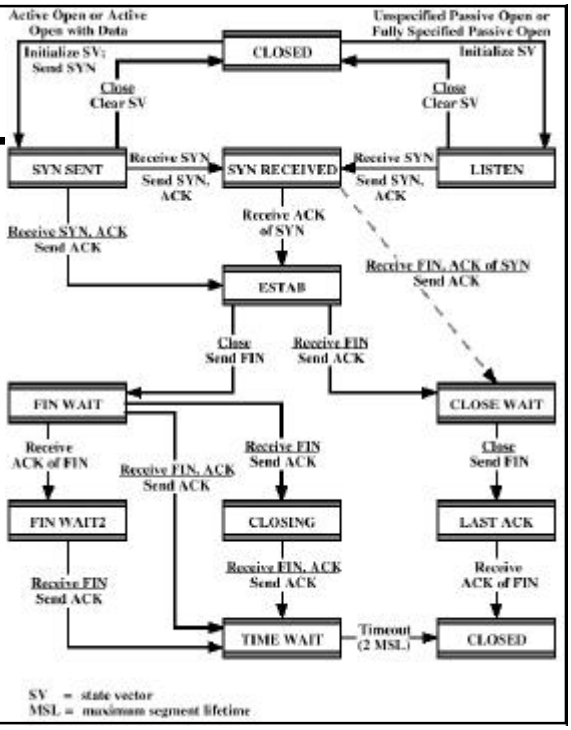
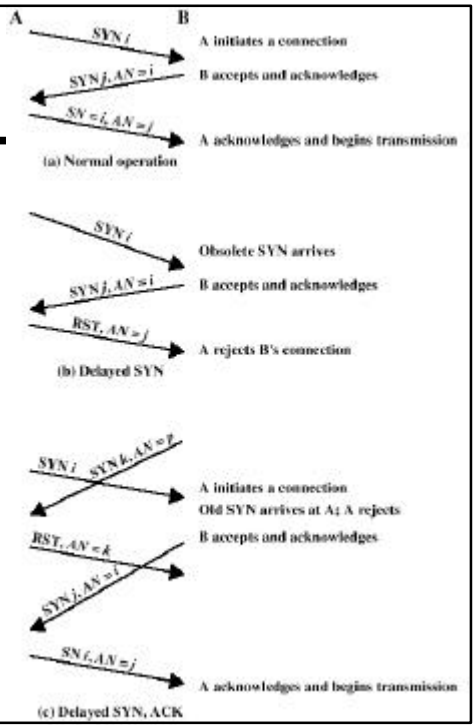
Three Way Handshake: State Diagram



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
  - (MSL)

Crash 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 (reset) in response to any i segment arriving
- User must decide whether to reconnect
  - Problems with lost or duplicate data

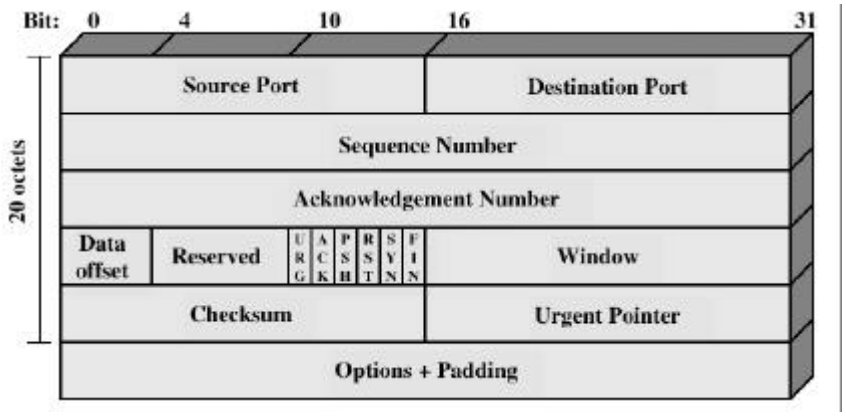
TCP & UDP

- 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



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)

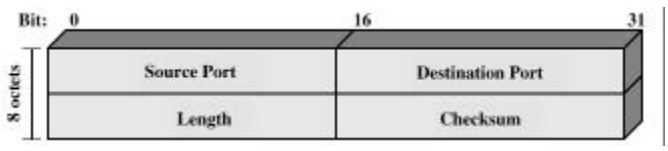
UDP

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

UDP Uses

- Inward data collection
- Outward data dissemination
- Request-Response
- Real time applications
- specifically:
  - directory services (name server)
  - network time protocol (synchronising time across machines)

UDP Header



HTTP introduction

- HTTP overview
- client, server, proxy, user agent, URL, cache, stateless protocol and how they apply to HTTP
- HTTP overall operation
- How HTTP runs. This is well described in section 1.4 of RFC 2616

HTTP messages

- HTTP has two types of messages, requests and responses. Requests can be simple or full, as can responses.
- Simple requests and responses originate with HTTP 0.9, but are now discouraged.
- Full requests and responses consist of a header line, followed by the entity body. The header line can be a general header or an entity header or either a request header (full request) or a response header (full response).

Further HTTP info:

- Stallings section 19.4 (pp726-739), Tanenbaum 681-695 (bit thin)
- RFC 2616
  http://www.w3.org/Protocols/rfc2616/rfc2616

SMTP

- Basic SMTP operation
- SMTP overview
- MIME

Further SMTP information

- Stallings pp 711-726, Tanenbaum 643-661
- on the www:
  - SMTP tutorial
  - http://www.rad.com/networks/1998/smtp/smtp.htm
  - Internet draft
  - http://www.ietf.org/internet-drafts/draft-ietf-drums-smtpupd-13.txt