Transport Protocols

Transport Protocols

- 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
  - e.g. reliable packet switched network using X.25
  - e.g. frame relay using LAPF control protocol
  - e.g. 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
  - 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
  - Know address ahead of time
    - e.g. collection of network device stats
  - Well known addresses
  - 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)
- 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
  - Clumsy
  - Multiplexed connections are controlled on aggregate flow

Coping with Flow Control Requirements (2)

- Use fixed sliding window protocol
  - —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

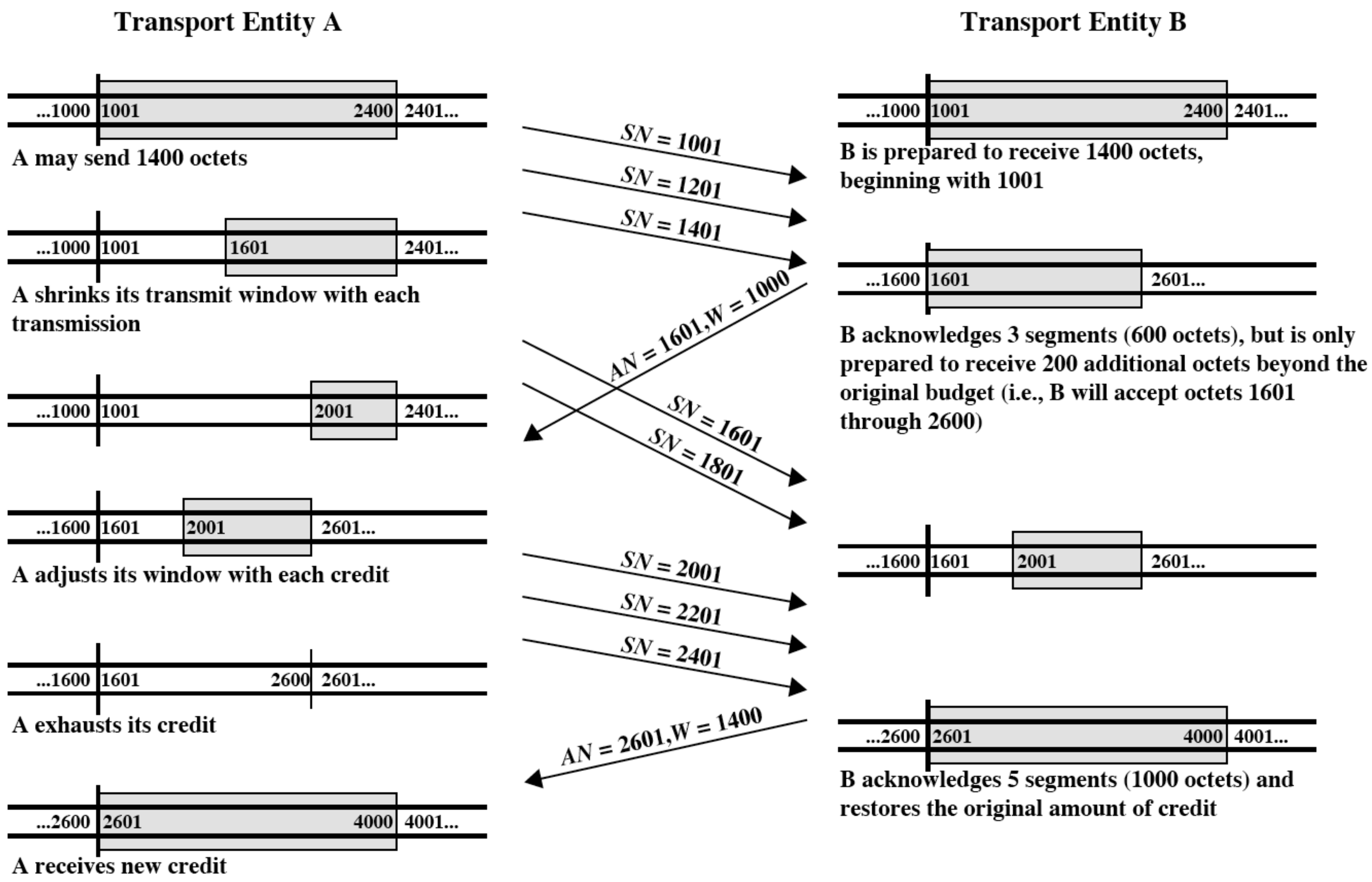
Credit Scheme

- 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

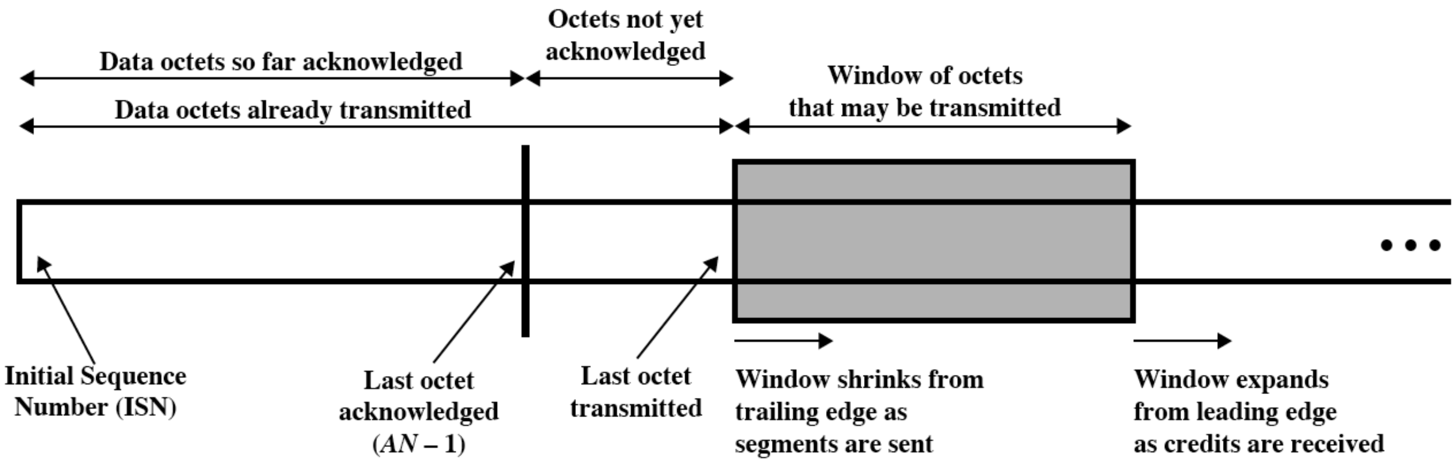
Use of Header Fields

- When sending, seq number is that of first octet in segment
- ACK includes
  - ack number *AN=i*,
  - window number *W=j*
- All octets through seq. num. *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*

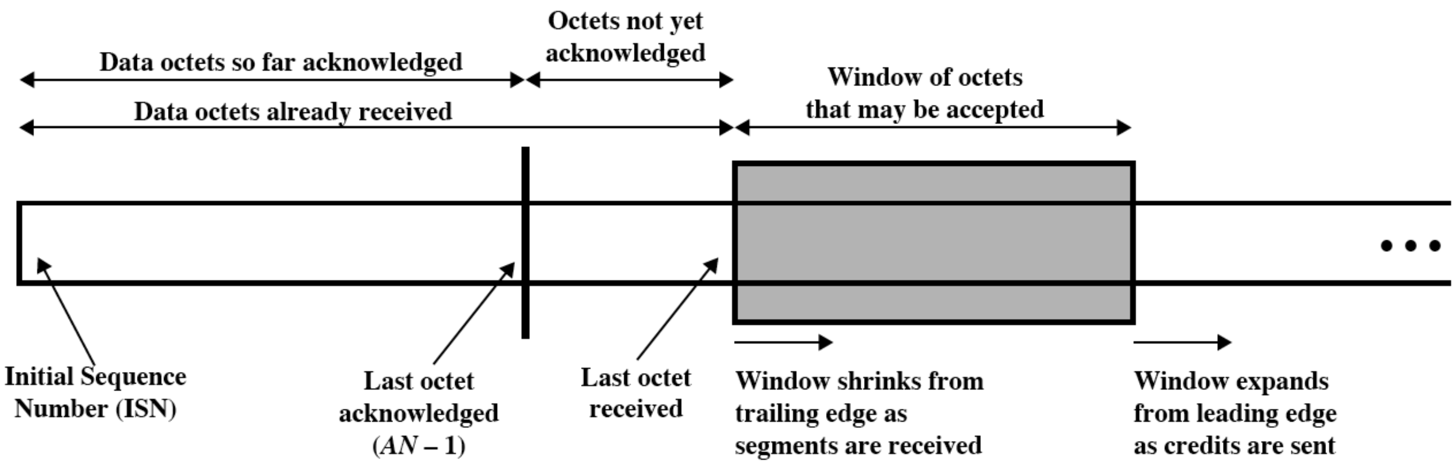
Credit Allocation



Sending and Receiving Perspectives



(a) Send sequence space

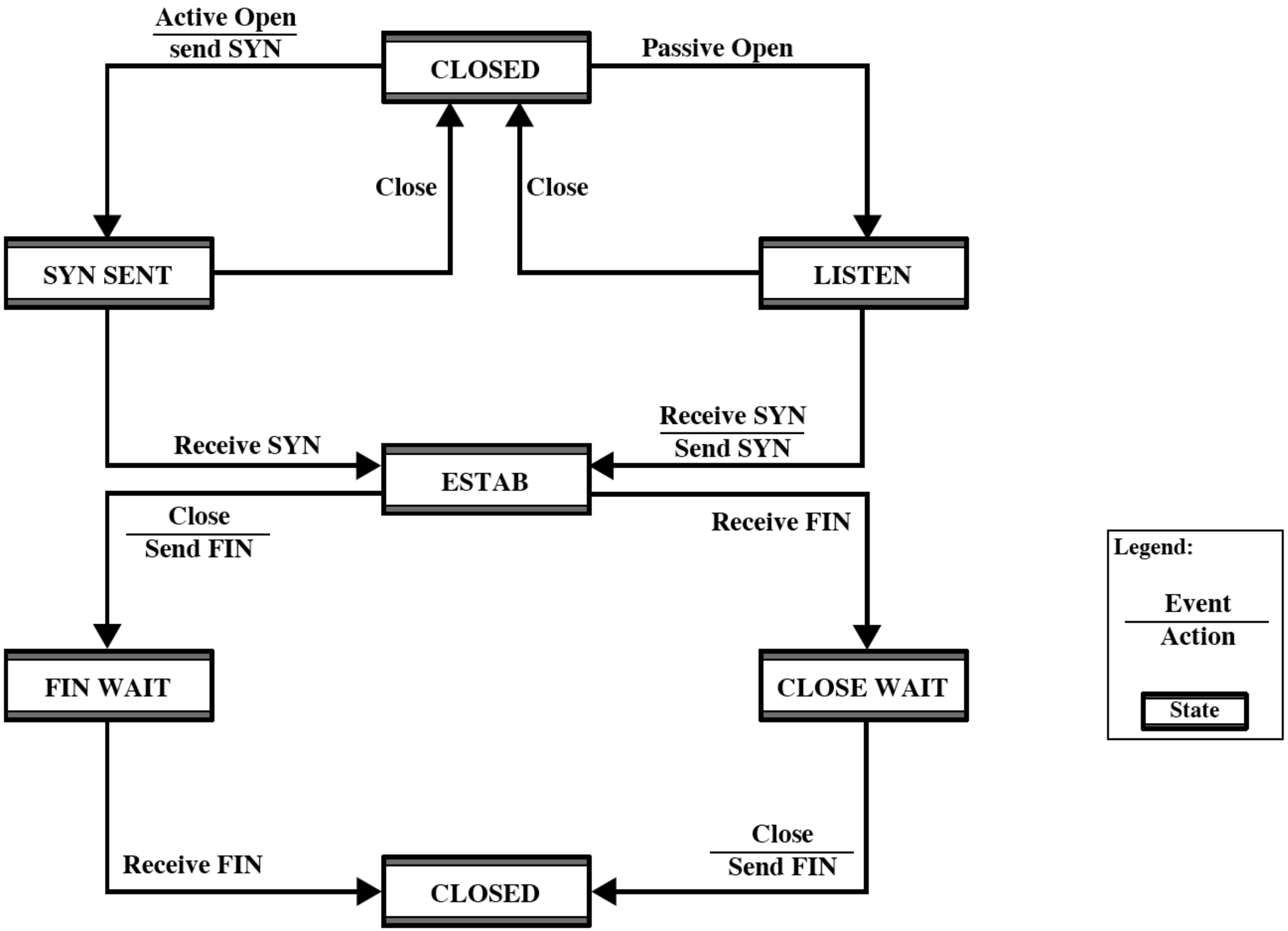


(b) Receive sequence space

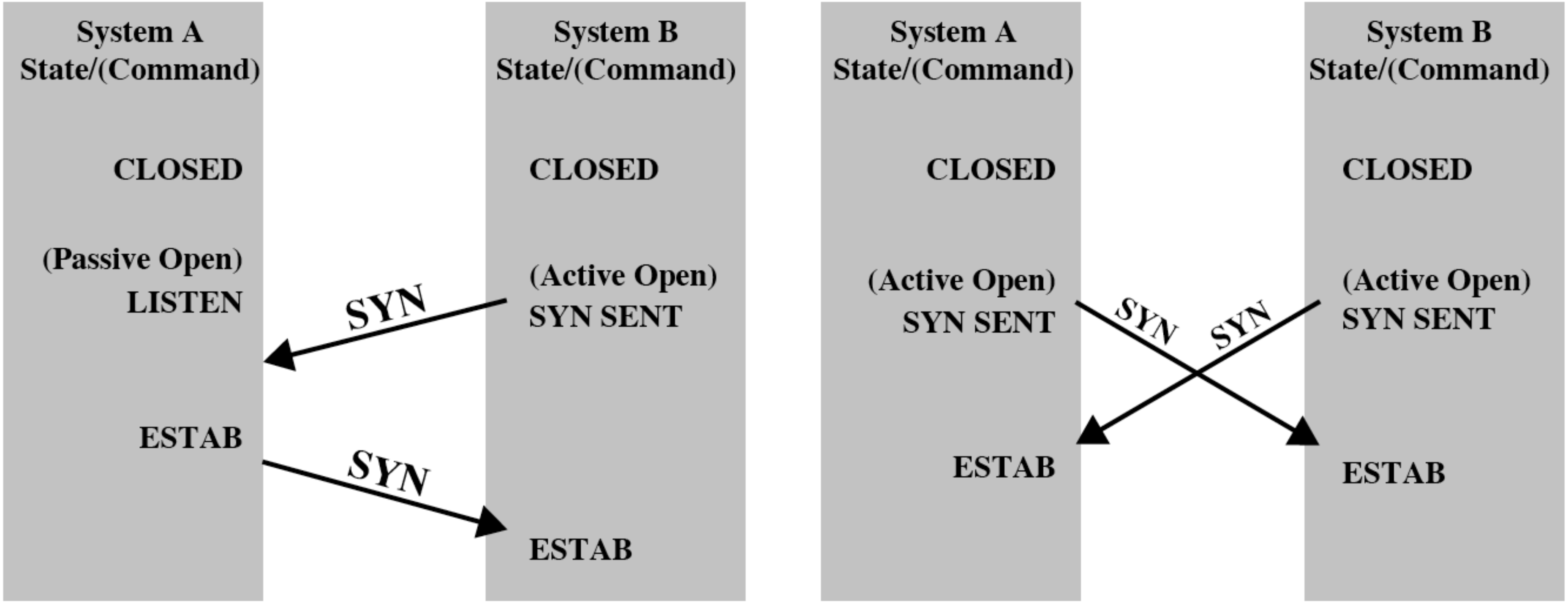
Establishment and Termination

- Allow each end to now the other exists
- Negotiation of optional parameters
- Triggers allocation of transport entity resources
- By mutual agreement

Connection State Diagram



Connection Establishment



(a) Active/Passive Open

(b) Active/Active Open

If TS is Not Listening

- Three things can happen
  - Reject with RST (Reset)
  - Queue request until matching open issued
  - Signal transport service (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

Graceful Degradation: Consider the 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

Now consider 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

- E.g.
  - 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

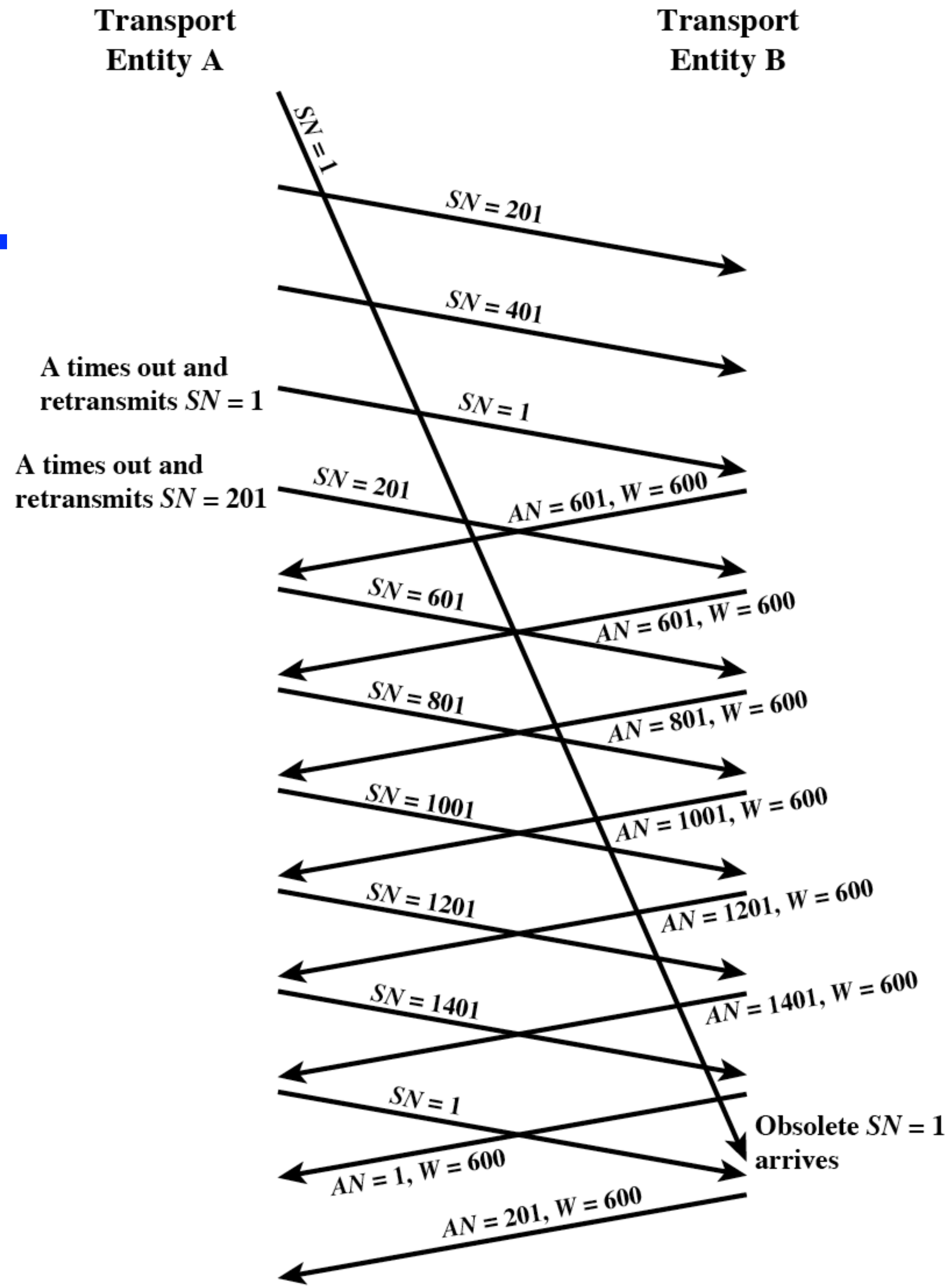
Timer Value

- 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

Incorrect Duplicate Detection



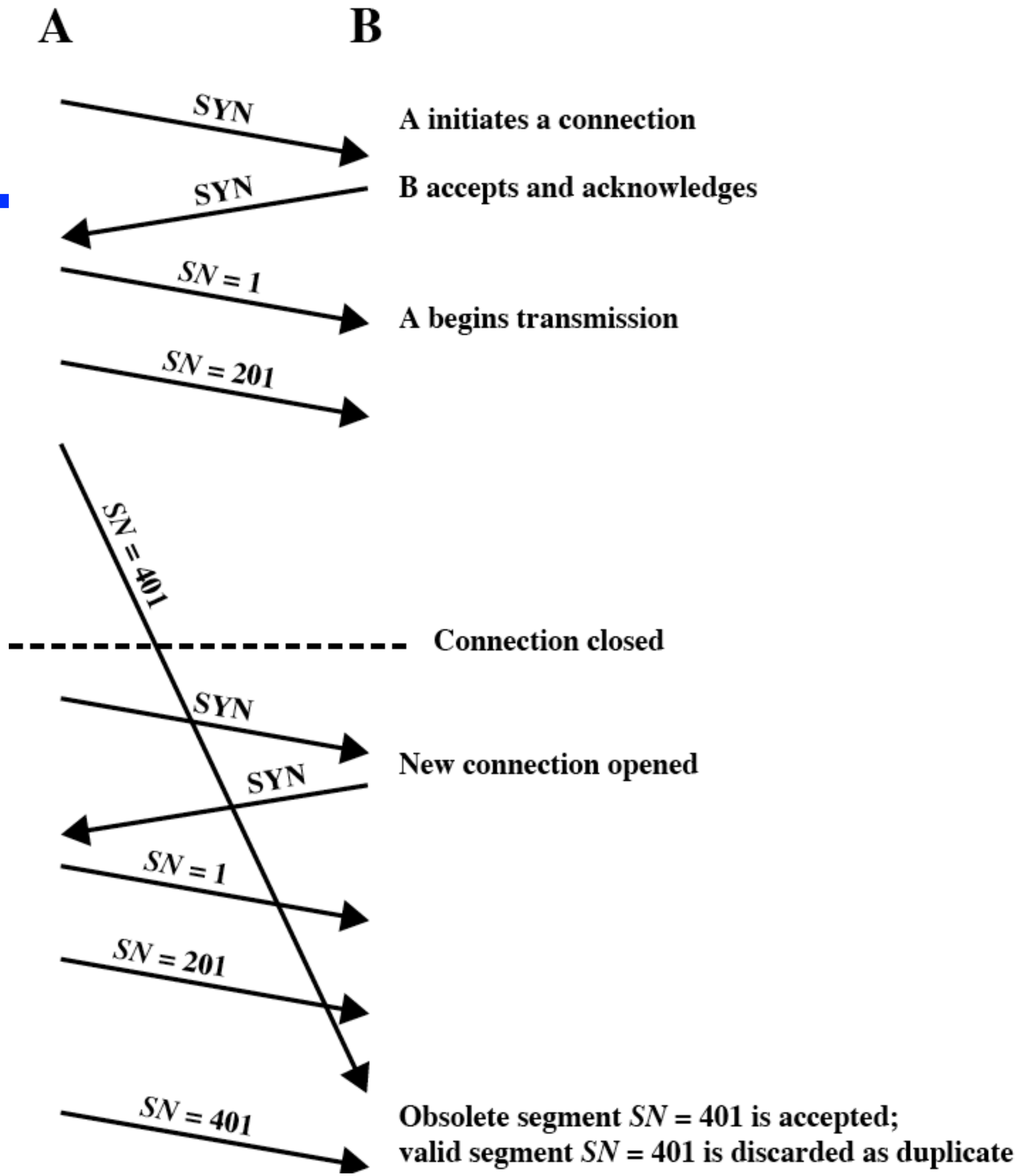
Flow Control

- 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

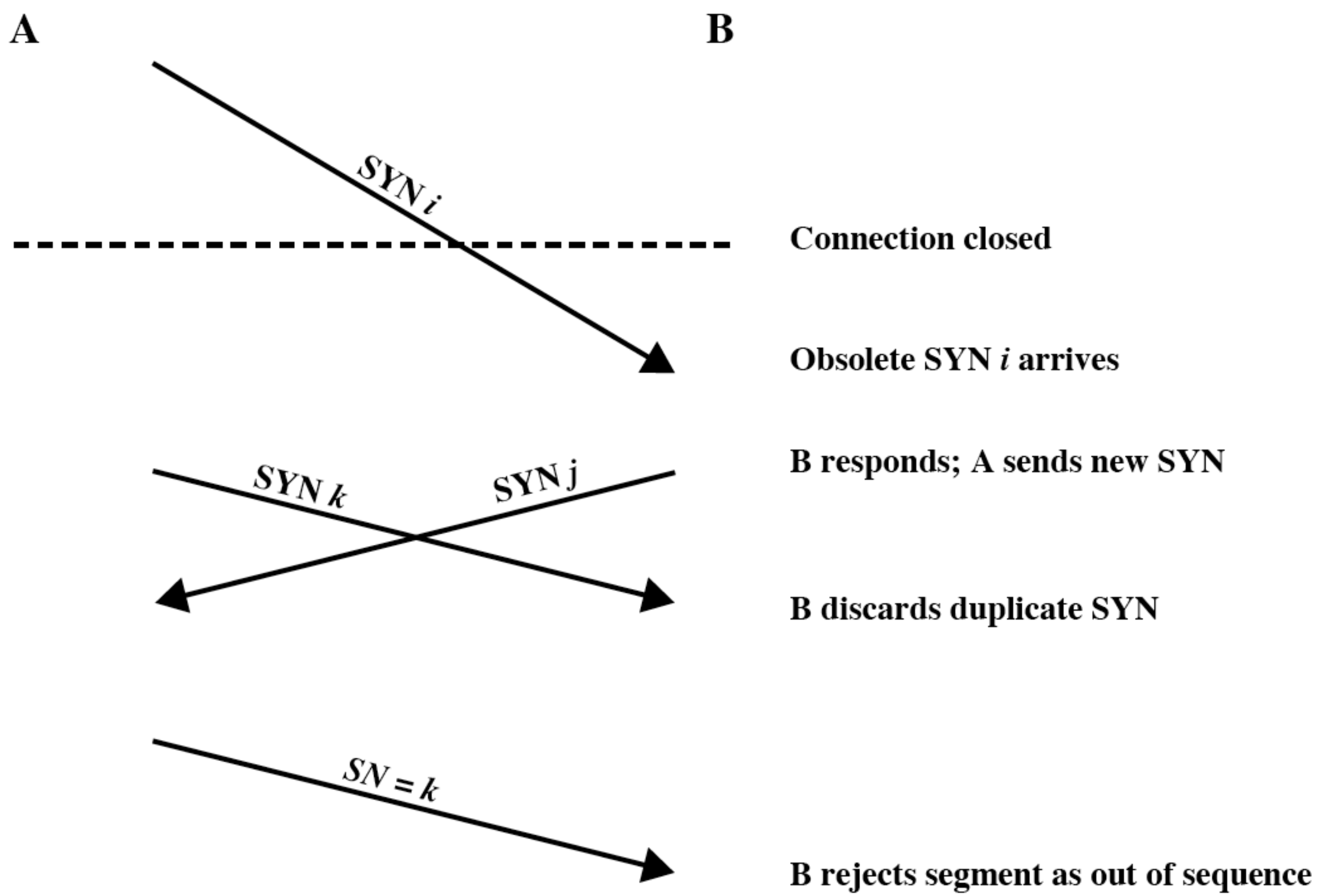
Connection Establishment

- 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

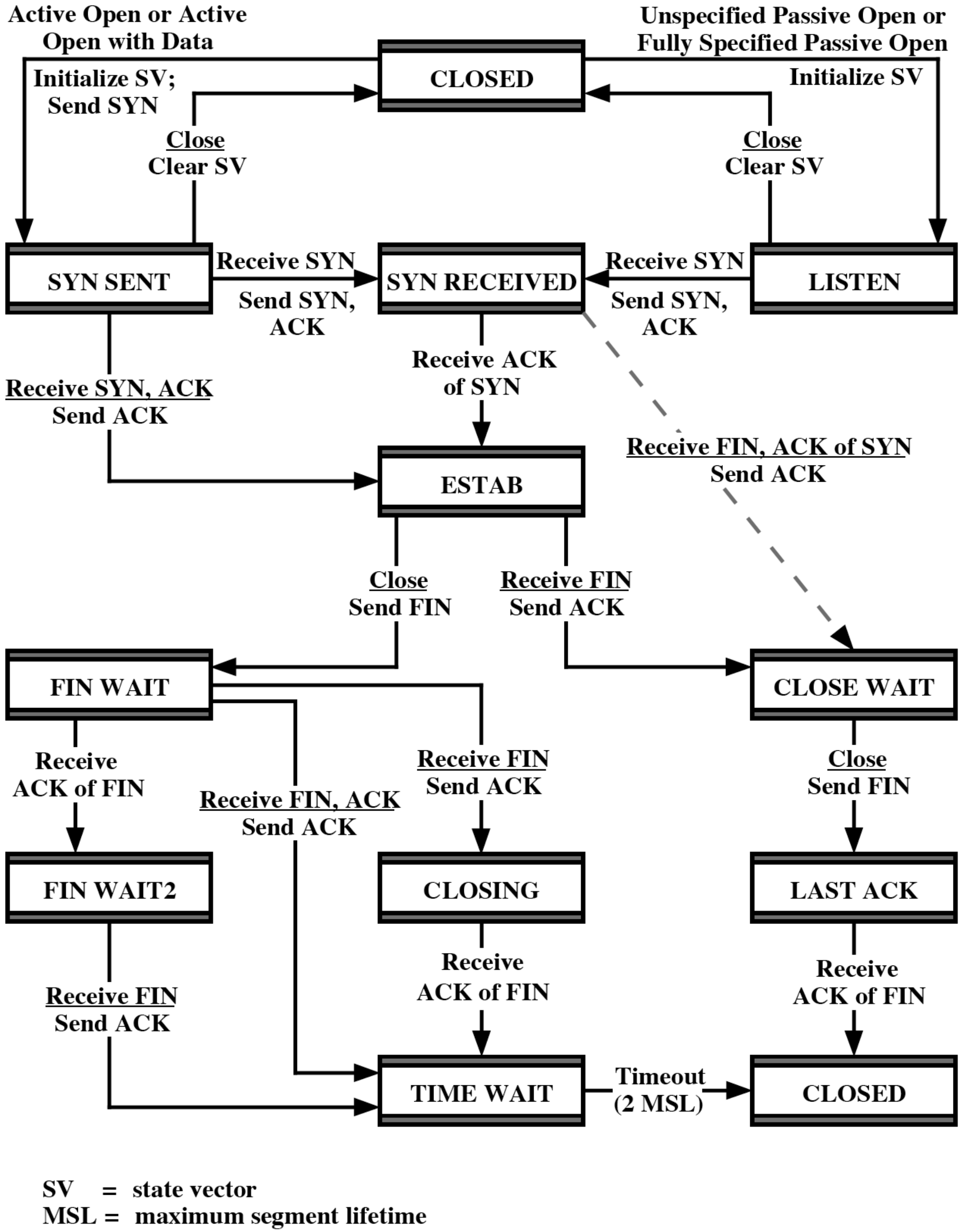
Two Way Handshake: Obsolete Data Segment



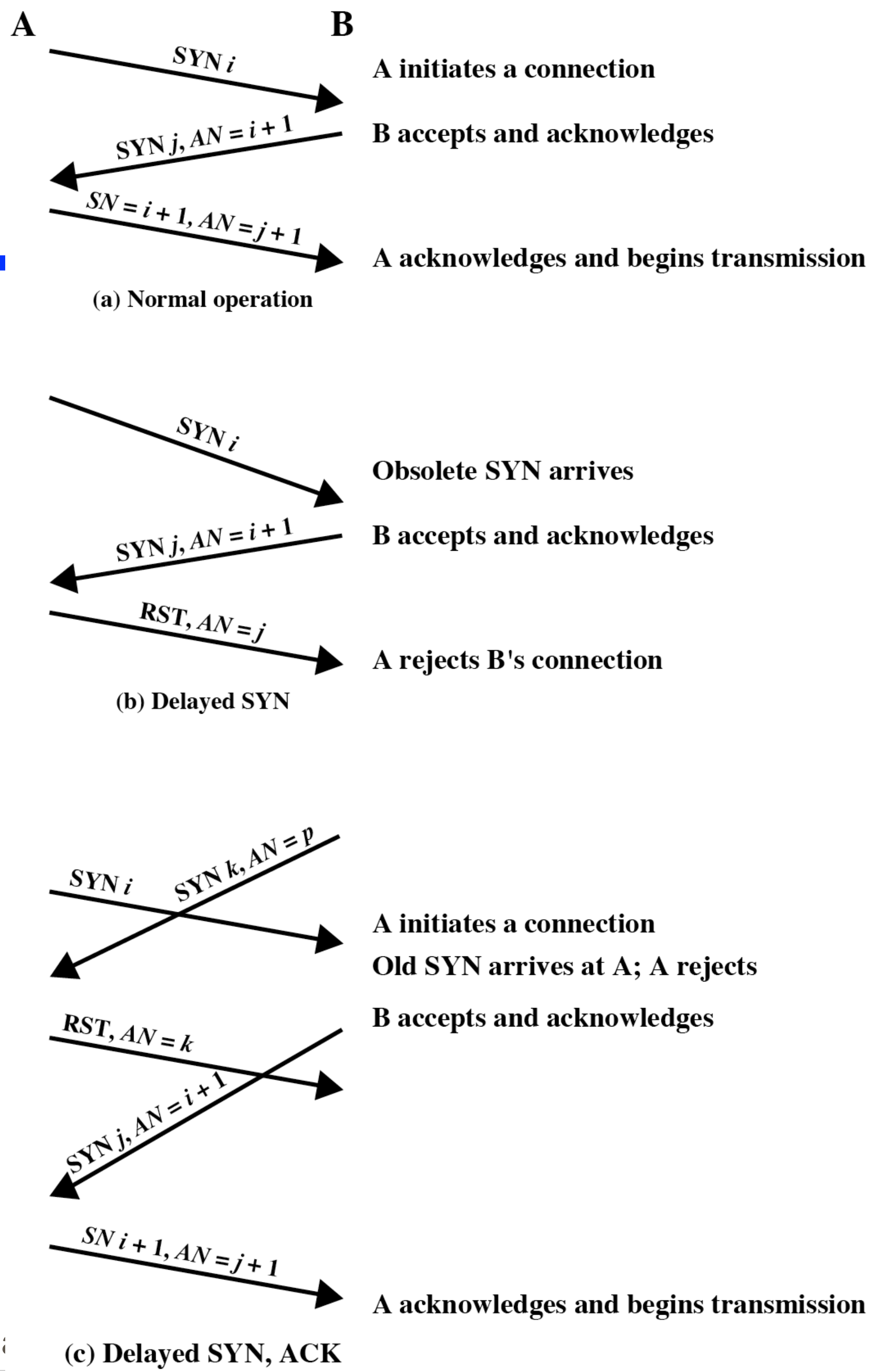
Two Way Handshake: Obsolete SYN Segment



Three Way Handshake: State Diagram



Three Way Handshake: Examples



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

Graceful Close

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

Failure Recovery

- 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

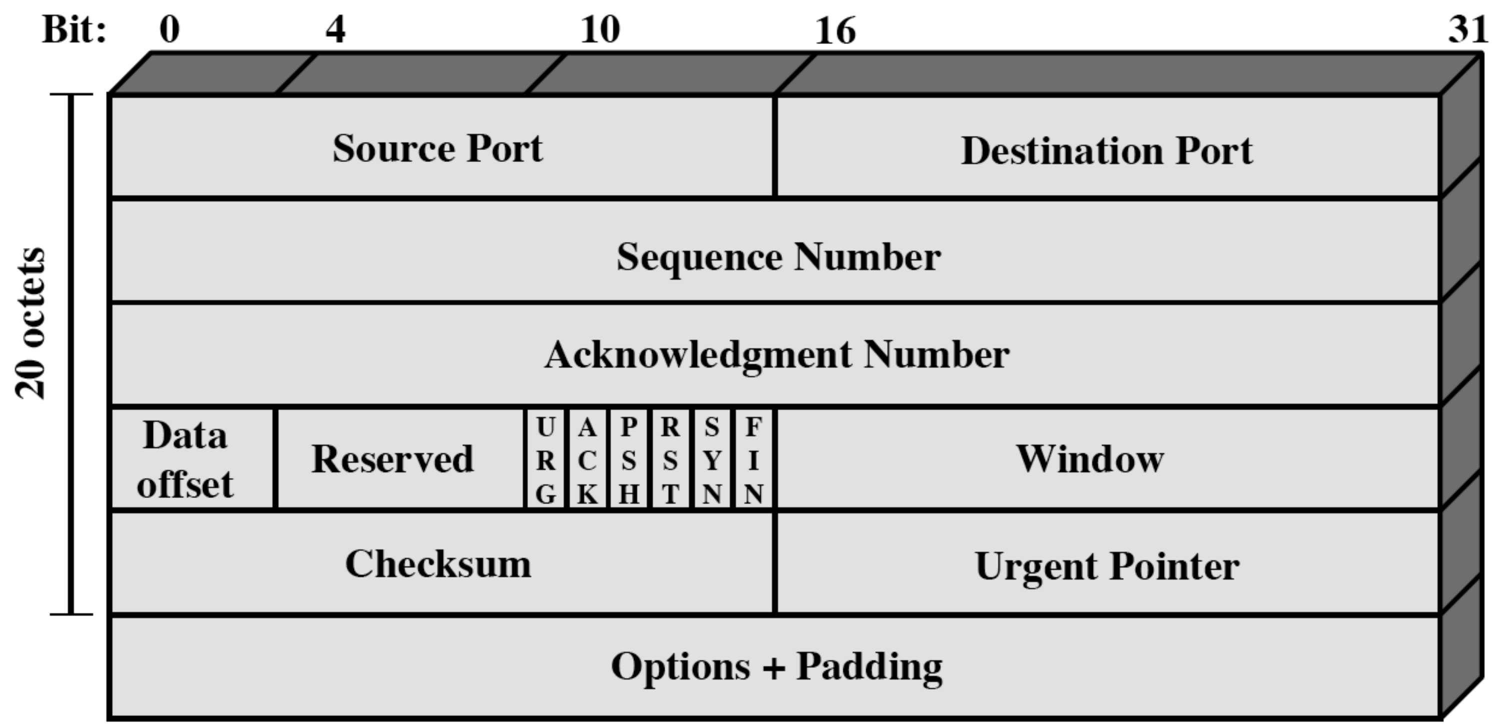
TCP & UDP

- Transmission Control Protocol (TCP)
  - 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
Acknowledgment 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^{23}
  - —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

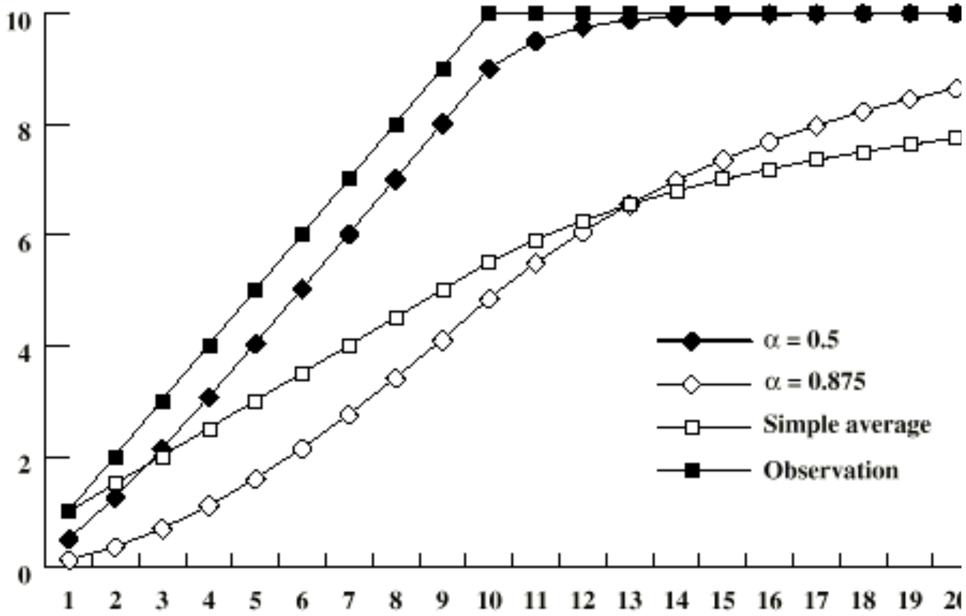
Acknowledgement

- Immediate
- Cumulative

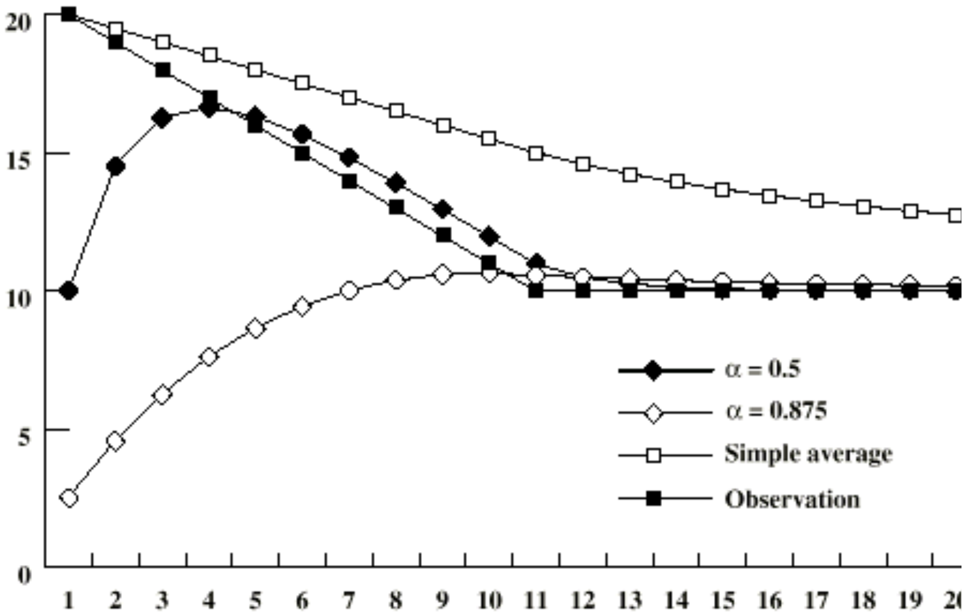
Congestion Control

- 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)

Use of Exponential Averaging

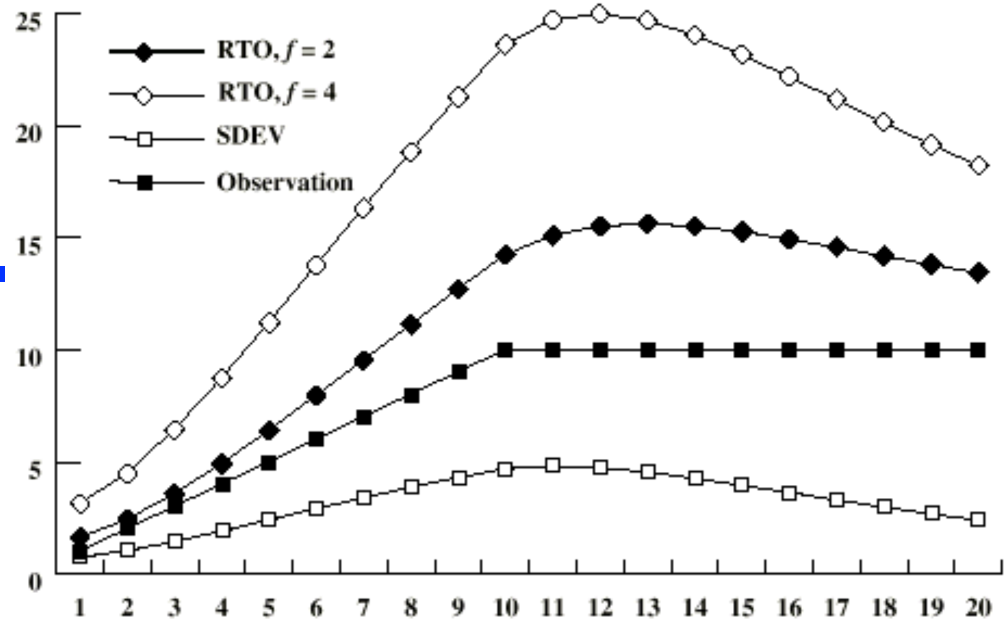


(a) Increasing function

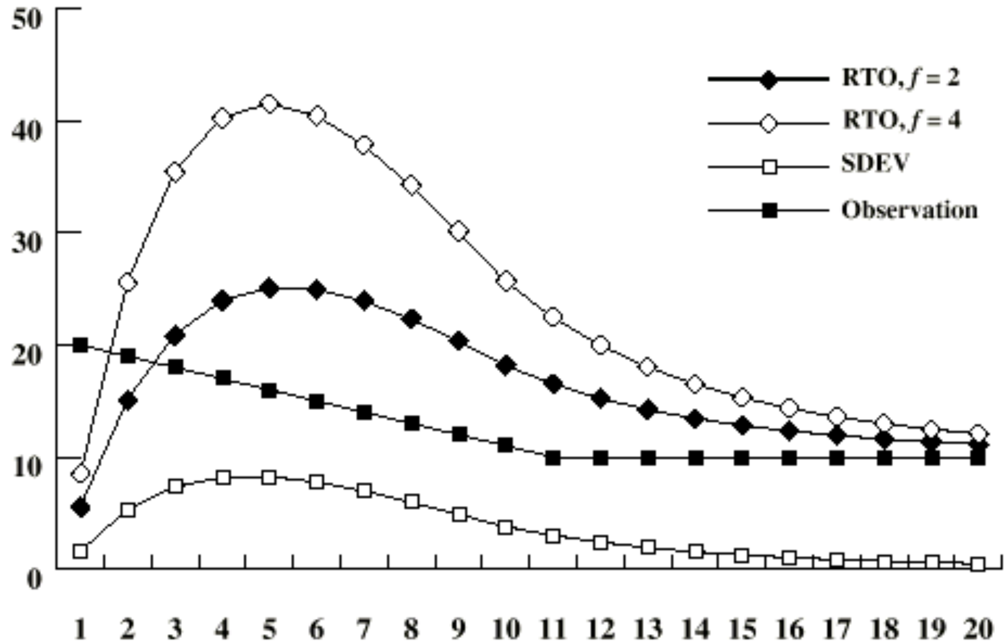


(b) Decreasing function

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
- 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

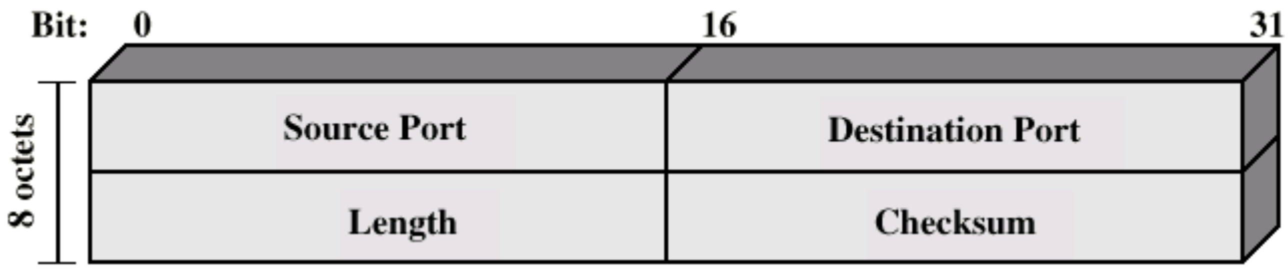
UDP

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

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

Summary

- connection-oriented network and transport mechanisms and services
- TCP services, mechanisms, policies
- TCP congestion control
- UDP