Outline

→ Segment format
→ Connection management
→ Error control
→ Flow control
SEGMENT FORMAT
Segment Header

- Source Port
- Destination Port
- Sequence Number
- Acknowledgement Number
- Checksum
- Urgent Pointer
- Receive window
- Flags
- RES
- HLEN
- Options
- Pad

32 bit
Sequence Numbers

L
Send ‘ABC’
Rec ‘DEF’

R
Send ‘XYZ’
Rec ‘ABC’

Seq=12 data=‘ABC’
Seq=34 Ack=15
Seq=34 data=‘XYZ’
Seq=15 Ack=37

Time
ACK Piggybacking

Send ‘ABC’

Seq=12 data= ‘ABC’

Rec ‘ABC’

Send ‘XYZ’

Seq=34 Ack=15 data= ‘XYZ’

Rec ‘DEF’

Seq=15 Ack=37
Flags

→ ACK: Acknowledgement field is valid

→ PSH: This segment requests a push

→ URG: Urgent Pointer is valid
Flags

→ SYN: Synchronize sequence number
    → Open a connection
→ FIN: Sender has reached end of its byte stream
    → Close a connection
→ RST: Reset connection
Three-way Handshake

App wants to open

L

SYN Seq=12

SYN ACK Seq=34 Ack=13

OK

ACK Seq=13 Ack=35

R

App accepts

time
Connection Termination

App closes

FIN Seq=12

FIN ACK Seq=34 Ack=13

Seq=34 data=...

Seq=45 data=...

ACK Seq=13 Ack=53

FIN Seq=53

FIN ACK Seq=13 Ack=54

OK

App closes
ERROR CONTROL
Retransmission

→ Go-back-N
  → Segments lost in bursts
  → Unnecessary retransmission
  → Overload

→ Selective retransmit optional
Time-out

Unnecessary retransmission

Double timer for outstanding segments
Time-out Too Short

Seq=12
Data=...

Seq=23
Data=...

7mer
ACK

Seq=34
Ack=31

Seq=12
Data=...

Seq=23
Data=...

ACK Seq=34 Ack=31

ACK Seq=34 Ack=31

Round Trip Time estimation

Unnecessary retransmission

Round Trip Time estimation
Cumulative Acknowledgement

L

Seq=12 Data=...
Seq=23 Data=...
ACK Seq=34 Ack=23
ACK Seq=34 Ack=31

R

All well up to 30

timer
Delayed Acknowledgements

L
Send ‘ABC’
Send ‘DEF’

Seq=12 data= ‘ABC’
Seq=15 data= ‘DEF’

Rec ‘ABC’
Rec ‘DEF’

Seq=34 Ack=18

R
Fast Retransmit

L

Seq=6
Seq=12
Seq=23
Seq=35
Ack=6
Ack=6
Ack=6
Seq=6
Seq=12
Seq=23
Seq=42

R

Timer

Ack=42
FLOW CONTROL
Sliding Window

<table>
<thead>
<tr>
<th>Acknowledged</th>
<th>Outstanding</th>
<th>Unsent</th>
</tr>
</thead>
</table>

ACK

Sliding window

<table>
<thead>
<tr>
<th>Acknowledged</th>
<th>Outstanding</th>
<th>Unsent</th>
</tr>
</thead>
</table>
Window Sizing

→ Sender buffer
  → Store unack'ed bytes

→ Receiver
  → Flow control

→ Network
  → Congestion control
On The Receiver Side

→ Provide upper layer with complete ordered stream

→ Buffer
  → Reorder segments
  → Keep data until picked up
Flow Control

- Buffer might become full
- Receiver window advertises available space
- Transmission window always smaller
Congestion Control

→ Network congestion
→ Lost segments
→ Go-back-N retransmission
→ More congestion
→ More lost segments
→ More retransmissions
Congestion Control

- Reduce window size when there is congestion
  - Duplicate ack
  - Time-out
- Increase unit: MSS
  - Maximum Segment Size
Congestion Control

→ Increase when there is no congestion
  → Start small (1 MSS) but increase fast (exponential)
  → Slow start
→ Continue slowly
  → Congestion avoidance
Duplicate Ack
→ Moderate congestion
→ Windows is halved
  → Multiplicative decrease
→ Increase of 1 MSS each received window
  → Additive increase
Time-out

→ Severe congestion
→ Window is set to 1 MSS
→ Increase of 1 MSS each received segment
  → Exponential increase
Time-out

→ Until size is half of window at time-out
→ Then additive increase
To Start

→ Slow start
  → Window = 1 MSS
  → Exponential increase
→ Until first loss event
  → Additive increase
Congestion Control

→ Many issues, several variants
→ None perfect
→ Especially on long fat pipes
  → Long reaction time
Saw Tooth Behavior

Window size vs. Time

Underutilization
Synchronization of Connections

→ Segments from multiple connections discarded during congestion

→ They all slow down

→ Then all increase
   → Until congestion

→ And all slow down ...