Message, Segment, Packet, and Frame

- **HTTP**
- **TCP**
- **IP**
- **Ethernet**
- **SONET**

Connecting components:
- **HTTP message**
- **TCP segment**
- **IP packet**
- **Ethernet frame**
- **SONET frame**

Network components:
- **host**
- **router**
TCP Congestion Control

• Congestion – a function of total number of packets in the network, and where they are

• First step – detection
  • Is packet loss an indication of congestion??
  • All TCP algorithms assume timeouts are caused by congestion

• Initial steps
  • When connection is established – use suitable window size
    • Loss will not occur due to buffers at receiver

• Two issues
  • Network Capacity
  • Receiver Capacity

October 30, 2018
TCP Congestion Control

• Network Capacity and Receiver Capacity
• Maintain two windows
  • Receiver window
  • Congestion window
  • Use the min (Receiver window and Congestion window)
• Initially
  • Sender sets congestion window to MSS (Max Seg Size)
  • If acked add one more MSS – 2 now
  • Repeat for each acked MSS
  • Congestion window grows exponentially
  • If timeout – go back to previous window size
  • SLOW START

October 30, 2018
Internet Congestion Control

• Use a Threshold – initially 64 KB
• When a timeout occurs set threshold to half the current congestion window and reset congestion window to 1 MSS
• Use slow start till the threshold is reached
• Then successful transmissions grow congestion window linearly
TCP Congestion Control (1)

TCP uses AIMD with loss signal to control congestion

- Implemented as a congestion window (cwnd) for the number of segments that may be in the network
- Uses several mechanisms that work together

<table>
<thead>
<tr>
<th>Name</th>
<th>Mechanism</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK clock</td>
<td>Congestion window (cwnd)</td>
<td>Smooth out packet bursts</td>
</tr>
<tr>
<td>Slow-start</td>
<td>Double cwnd each RTT</td>
<td>Rapidly increase send rate to reach roughly the right level</td>
</tr>
<tr>
<td>Additive Increase</td>
<td>Increase cwnd by 1 packet each RTT</td>
<td>Slowly increase send rate to probe at about the right level</td>
</tr>
<tr>
<td>Fast retransmit / recovery</td>
<td>Resend lost packet after 3 duplicate ACKs; send new packet for each new ACK</td>
<td>Recover from a lost packet without stopping ACK clock</td>
</tr>
</tbody>
</table>
TCP Congestion Control (2)

Congestion window controls the sending rate

- Rate is cwnd / RTT; window can stop sender quickly
- ACK clock (regular receipt of ACKs) paces traffic and smooths out sender bursts
TCP Congestion Control (3)

Slow start grows congestion window exponentially

• Doubles every RTT while keeping ACK clock going

Increment cwnd for each new ACK
TCP Congestion Control (4)

Additive increase grows cwnd slowly
- Adds 1 every RTT
- Keeps ACK clock
TCPCongestion Control (5)

Slow start followed by additive increase (TCP Tahoe)

- Threshold is half of previous loss cwnd

Loss causes timeout; ACK clock has stopped so slow-start again
TCP Congestion Control (6)

- With fast recovery, we get the classic sawtooth (TCP Reno)
  - Retransmit lost packet after 3 duplicate ACKs
  - New packet for each dup. ACK until loss is repaired

The ACK clock doesn’t stop, so no need to slow-start
TCP Congestion Control (7)

SACK (Selective ACKs) extend ACKs with a vector to describe received segments and hence losses

- Allows for more accurate retransmissions / recovery

No way for us to know that 2 and 5 were lost with only ACKs
Wireless TCP and UDP

Splitting a TCP connection into two connections.

Sender
TCP #1
Base station
TCP #2
Mobile host
Router
Antenna

October 30, 2018
Transactional TCP

(a) RPC using normal TPC.
(b) RPC using T/TCP.

October 30, 2018
Performance Issues

Many strategies for getting good performance have been learned over time

- Performance problems »
- Measuring network performance »
- Host design for fast networks »
- Fast segment processing »
- Header compression »
- Protocols for “long fat” networks »
Performance Problems

Unexpected loads often interact with protocols to cause performance problems

• Need to find the situations and improve the protocols

Examples:

• Broadcast storm: one broadcast triggers another
• Synchronization: a building of computers all contact the DHCP server together after a power failure
• Tiny packets: some situations can cause TCP to send many small packets instead of few large ones
Host Design for Fast Networks

Poor host software can greatly slow down networks.

Rules of thumb for fast host software:

• Host speed more important than network speed
• Reduce packet count to reduce overhead
• Minimize data touching
• Minimize context switches
• Avoiding congestion is better than recovering from it
• Avoid timeouts
Fast Segment Processing (1)

Speed up the common case with a fast path [pink]

- Handles packets with expected header; OK for others to run slowly
Fast Segment Processing (2)

Header fields are often the same from one packet to the next for a flow; copy/check them to speed up processing

TCP header fields that stay the same for a one-way flow (shaded)

IP header fields that are often the same for a one-way flow (shaded)
Fast TPDU Processing (3)

A timing wheel.

October 30, 2018
Header Compression

Overhead can be very large for small packets
  • 40 bytes of header for RTP/UDP/IP VoIP packet
  • Problematic for slow links, especially wireless

Header compression mitigates this problem
  • Runs between Link and Network layer
  • Omits fields that don’t change or change predictably
    • 40 byte TCP/IP header → 3 bytes of information
  • Gives simple high-layer headers and efficient links
Protocols for “Long Fat” Networks (1)

Networks with high bandwidth (“Fat”) and high delay (“Long”) can store much information inside the network

- Requires protocols with ample buffering and few RTTs, rather than reducing the bits on the wire

Starting to send 1 Mbit
San Diego → Boston

20ms after start

40ms after start
Protocols for “Long Fat” Networks (2)

You can buy more bandwidth but not lower delay

- Need to shift ends (e.g., into cloud) to lower further

Minimum time to send and ACK a 1-Mbit file over a 4000-km line

October 30, 2018
Delay Tolerant Networking

DTNs (Delay Tolerant Networks) store messages inside the network until they can be delivered

- DTN Architecture »
- Bundle Protocol »
Messages called bundles are stored at DTN nodes while waiting for an intermittent link to become a contact

- Bundles might wait hours, not milliseconds in routers
- May be no working end-to-end path at any time
Example DTN connecting a satellite to a collection point
The Bundle protocol uses TCP or other transports and provides a DTN service to applications.
Bundle Protocol (2)

Features of the bundle message format:

- Dest./source add high-level addresses (not port/IP)
- Custody transfer shifts delivery responsibility
- Dictionary provides compression for efficiency

Diagram:

- Primary block
  - Ver.
  - Flags
  - Dest.
  - Source
  - Report
  - Custodian
  - Creation
  - Lifetime
  - Dictionary

- Payload block
  - 8 bits
  - 20 bits
  - Variable

- Optional blocks
  - 8 bits
  - 6 bits
  - Variable
  - Type
  - Flags
  - Length
  - Data

- Additional fields:
  - Status report
  - Class of service
  - General