Multiplexing

- downward 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 (upward multiplexing)
  - 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
  - note that these problems are independent of the network level flow control issues
- Results in transport entity 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
  - Already discussed in the context of HDLC and X25
  - 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

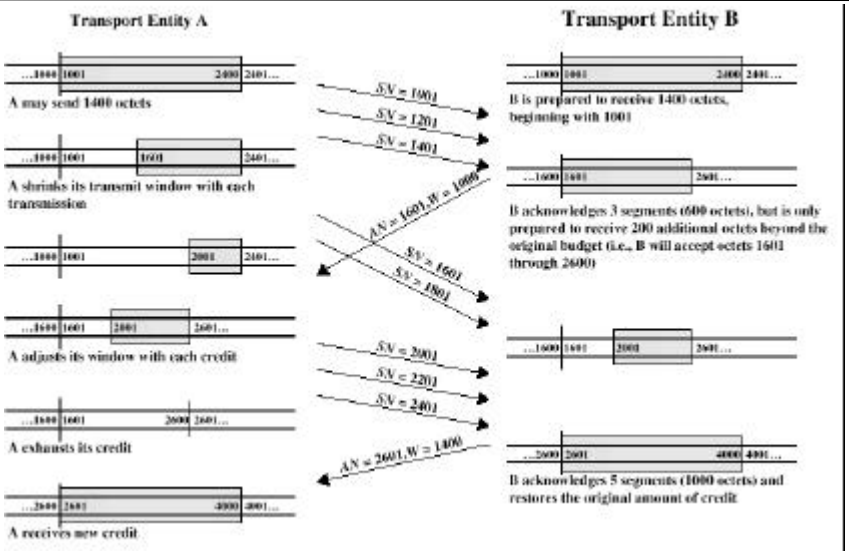
Credit Scheme

- provides 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

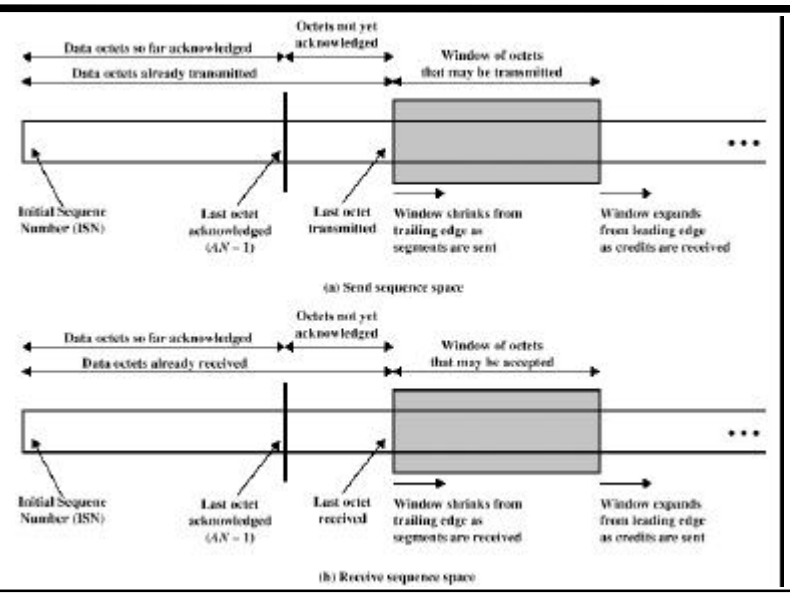
Use of Header Fields

- When sending, seq number is that of first octet in segment
- ACK includes acknowledgement no AN=i, credit window 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 up to i+j-1

Credit Allocation



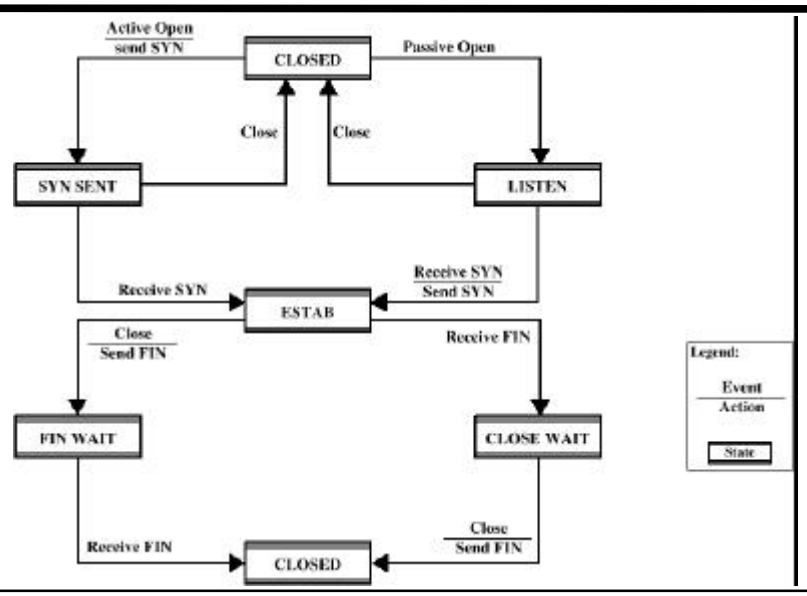
Sending and Receiving Perspectives



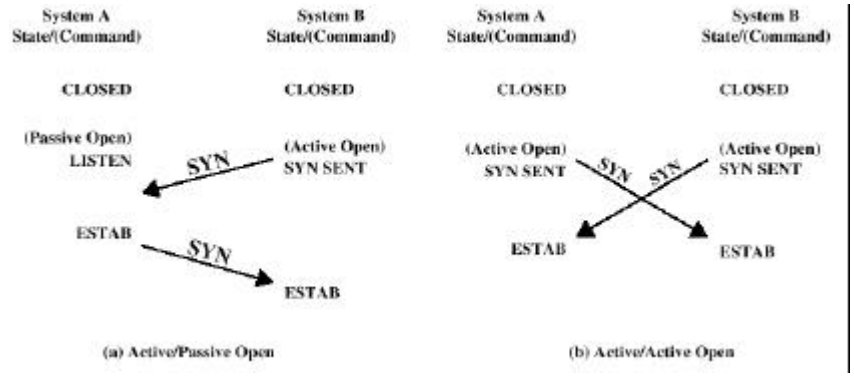
Establishment and Termination

- Allow each end to know the other exists
- Negotiation of optional parameters
  - maximum segment size, maximum window size
  - quality of service
- Triggers allocation of transport entity resources
  - buffer space, entry in connection table
- Connection establishment is by mutual agreement

Connection State Diagram



Connection Establishment



What about receiving a SYN when Not Listening (CLOSED)

- 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

- can be initiated by either or both sides
- By mutual agreement
- Either 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

- E.g.
  - internet using IP,
  - IEEE 802.3 using unacknowledged connectionless LLC
- Segments may get lost
- Segments may arrive out of order
  - we do assume that a delivered segment is undamaged

Problems

- Ordered Delivery
- Retransmission strategy
- Duplication detection
- Flow control (again!)
- Connection establishment
- Connection termination
- Crash recovery

Ordered Delivery

- Segments may arrive out of order
- Number segments sequentially
- TCP
  - numbers each octet sequentially (32 bits)
  - Segments are numbered by the first octet number in the segment

Retransmission Strategy

- Segment fails to arrive
- Transmitter does not know of failure
  - Receiver must acknowledge successful receipt of received segments
  - Use cumulative acknowledgement
    - rather than acknowledging each individually
- Time out waiting for ACK triggers re-transmission
  - but at what delay value?

Timer Value

- Fixed timer
  - Based on understanding of network behavior
  - does 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

(Transport layer Timers)

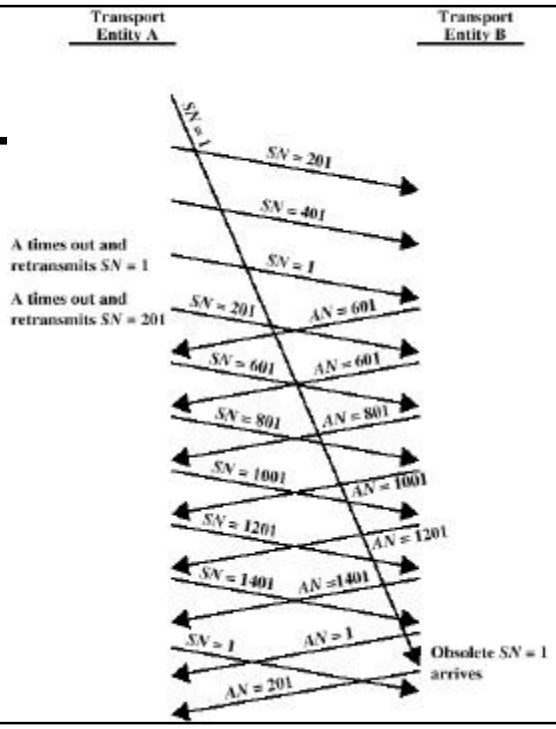
- Retransmission timer
  - how long to wait before retransmitting an unack'd segment
- Reconnection timer
  - minimum time between closing a connection, and re-opening it
- Window timer
  - max time between ACK and credit segments
- retransmit-SYN timer
  - time between attempts to open a connection
- Persistence timer
  - how long to wait before aborting whn to segments are acknowledged
- Inactivity timer
  - how long to wait when no segments are received

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

Failure caused by octet numbering wraparound



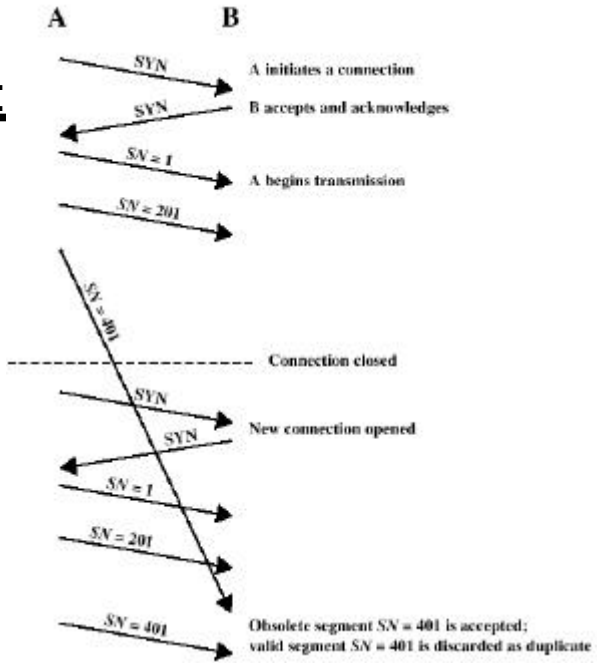
Flow Control

- Credit allocation is robust here
  - if an ACK/CREDIT is lost, future ones will resynchronise
- 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
  - for example re-transmit 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
  - Segments from old connections
  - Start segment numbers far removed from previous connection
    - Use SYN I (segment number)
    - Need ACK to include i
    - Three Way Handshake

Two Way Handshake: Obsolete Data Segment



Two Way Handshake: Obsolete SYN Segment