TCP Connection Management

- Setup
  - asymmetric 3-way handshake
- Transfer
  - sliding window; data and acks in both directions
- Teardown
  - symmetric 2-way handshake
- Client-server model
  - initiator (client) contacts server
  - listener (server) responds, provides service
Three-Way Handshake

- Opens both directions for transfer

Active participant (client)

<table>
<thead>
<tr>
<th>SYN, SequenceNum = x</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN + ACK, SequenceNum = y</td>
</tr>
<tr>
<td>ACK, Acknowledgment = x + 1</td>
</tr>
<tr>
<td>ACK, Acknowledgment = y + 1</td>
</tr>
</tbody>
</table>

Passive participant (server)

TCP Transfer

- Connection is bi-directional
  - acks can carry response data

(client) (server)

<table>
<thead>
<tr>
<th>Seq = x + MSS; Ack = y + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq = x + 2*MSS; Ack = y + 1</td>
</tr>
<tr>
<td>Seq = y + MSS; Ack = x + 2*MSS + 1</td>
</tr>
<tr>
<td>Seq = x + 3*MSS; Ack = y + MSS + 1</td>
</tr>
</tbody>
</table>
TCP Connection Teardown

Symmetric: either side can close connection

Web server

FIN

data, ACK

data, ACK

FIN

ACK

ACK

Web browser

Half-open connection; data can be continue to be sent

Can reclaim connection right away
(must be at least 1MSL after first FIN)

Can reclaim connection after 2 MSL

TCP State Transitions

CLOSED

LISTEN

SYN_RCVD

SYN_SYN_ACK

SYN_SENT

ESTABLISHED

FIN_WAIT_1

FIN_WAIT_2

CLOSING

TIME_WAIT

CLOSED

TIME_WAIT

CLOSE_WAIT

LAST_ACK

CLOSED

Active open/SYN

Close

Send/SYN

SYN + ACK/ACK

ACK

FIN/ACK

FIN_ACK

ACK

Timeout after two segment lifetimes

ACK

ACK

ACK

ACK

ACK
TCP Connection Setup, with States

Active participant (client)  Passive participant (server)
SYN_SENT  LISTEN
SYN, SequenceNum = x  SYN_RCVD
SYN + ACK, SequenceNum = y  ACK, Acknowledgment = x + 1
ACK, Acknowledgment = y + 1
+data

TCP Connection Teardown

Web server  Web browser
FIN_WAIT_1  FIN
FIN_WAIT_2  ACK
TIME_WAIT  FIN
CLOSED  LAST_ACK
...  ACK
CLOSED  CLOSED
The TIME_WAIT State

- We wait 2MSL (two times the maximum segment lifetime of 60 seconds) before completing the close
- Why?
- ACK might have been lost and so FIN will be resent
- Could interfere with a subsequent connection

TCP Handshake in an Uncooperative Internet

- TCP Hijacking
  - if seq # is predictable, attacker can insert packets into TCP stream
  - many implementations of TCP simply bumped previous seq # by 1
  - attacker can learn seq # by setting up a connection
- Solution: use random initial sequence #'s
  - weak form of authentication

![TCP Handshake Diagram](image-url)
TCP Handshake in an Uncooperative Internet

- TCP SYN flood
  - server maintains state for every open connection
  - if attacker spoofs source addresses, can cause server to open lots of connections
  - eventually, server runs out of memory

TCP SYN cookies

- Solution: SYN cookies
  - Server keeps no state in response to SYN; instead makes client store state
  - Server picks return seq # $y = \pi(x)$ that encrypts $x$
  - Gets $\pi(x)$ from sender; unpacks to yield $x$
**IP Fragmentation**

- Both TCP and IP fragment and reassemble packets. Why?
  - IP packets traverse heterogeneous nets
  - Each network has its own max transfer unit
    - Ethernet ~ 1400 bytes; FDDI ~ 4500 bytes
    - P2P ~ 532 bytes; ATM ~ 53 bytes; Aloha ~ 80 bytes
  - Path is transparent to end hosts
    - can change dynamically (but usually doesn’t)
- IP routers fragment; hosts reassemble

**Fragmentation Example**

- Different networks may have different frame limits (MTUs)
  - Ethernet 1.5K, FDDI 4.5K
- Don’t know if packet will be too big beforehand
  - IPv4: fragment on demand and reassemble at destination
  - IPv6: network returns error message so host can learn limit
### Fragment Fields

- **Fragments** identified by (src, dest, id)
  - Offset gives start, length
- **Flags**
  - More Fragments (MF)
  - Don’t Fragment (DF)

### Fragment Considerations

- **Relating fragments to original datagram provides:**
  - Tolerance of loss, reordering and duplication
  - Ability to fragment fragments
- **Consequences of fragmentation:**
  - Loss of any fragments causes loss of entire packet
    - possibility of congestion collapse!
  - Need to time-out reassembly when any fragments lost
  - Complicates router hardware => slow path
  - Limits sending speed?
    - To avoid duplicates confusion: $2^{16}/MSL \approx 500$ pkts/sec
- **Better to avoid fragmenting if at all possible!**
How can TCP choose segment size?

- Pick LAN MTU as segment size?
  - LAN MTU usually larger than WAN MTU
  - => Most Internet traffic would be fragmented
- Pick smallest MTU across all networks in Internet?
  - Most traffic is local!
    - Local file server, web proxy, DNS cache, ...
  - Increases packet processing overhead
- Discover MTU to each destination?
  - Path MTU discovery

Path MTU Discovery

- Path MTU is the smallest MTU along path
  - Packets less than this size don’t get fragmented
- Hosts send packets with DF (don’t fragment) set
  - Routers return ICMP packet to source if too large
    - similar to TTL exceeded
  - Binary search to find largest MTU that doesn’t fragment
    - separately for each destination subnet
  - Source TCP uses as segment size
  - Destination TCP reassembles
Layering Revisited

- IP layer “transparent” packet delivery
  - Implementation decisions affect higher layers (and vice versa)
    - Fragmentation => reassembly overhead
      - path MTU discovery
    - Packet loss => congestion or lossy link?
      - link layer retransmission
    - Reordering => packet loss or multipath?
      - router hardware tries to keep packets in order
      - FIFO vs. active queue management

IP Packet Header Limitations

- Fixed size fields in IPv4 packet header
  - source/destination address (32 bits)
    - limits to ~ 4B unique public addresses; about 800M allocated
    - NATs map multiple hosts to single public address
  - IP ID field (16 bits)
    - limits to 65K fragmented packets at once
    - in practice, fewer than 1% of all packets fragment
  - Type of service (8 bits)
    - unused until recently; needed to express priorities
  - TTL (8 bits)
    - limits maximum Internet path length to 255; typical is 30
  - Length (16 bits)
    - Much larger than most link layer MTU’s; path MTU discovery
TCP Packet Header Limitations

- Fixed size fields in TCP packet header
  - seq#/ack# -- 32 bits (can't wrap within MSL)
    - T1 ~ 6.4 hours; OC-192 ~ 3.5 seconds
  - source/destination port# -- 16 bits
    - limits # of connections between two machines (NATs)
    - ok to give each machine multiple IP addresses
  - header length
    - limits # of options
  - receive window size -- 16 bits (64KB)
    - rate = window size / delay
    - Ex: 100ms delay => rate ~ 5Mb/sec