CSE 461 – Module 10

Introduction to the Transport Layer
Last Time

• We finished up the Network layer
  – Internetworks (IP)
  – Routing (DV/RIP, LS/OSPF, BGP)

• It was all about routing: how to provide end-to-end delivery of packets.
This Time

• We begin on the Transport layer

• Focus
  – Process-to-process communication
    • Fast?
    • Reliable?
  – Impact on the network
    • Congestion control

• Topics
  – The Transport layer
  – Acknowledgements and retransmissions (ARQ)
  – Sliding windows
The Transport Layer

- Builds on the services of the Network layer
  - "TCP/IP"

- Communication between processes running on hosts
  - Naming/Addressing

- Stronger guarantees of message delivery make sense
  - This is the first layer that is talking "end-to-end"
Internet Transport Protocols

• **UDP**
  - Datagram abstraction between processes
  - With error detection

• **TCP**
  - Bytestream abstraction between processes
  - With reliability
  - Plus congestion control (later!)
## UDP/IP Properties (User Datagram Protocol)

<table>
<thead>
<tr>
<th>UDP</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Datagram oriented</td>
<td>• Datagram oriented</td>
</tr>
<tr>
<td>• Lost packets</td>
<td>• Lost packets</td>
</tr>
<tr>
<td>• Reordered packets</td>
<td>• Reordered packets</td>
</tr>
<tr>
<td>• Duplicate packets</td>
<td>• Duplicate packets</td>
</tr>
<tr>
<td>• Limited size packets</td>
<td>• Limited size packets</td>
</tr>
</tbody>
</table>
TCP/IP Properties
(Transmission Control Protocol)

TCP
- Connection-oriented
- Reliable byte-stream delivery
  - In-order delivery
  - Single delivery
  - Arbitrarily long messages
- Synchronization
- Flow control
- Congestion control

IP
- Datagram oriented
- Lost packets
- Reordered packets
- Duplicate packets
- Limited size packets
TCP Packet Format

16 bit window size gets cramped with large Bandwidth x delay

- 16 bits --> 64K
- BD ethernet: 122KB
- STS24 (1.2Gb/s): 14.8MB

32 bit sequence number must not wrap around faster than the maximum packet lifetime. (120 seconds)

-- 622Mb/s link: 55 seconds
TCP End-to-End Properties

• TCP provides a full-duplex connection
  – Each side of a connection can send to the other

• Connection is a stream
  – Packet boundaries may not be visible to application

• Sliding window
  – Endpoints exchange window sizes
  – Packets carry sequence numbers
    • Actually, byte counts in the connection stream
  – Performance
  – Reliability (ARQ)
End-to-end Properties

• Performance
  – Sliding Window
    • Try to enable sender to put bandwidth \( x \) delay product bytes on the wire

• Reliability
  – Lost packets?
    • Sliding window performs flow control
    • Sliding window performs ARQ (Automatic Repeat Request)
  – Duplicate / out-of-order packets?
    • Sliding window receive (re-order) buffer
Network Property: Congestion Control

- TCP also implements congestion control
  - High level goal: keep from over-loading the bottleneck network link
  - Immediate goal: find the fastest transmission rate that doesn’t overload the bottleneck

- Does it make sense to put congestion control in TCP?
  - Could it be in some other layer?
    - Would it make sense to apply it to UDP?

- Another goal: fairness
  - I’m not slowing down, you slow down...
# TCP / UDP comparison

<table>
<thead>
<tr>
<th>TCP</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable</td>
<td>Unreliable</td>
</tr>
<tr>
<td>Stream-oriented</td>
<td>Packet-oriented</td>
</tr>
<tr>
<td>Connection</td>
<td>Connectionless</td>
</tr>
</tbody>
</table>
TCP / UDP comparison

• Stream- vs. packet-oriented
  – Visible packet boundaries can act as “end of record” indicators to application
  – In a stream, if the application wants the notion of “records”, it must embed them in the data
    • Example: lines in a text file
  – Since TCP doesn’t know about app record boundaries, reading records can be cumbersome
    • Each read() operation returns whatever data happens to have arrived in the stream to this point
TCP / UDP comparison

• Connection vs. connectionless
  – UDP: “flexible” (or “you don’t know who you’re talking with”)
    • Incoming data can be from anywhere
    • Outgoing data can go anywhere
    • *(Java API provides a connect() interface – filters packets before returning them to app)*
  
  – TCP: incoming/outgoing packets are separated into “flows”
    • Provides a nice programming abstraction for many apps
    • How do I open a connection?
    • How do I close one?
    • How do I know when the other side has stopped listening/sending