The Road Ahead

- Transport introduction
- Error recovery & flow control basics
- TCP Flow Control basics

Transport Protocols

- Lowest level end-to-end protocol.
  - Header generated by sender is interpreted only by the destination
  - Routers view transport header as part of the payload
Functionality Split

- Network provides best-effort delivery
- End-systems implement many functions
  - Reliability
  - In-order delivery
  - De-multiplexing
  - Message boundaries
  - Connection abstraction
  - Congestion control
  - …

Transport Protocols

- UDP provides just integrity and demux
- TCP adds…
  - Connection-oriented
  - Reliable
  - Ordered
  - Point-to-point
  - Byte-stream
  - Full duplex
  - Flow and congestion controlled
- Request-reply service
  - RPC-like
  - Not covered here

UDP: User Datagram Protocol

- “No frills,” “bare bones” Internet transport protocol
- “Best effort” service, UDP segments may be
  - Lost
  - Delivered out of order to app
- Connectionless:
  - No handshaking between UDP sender, receiver
  - Each UDP segment handled independently of others

Why is there a UDP?
- No connection establishment (which can add delay)
- Simple: no connection state at sender, receiver
- Small header
- No congestion control: UDP can blast away as fast as desired
More on UDP

- Often used for streaming multimedia apps
  - Loss tolerant
  - Rate sensitive
- Other UDP uses (why?):
  - DNS, SNMP
- Reliable transfer over UDP
  - Must be at application layer
  - Application-specific error recovery

TCP

<table>
<thead>
<tr>
<th>Flags</th>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>URG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In-order, Connection oriented, Byte stream abstraction

Reliability: Straw-man approaches

- Receiver sends acknowledgement (ACK)
  - Sender waits for ACK and timeouts if it does not arrive within some time period
- Simplest version: Send a packet, stop and wait until ACK arrives
  - Problems
Recovering from Error

How to Recognize Duplicates?
- Use sequence numbers
  - Both packets and acks
- Sequence # in packet is finite
  - How big should it be?
- For stop and wait?
  - One bit – won’t send seq #1 until received ACK for seq #0
- Problem with Stop and Wait:
  - Poor efficiency

How to Ensure Efficiency?
- How to “keep the pipe full”? (Answer: Pipelining)
- Send multiple packets without waiting for first to be acknowledged
- How many such packets max?
  - Suppose 10Mbps link, 4ms delay, 500byte pkts
  - Round trip delay * bandwidth = capacity of pipe
  - How many? 10? 20? 50?
  - Consequences:
    - Cannot use 1 bit sequence number any more
    - Buffering may be required
    - Range of sequence number and buffer size will depend on loss recovery
Pipelining Implementation:
“Sliding Window”
• Sliding buffer at sender and receiver
  - Packets in transit ≤ sender buffer size
  - Advance when sender and receiver agree packets at beginning have been received
• Receiver has to buffer a packet until all prior packets have arrived
• Goal: provides reliable, ordered delivery
• Two ways to do this:
  - Go-Back-N
  - Selective Repeat

GBN Window Sliding – Common Case
• On reception of new ACK (i.e., ACK for something that was not acked earlier)
  - Increase sequence of max ACK received
  - Send next packet
• On reception of new in-order data packet (next expected)
  - Send cumulative ACK – acknowledges receipt of all in-sequence packets up to sequence number
  - Increase sequence of max acceptable packet

Go-Back-N:
Sender/Receiver State
Go-Back-N with Losses

- On reception of out-of-order packet
  - Don’t ACK (wait for source to timeout)
  - Discard out of order packets
  - Cumulative ACK (helps source identify loss)

- Timeout (Go-Back-N recovery)
  - Set timer upon transmission of packet
  - Retransmit all unacknowledged packets

Go-Back-N With Losses

- Simple behavior
  - One timeout
  - Simple buffering

- Performance during loss recovery
  - No longer have an entire window in transit
  - Can have much more clever loss recovery

Selective Repeat

- Receiver individually acknowledges all correctly received packets
  - Buffers packets, as needed, for eventual in-order delivery to upper layer

- Sender only resends packets for which ACK not received
  - Sender timer for each unACKed packet

- Sender window
  - N consecutive seq #’s
  - Again limits seq #’s of sent, unACKed packets
Selective Repeat: 
Sender, Receiver Windows

Sequence Numbers
- How large do sequence numbers need to be?
  - Depends on sender/receiver window size
    - Must take wrap-around into account
- E.g.
  - Max seq = 7, window_size = 7
    - If pkts 0..6 are sent successfully and all acks lost
      - Receiver expects 7, 0..5, sender retransmits old 0..6!!!
  - Max sequence must be \( \geq 2 \times \text{window}_\text{size} \)
- TCP uses 32 bit sequence numbers

TCP Flow Control
- TCP is a sliding window protocol
  - For window size \( n \), can send up to \( n \) bytes without receiving an acknowledgement
  - When the data is acknowledged then the window slides forward
- Each packet advertises a window size
  - Indicates number of bytes the receiver has space for
- Original TCP always sent entire window
  - Congestion control now limits this
TCP Sequence Numbers

- Sequence Number Space
  - Each byte in byte stream is numbered.
  - 32 bit value
  - Wraps around

- Initial values selected at start up time
  - TCP breaks up the byte stream in packets.

- Packet size is limited to the Maximum Segment Size
  - Each packet has a sequence number.
  - Indicates where it fits in the byte stream

Window Flow Control: Send Side

Packet Sent

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest. Port</th>
<th>Sequence Number</th>
<th>Acknowledgment</th>
<th>HL/Flags</th>
<th>Window</th>
<th>Checksum</th>
<th>Urgent Pointer</th>
<th>Options</th>
</tr>
</thead>
</table>

Packet Received

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest. Port</th>
<th>Sequence Number</th>
<th>Acknowledgment</th>
<th>HL/Flags</th>
<th>Window</th>
<th>Checksum</th>
<th>Urgent Pointer</th>
<th>Options</th>
</tr>
</thead>
</table>

Summary

- Transport service
  - UDP: mostly just IP service
  - TCP: congestion controlled, reliable, byte stream

- Types of ARQ protocols
  - Stop-and-wait: slow, simple
  - Go-back-n: can keep link utilized (except w/ losses)
  - Selective repeat: efficient loss recovery

- Sliding window flow control
  - Addresses buffering issues and keeps link utilized
  - TCP uses sliding window
  - 32 bit sequence numbers