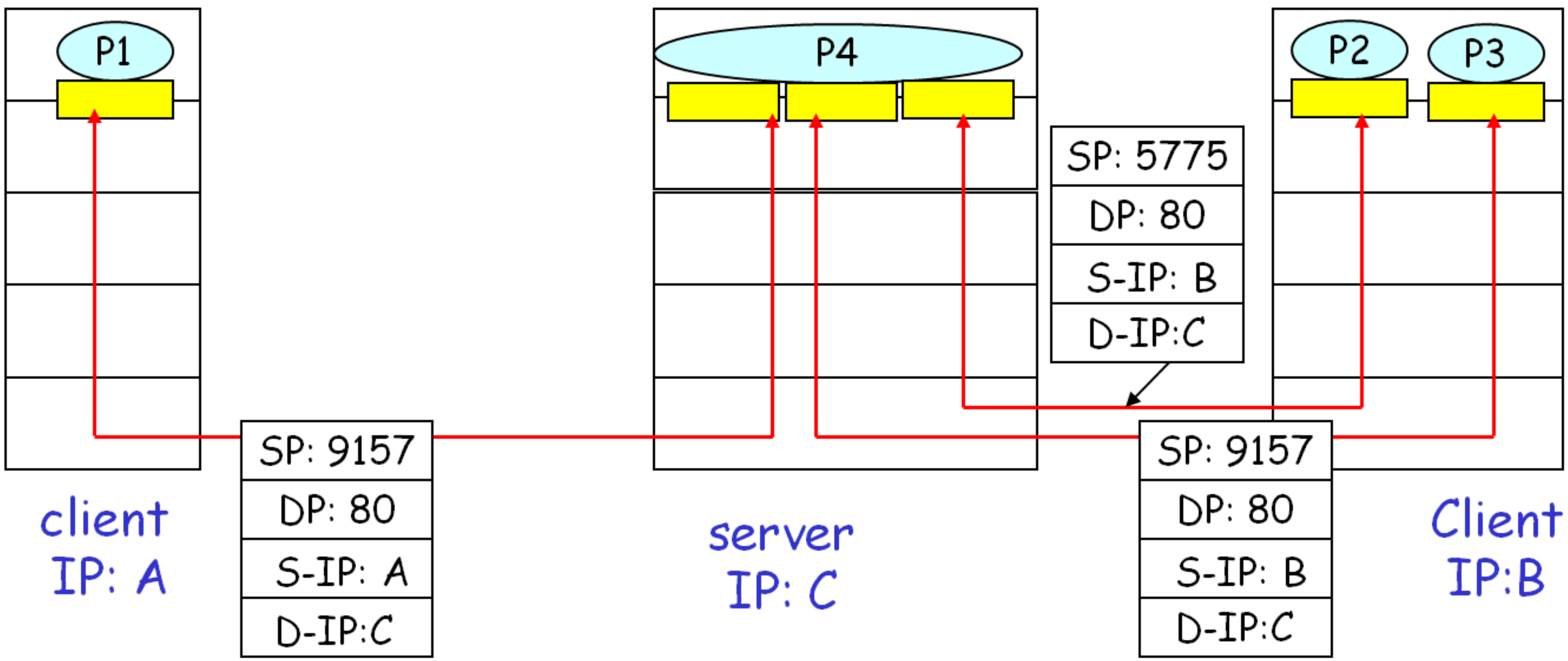
Demultiplexing in TCP

TCP connection identified by 4-tuple:
<Source IP, Source Port, Destination IP, Destination Port>



Threaded Web Server

Application with multiple threads, can be seen as a downward multiplexing

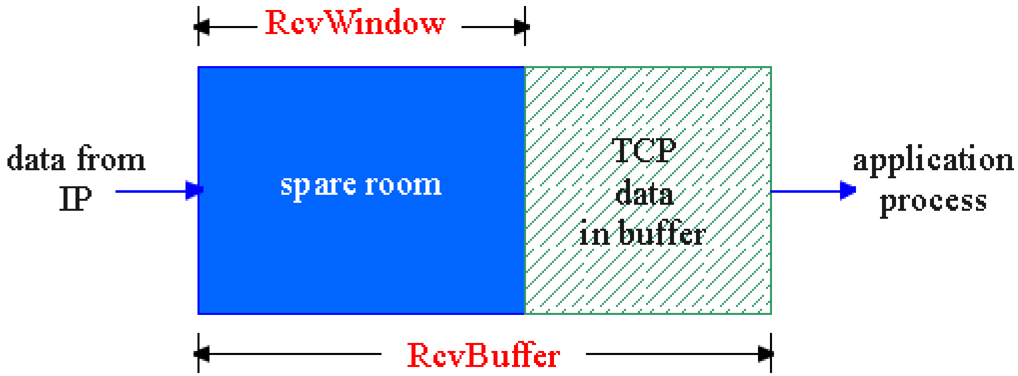


Flow Control

- Flow control in multi-layers
  - Sender won't overflow receiver's buffer by transmitting too much too fast
  - The receiving TS user can not keep up
    - Transport buffer may overflow
  - The receiving transport entity can not keep up
    - Network buffer may overflow
- Speed-matching service
  - Matching the send rate to the receiving app's receive rate

Receive Buffer

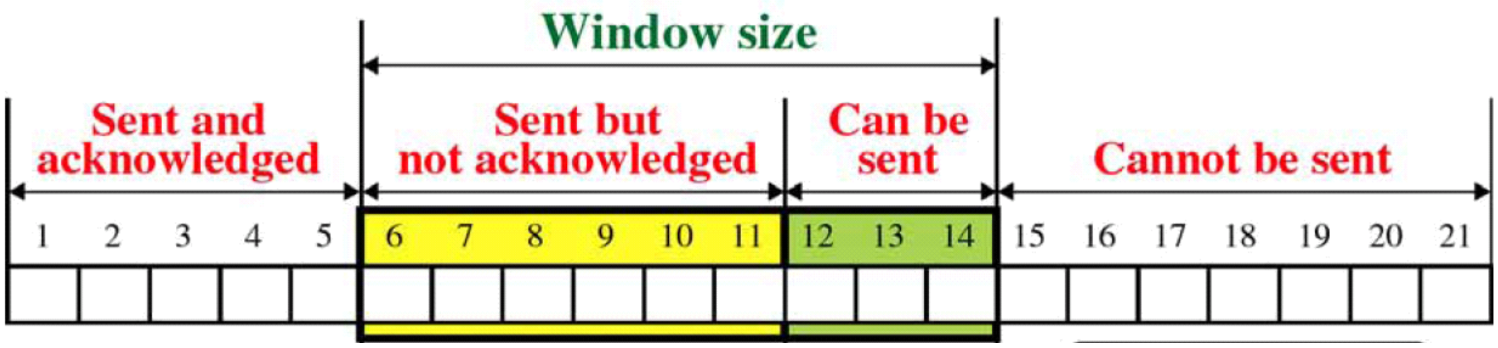
- The receive side of TS connection has a receive buffer



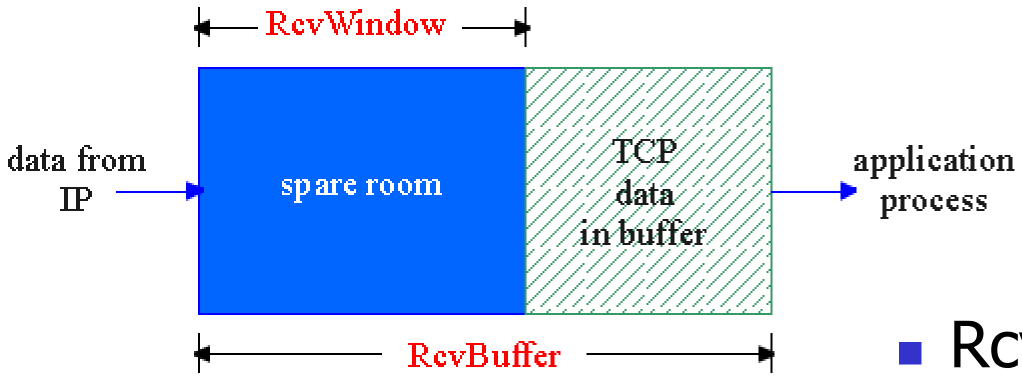
- App process may be slow at reading from buffer

Handle Flow Control

- Fixed sliding window
  - Works well on reliable direct links
- Problem:
  - Failure to receive *ACK* is taken as flow control indication
  - Can not distinguish between lost segment and flow control
  - Not flexible for congestion control mandated in Internet



Credit Scheme (1)



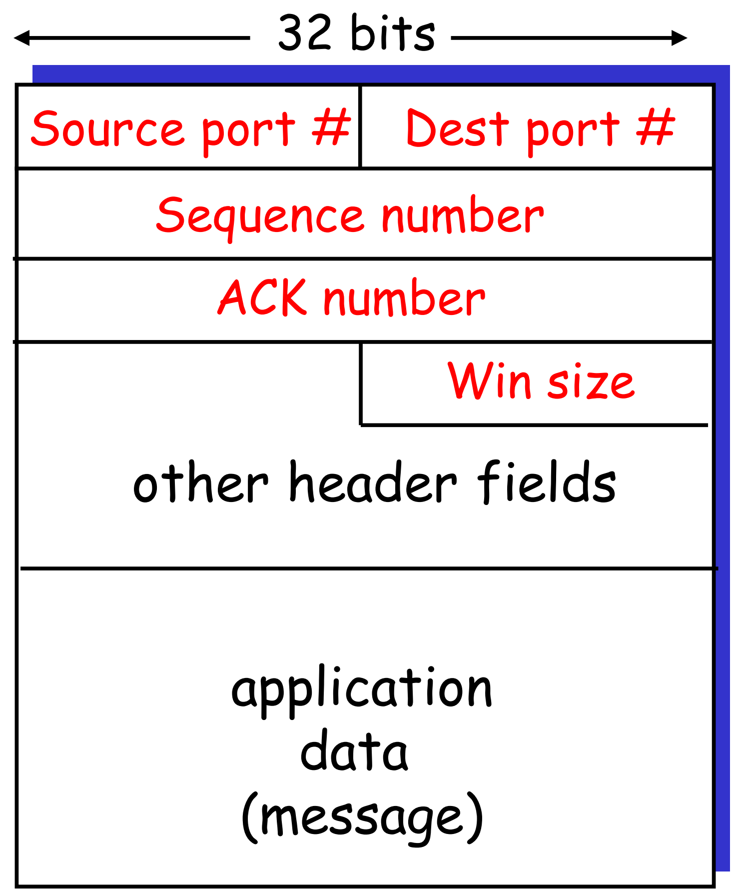
- Spare room in receive buffer

  RcvWindow = RcvBuffer - [LastByteRcvd - LastByteRead]

- Rcvr advertises spare room (credits) by including value of RcvWindow in segments
- Sender limits unACKed data to RcvWindow

Credit Scheme (2)

- Greater control on Internet
- Decouples flow control from *ACK*
  - May *ACK* without granting credit
- Each octet has a sequence number
- Each transport segment has seq number, ack number and window size in header

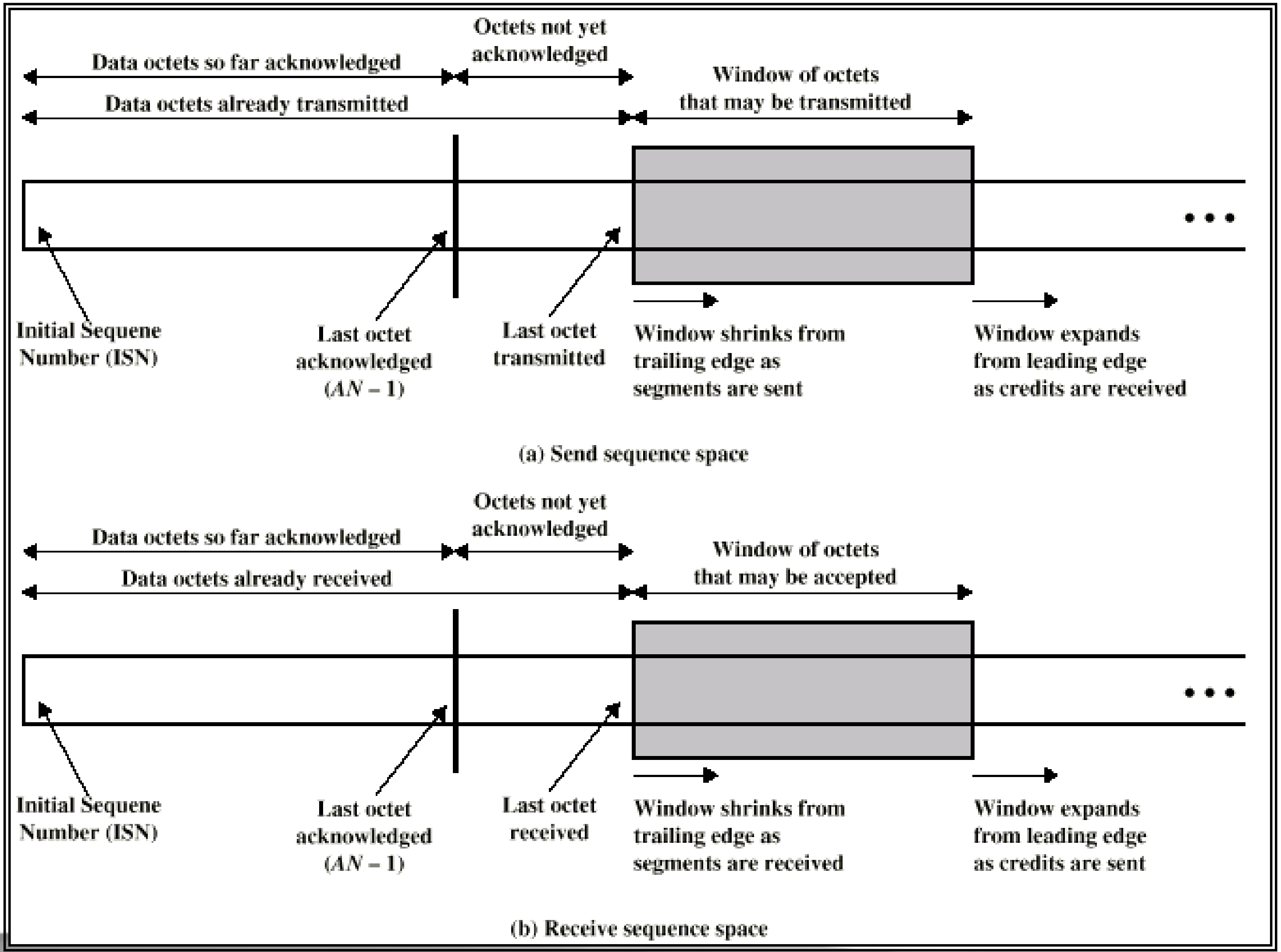


TCP segment format

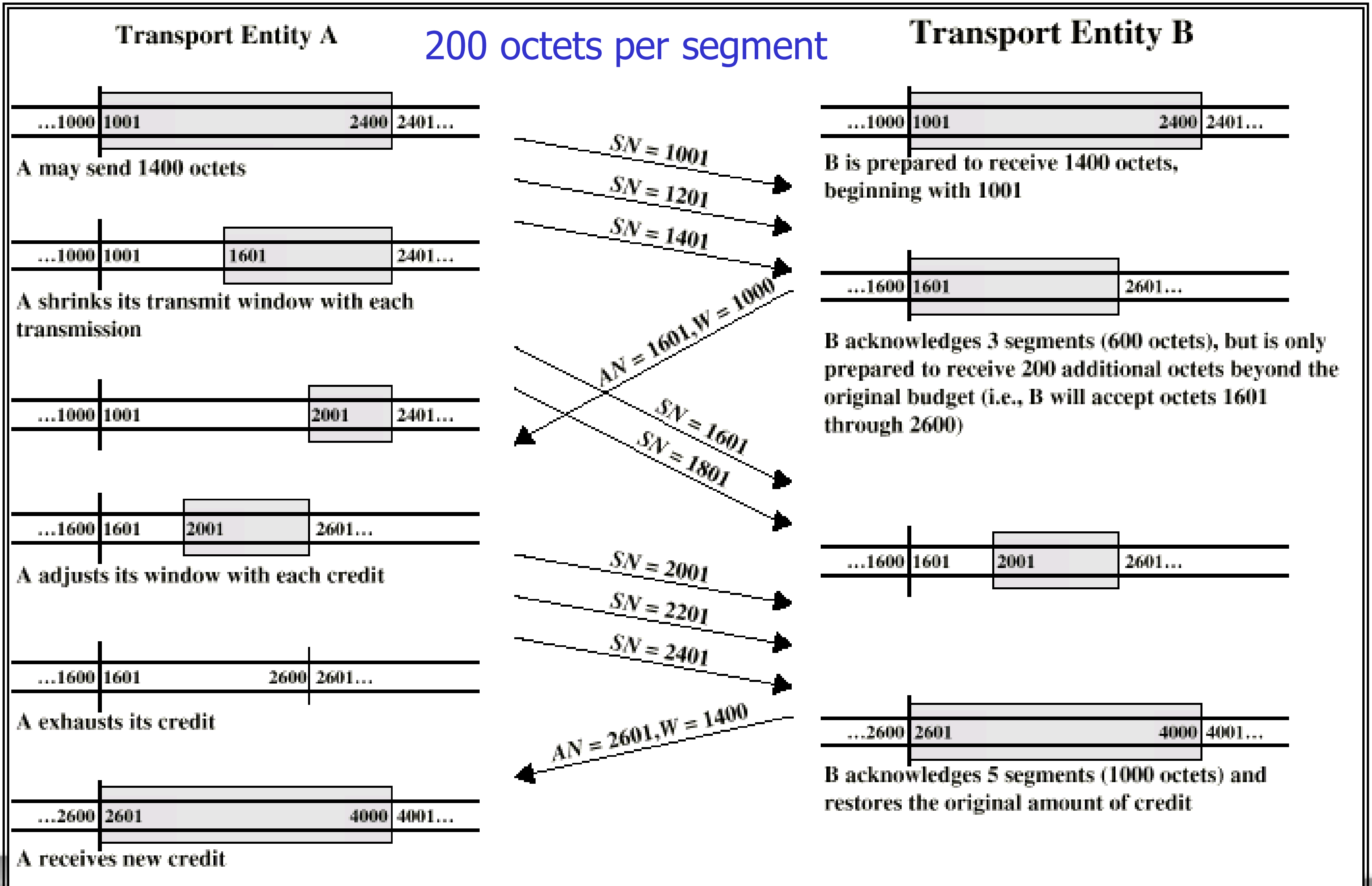
Use of Header Fields

- When sending a segment
  - seq number (SN) 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 from i to i+j-1

Sending and Receiving Windows



Credit Allocation Procedure



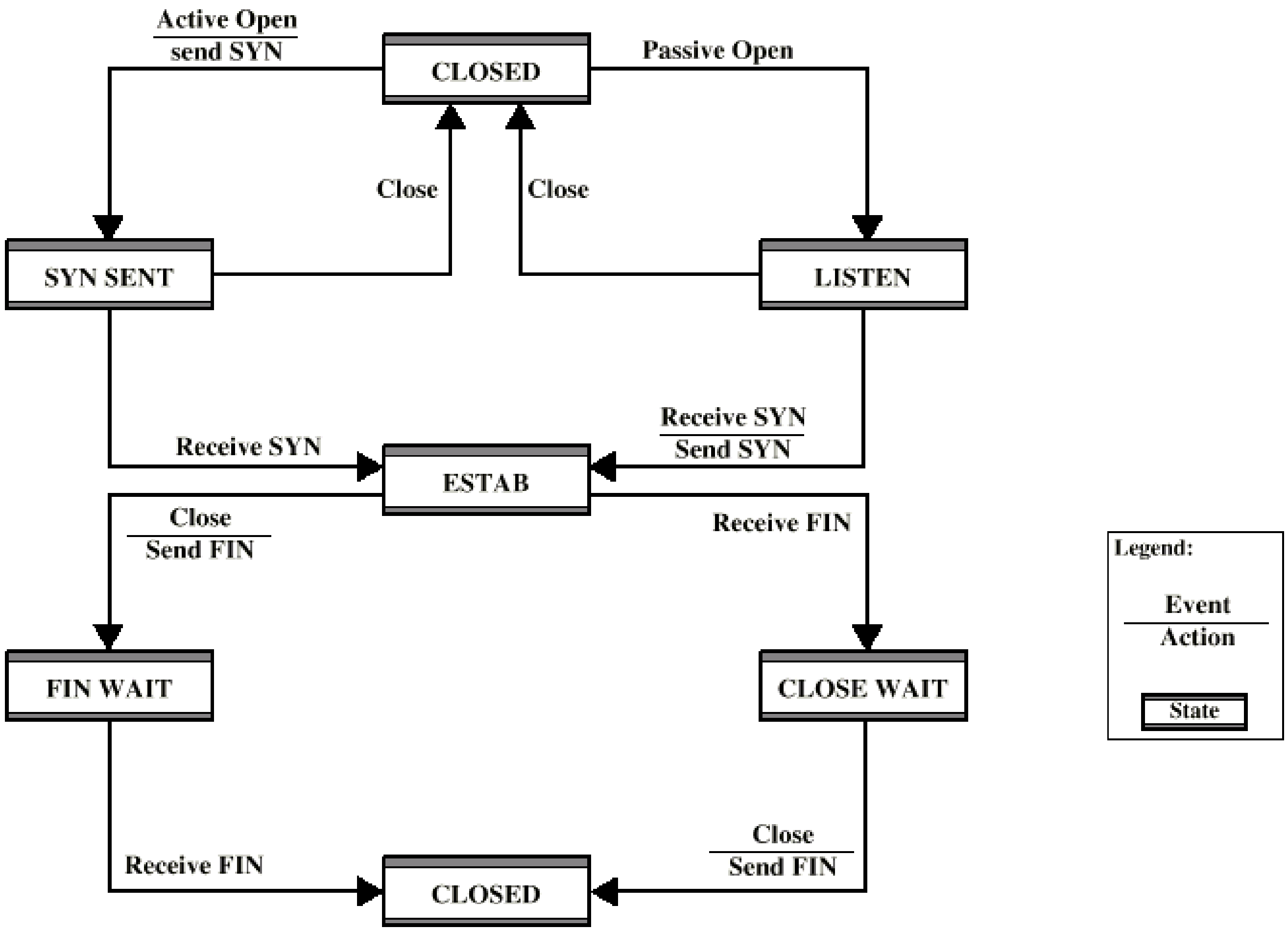
Connection Establishment and Termination

- 2 ends establish connection before exchanging data segments
  - Allow each end to know the other exists
  - Negotiation of optional parameters
    - e.g. initial *seq numbers*, max segment size, max window size, IP *QOS*
  - Allocation of transport entity resources
    - e.g. buffers
- Gets mutual agreement
  - On reliable sequencing networks
  - On unreliable IP internets

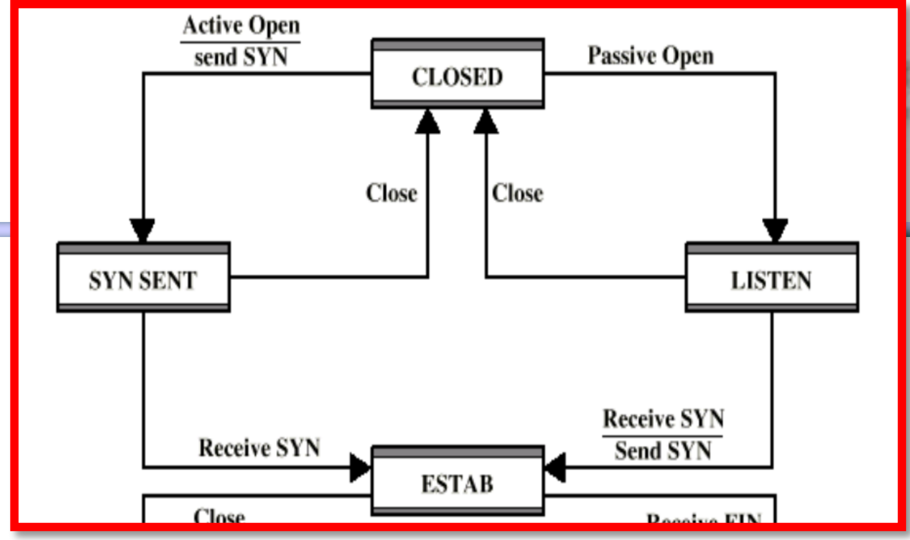
Reliable Sequencing Network Service

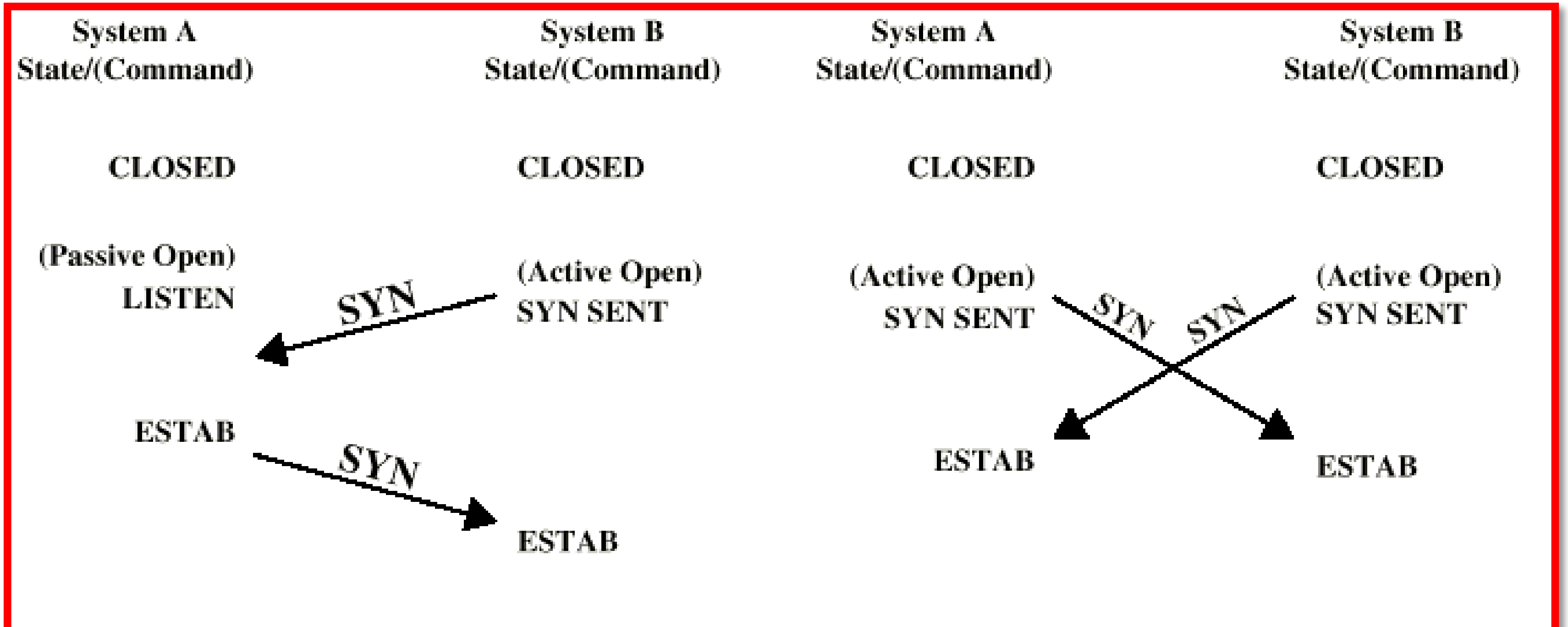
- Assume arbitrary length message delivered in sequence
- Assume virtually 100% reliable delivery by network service
- Examples
  - Reliable packet switched network using X.25
  - Frame relay using LAPF control protocol
  - IEEE 802.3 using connection oriented LLC service

Simple Connection State Diagram



Connection Establishment





(a) Active/Passive Open

(b) Active/Active Open

Handle Pending Request

- SYN comes while requested TS user is not listening
  - Reject with RST (Reset)
  - Queue request until matching open issued
  - Signal TS user to notify of pending request
    - Just accept without passive open

Termination

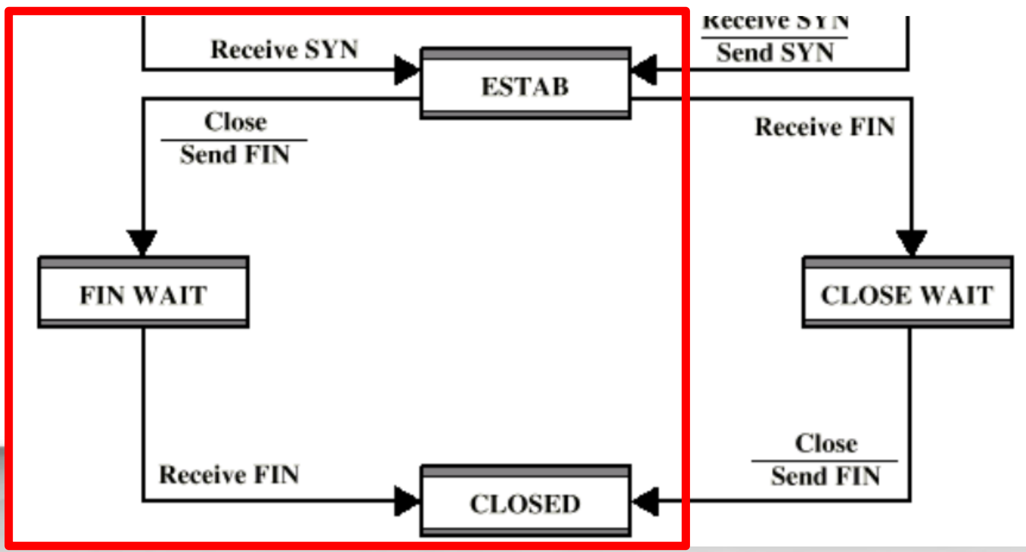
- Either or both sides issue terminate
- Reach mutual agreement

- Abrupt termination
  - Pending segments from other end may lost

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

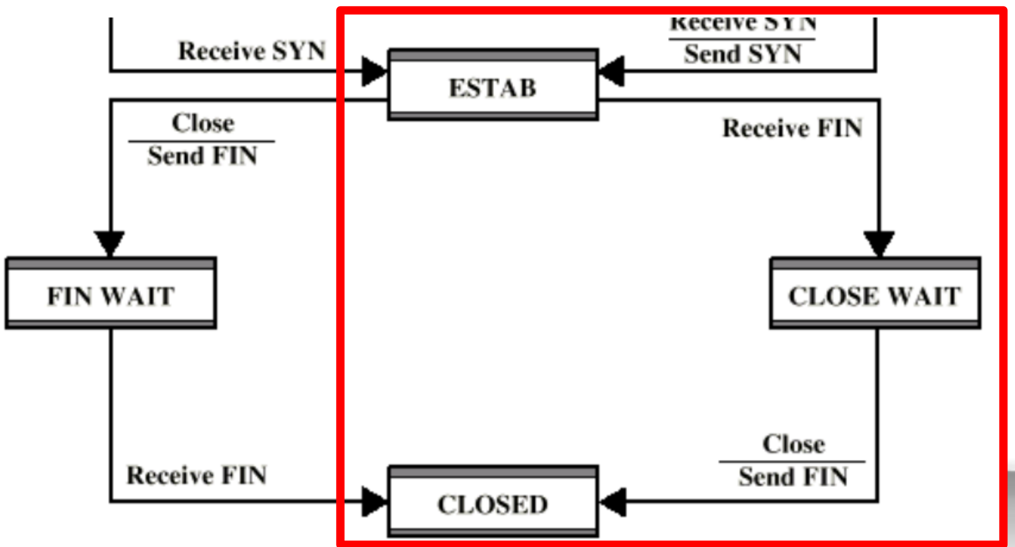
Side Initiating Termination

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



Side Not Initiating Termination

- FIN received, Inform TS user
  - No more data come from the other end
- 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



Unreliable Network Service

- Segments may get lost
- Segments may arrive out of order

- Examples
  - Internet using IP
  - Frame relay using LAPF core
  - IEEE 802.3 using unacknowledged connectionless LLC

Seven Issues

- Ordered Delivery
- Retransmission Strategy
- Duplication Detection
- Traffic Control
- Connection Establishment
- Connection Termination
- Crash Recovery

- 按序交付
- 重传策略
- 副本检测
- 流量控制
- 连接建立
- 连接终止
- 崩溃恢复

(1) Ordered Delivery

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

(2) Retransmission Strategy

- Problem
  - Segment damaged in transit, or
  - Segment dropped due to buffer overflow at router
  - Sender may not know of failure
- Handle
  - Receiver: acknowledge successful receipt
  - Can use cumulative acknowledgement
  - Sender: waiting Timer for ACK timeout triggers re-transmission

Setting Re-transmission Timer

- Should adapt to changing network conditions
  - Fixed timer is not suitable
  - Too small leads to unnecessary re-transmissions
  - Too large and response to lost segments is slow

- Can be set a bit longer than round-trip time
  - Receiver may not ACK immediately
  - Sender can not distinguish between ACK of original segment and re-transmitted segment
  - Should adapt to network congestion

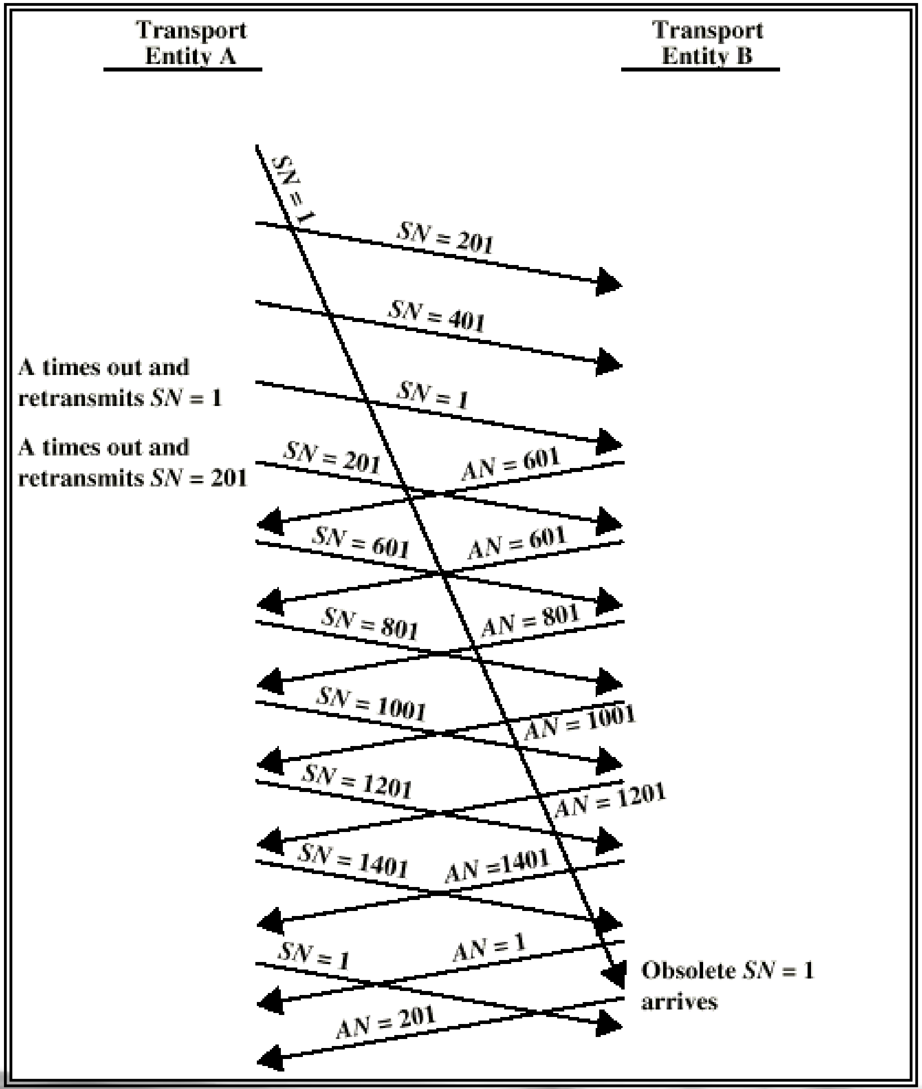
(3) Duplication Detection

- If re-transmission Timer timeout, sender re-transmits segment
  - If segment just delayed, receiver must recognize duplicates
- Duplicate received within a connection
  - Receiver assumes *ACK* lost and re-transmits *ACKs*
  - Sender must not get confused with multiple *ACKs*
  - Space of *seq number* should be large enough to not cycle within maximum life of segment
- Duplicate received after connection closed

Incorrect Duplicate Detection (1)

- Seq number cycled within life of a segment

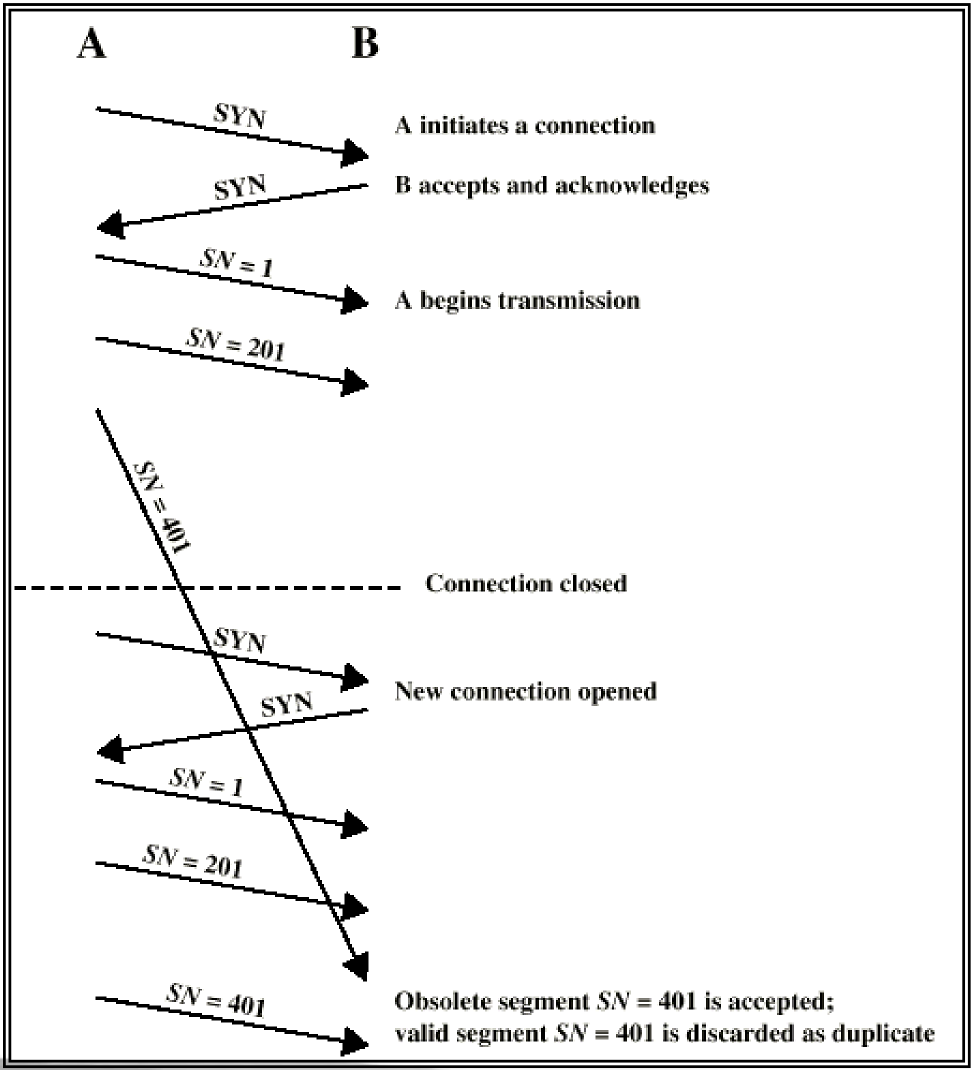
Q: How to handle it?



Incorrect Duplicate Detection (2)

- Segments slipped into another connection

Q: How to handle it?



(4) Traffic Control

Credit allocation flow control mechanism

- Suppose that the last octet of data received by B was octet number i-1, and that the last segment issued by B was (AN=i, W=j). Then
  - To increase credit to an amount k (k>j) when no additional data have arrived, B issues (AN=i, W=k)
  - To acknowledge an incoming segment containing m octets of data (m<j) without granting additional credit, B issues (AN=i+m, W=j-m)
- If an ACK/CREDIT segment is lost, little harm is done. Future acknowledgments will resynchronize the protocol.
- Further, if the sender times out and retransmits a data segment, it triggers a new acknowledgment.

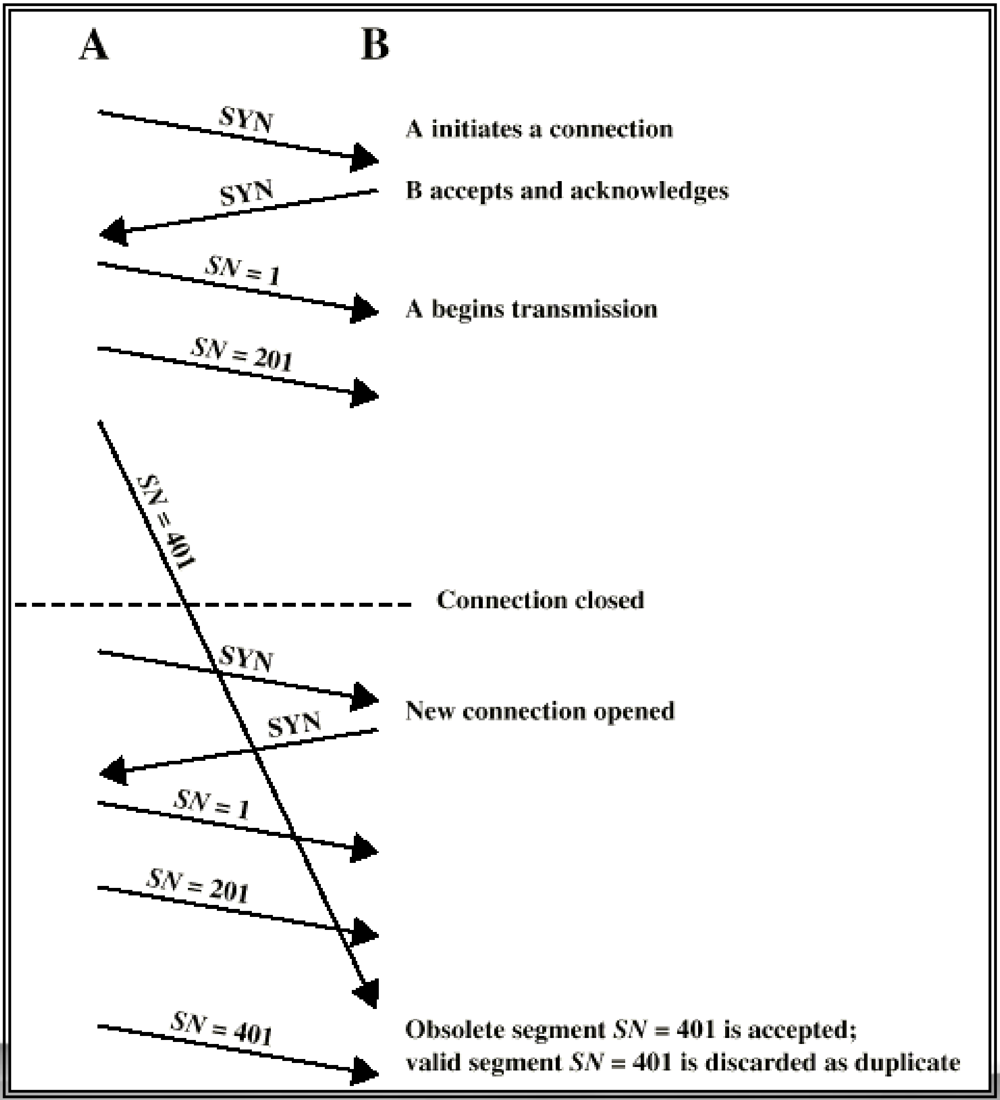
- Credit allocation deadlock
  - Segment with AN=i, W=0 closing rcv-window
  - Should send AN=i, W=j to reopen, but this maybe lost
  - Now sender thinks window is closed, receiver thinks it is open and wait
- Handle
  - Use window timer
  - If timer expires without any receiving, send something
  - Could be re-transmission of previous segment

(5) Connection Establishment

- 2-way handshake
  - A sends SYN, B replies with SYN
  - Lost SYNs handled by re-transmission
  - Ignore duplicate SYNs once connected

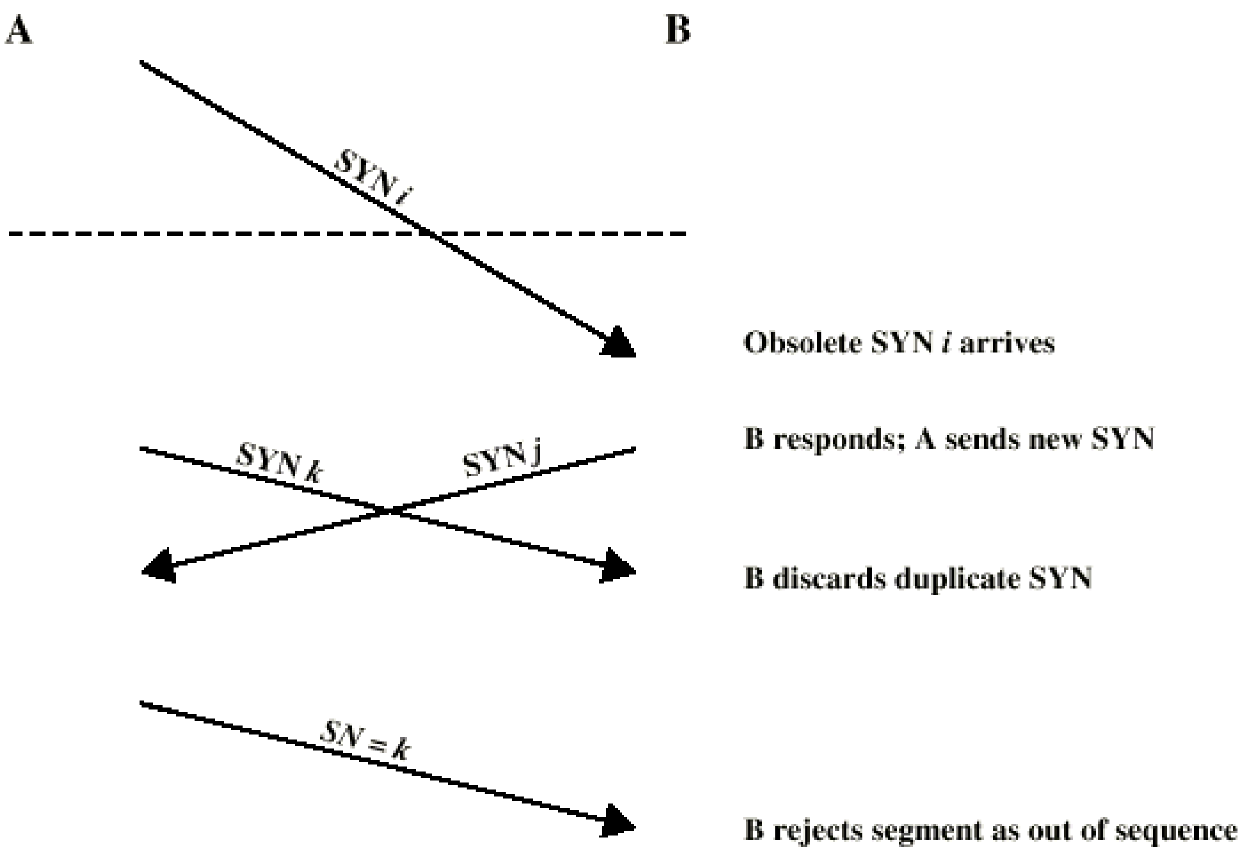
- Problem
  - How to recognize slipped segments from old connection
  - How to recognize duplicated obsolete SYN

2-Way Handshake: Slipped Data Segment

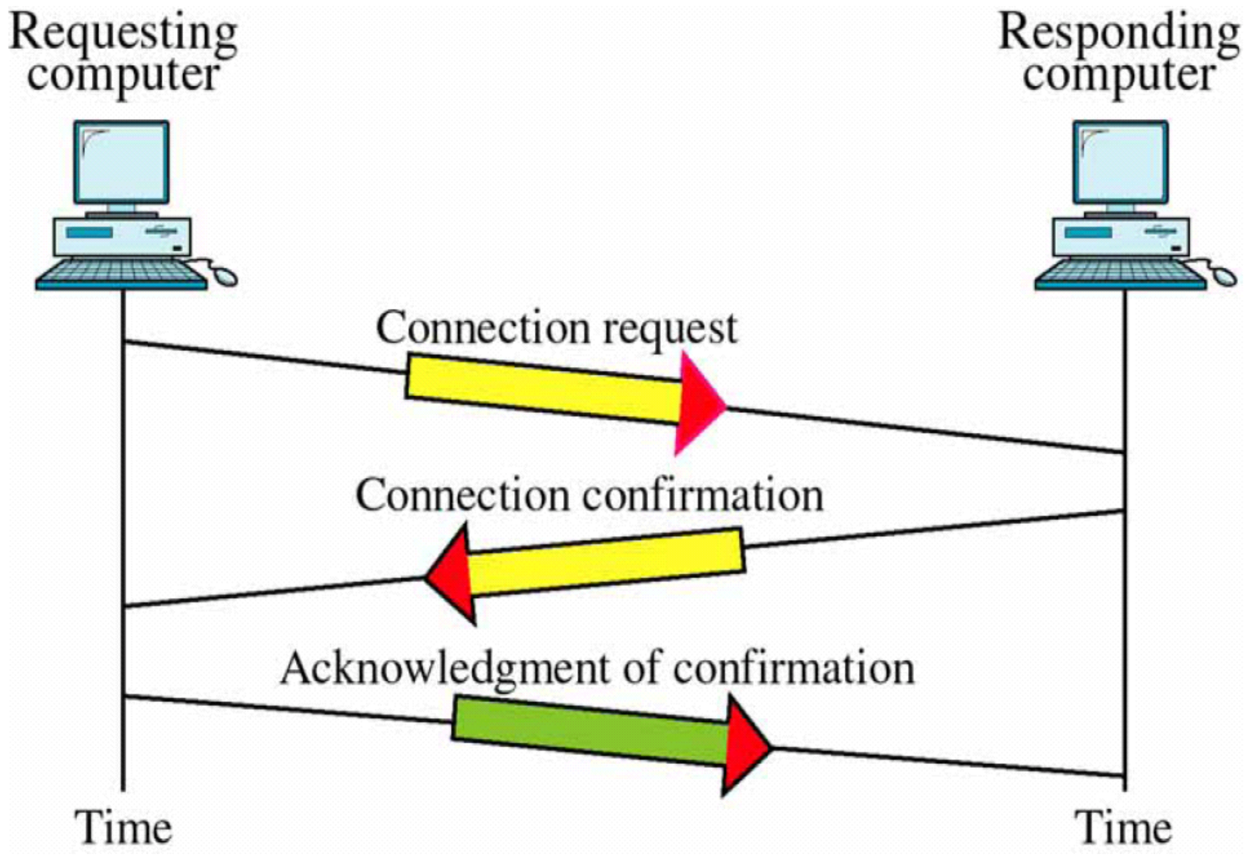


- Handle
  - Start each new connection with a different sequence number far from previous connection
  - The connection request is of the form SYN i+1, where i is the sequence number of the first data segment that will be sent on this connection.
- However:

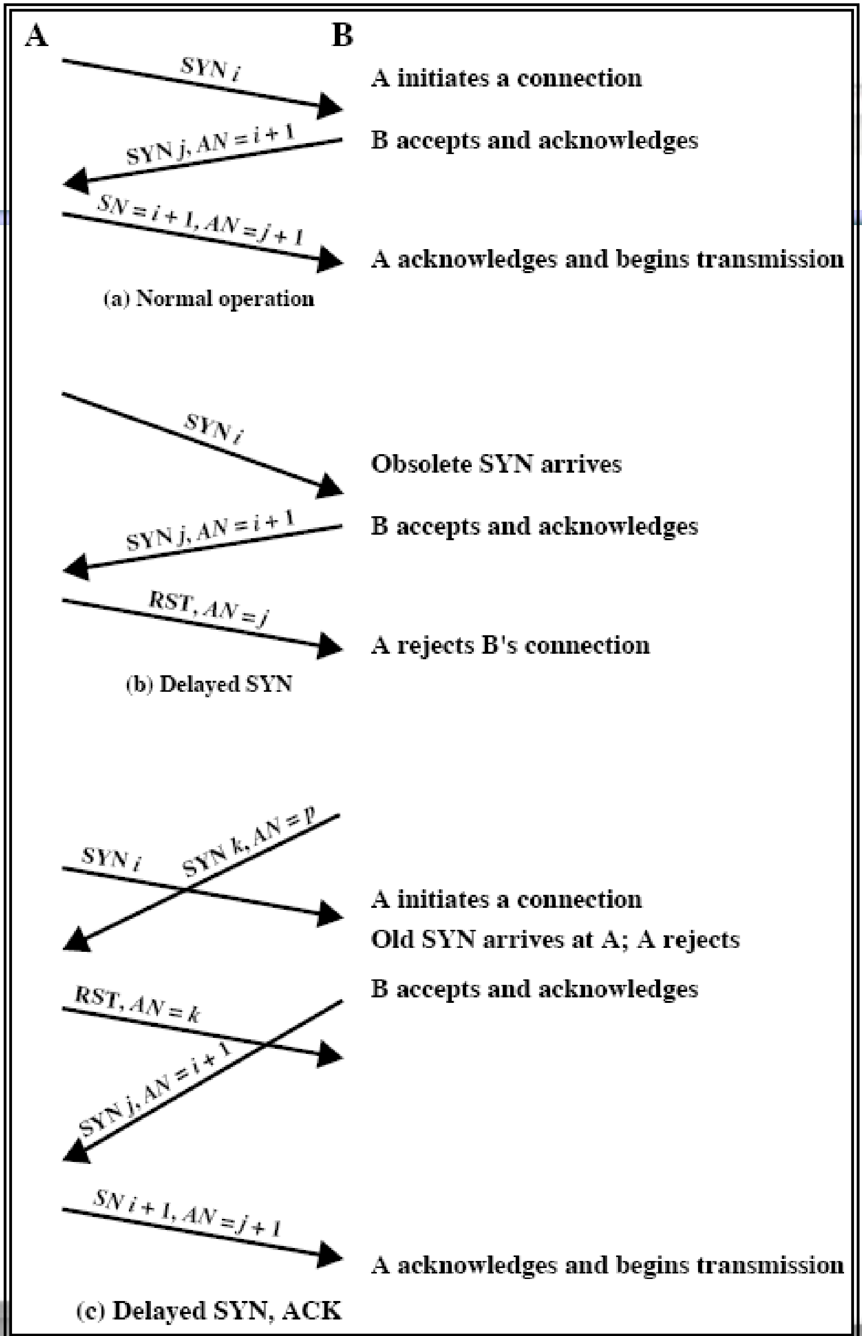
2-Way Handshake: Obsolete SYN



Solution: 3-Way Handshake



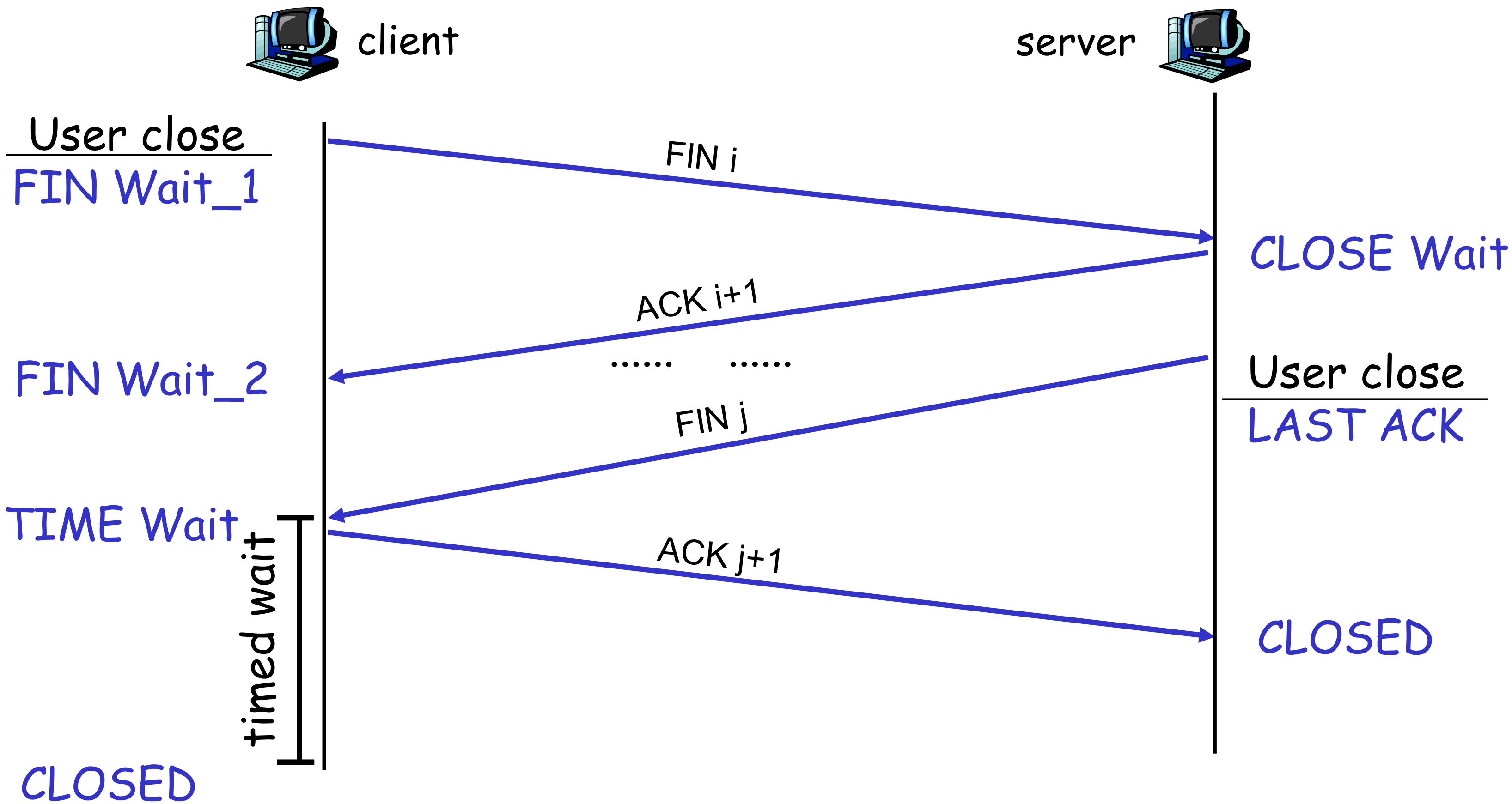
3-Way Handshake: Examples



(6) Connection Termination

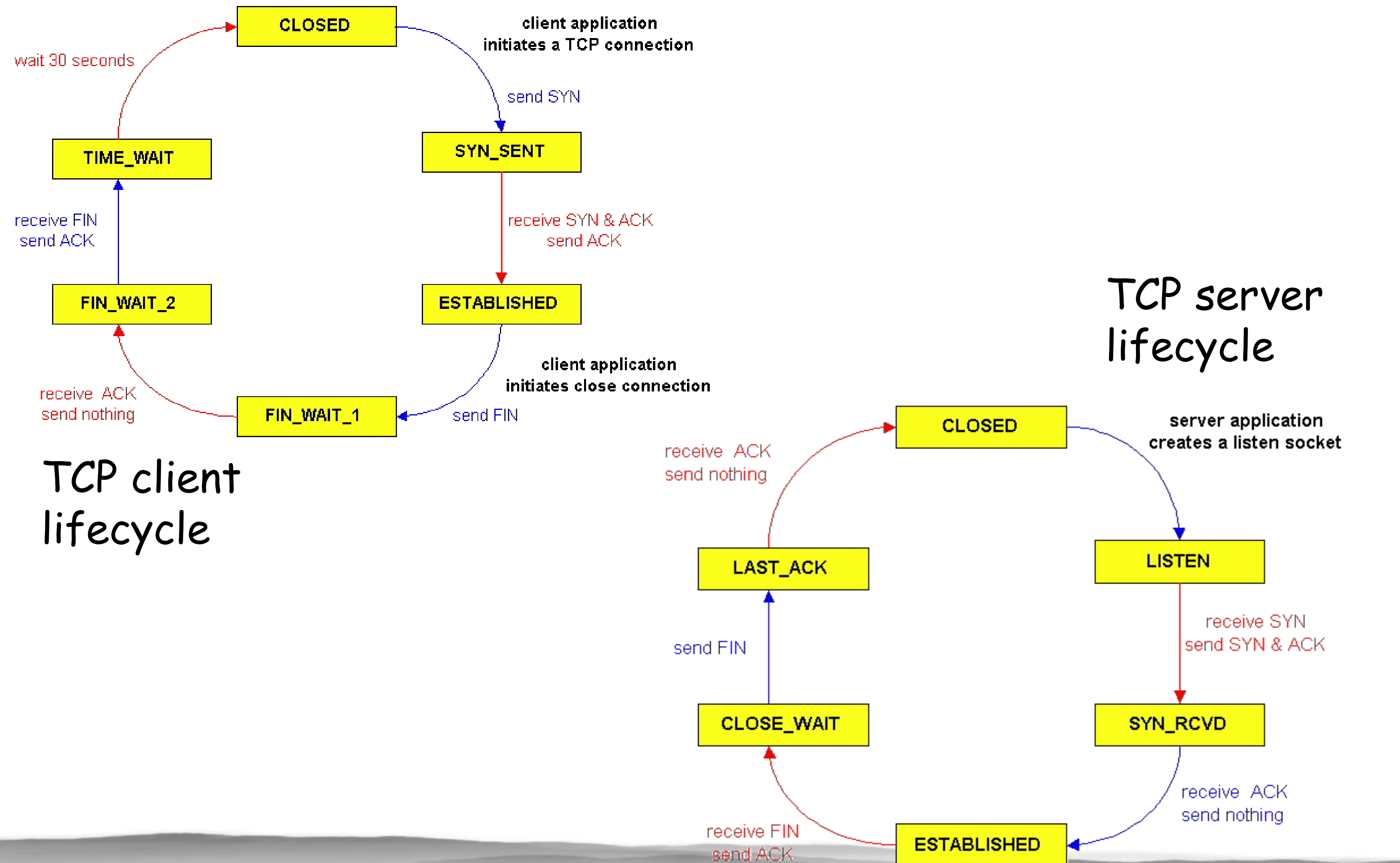
- A problem with 2-way termination
  - Entity in *CLOSE Wait* state sends last data segment, followed by FIN
  - FIN arrives before last data segment
  - Receiver accepts FIN and closes connection
  - Now last data segment lost
- Handle
  - Associate sequence number with FIN
  - Receiver waits for all segments before FIN seq number
  - *ACK* FIN, use 3-way termination

Graceful Close

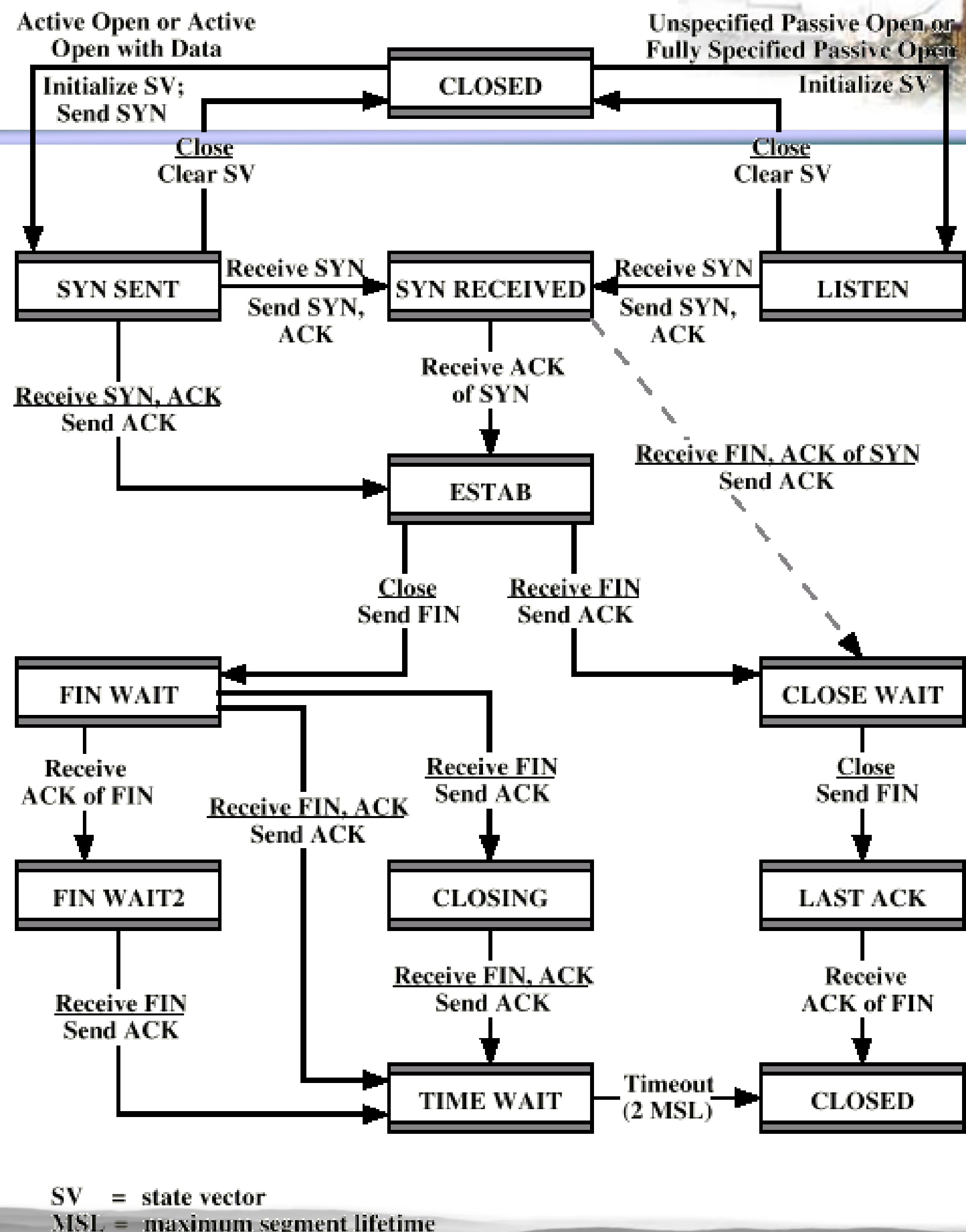


Q: What does TIME Wait stand for?

State Diagram: Client and Server



3-Way Handshake: State Diagram



(7) Crash Recovery

Problem

- If a side restarts, all state info is lost
- Connection is half open now
  - Side that did not crash still thinks it is connected

Handle

- Close connection using Persistence Timer
  - Wait for *ACK* for (timeout) $\times$ (number of retries)
  - When expired, close connection and inform user
- Restarted side sends RST *i* in response to any *i* segment arriving
  - Other side verifies RST *i*, then closes connection
  - Restarted user can reconnect immediately

Summary

- Internet Transport Service
  - Addressing and multiplexing
  - Connection-oriented mechanisms
    - Flow control
    - Connection establishment and termination
    - Reliable sequencing communication
- Dealing with unreliable network service: seven issues
  - Ordered Delivery
  - Retransmission Strategy
  - Duplication Detection
  - Traffic Control
  - Connection Establishment: 3-way handshake
  - Connection Termination
  - Crash Recovery