Lecture 6: Transport Layer Protocols

CSE 123: Computer Networks
Alex C. Snoeren
Overview

- Process naming/demultiplexing
- User Datagram Protocol (UDP)
- Transport Control Protocol (TCP)
  - Three-way handshake
  - Flow control
Today: Transport Layer

Application Layer

Transport Layer

Network Layer

Link Layer

- HTTP
- TCP
- IP
- Ethernet
- SONET
- Ethernet interface
- SONET interface

host

router

CSE 123 – Lecture 6: Transport Protocols
Transport Layer Tasks

- Define and provide specific delivery semantics
  - To which end point?
  - When?
  - How?
  - If?

- Multiplexing

- Reliability
Naming Processes/Services

- Process here is an abstract term for your Web browser (HTTP), Email servers (SMTP), hostname translation (DNS).

- How do we identify for remote communication?
  - Process id or memory address are OS-specific and transient.

- So TCP and UDP use **ports**
  - 16-bit integers representing mailboxes that processes “rent”
  - Identify process uniquely as (IP address, protocol, port)
Picking Port Numbers

- We still have the problem of allocating port numbers
  - What port should a Web server use on host X?
  - To what port should you send to contact that Web server?

- Servers typically bind to well-known port numbers
  - e.g., HTTP 80, SMTP 25, DNS 53, … look in /etc/services
  - Ports below 1024 traditionally reserved for well-known services

- Clients use OS-assigned temporary (ephemeral) ports
  - Above 1024, recycled by OS when client finished
User Datagram Protocol (UDP)

- Provides *unreliable message delivery* between processes
  - Source port filled in by OS as message is sent
  - Destination port identifies UDP delivery queue at endpoint
- Connectionless (no state about who talks to whom)

<table>
<thead>
<tr>
<th></th>
<th>SrcPort</th>
<th>DstPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Checksum</td>
<td>Length</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

CSE 123 – Lecture 6: Transport Protocols
UDP Delivery

Packets arrive

Ports

Message Queues

Application process

Application process

Application process

DeMux

Kernel boundary

CSE 123 – Lecture 6: Transport Protocols
UDP Checksum

- UDP includes optional protection against errors
  - Checksum intended as an end-to-end check on delivery
  - So it covers data, UDP header, and IP pseudoheader (history)
Applications for UDP

- Streaming media (e.g., live video)
- DNS (Domain Name Service)
- NTP (Network Time Protocol) (synchronizing clocks)
- FPS multi-player video games (e.g., Call of Duty)
- Why might UDP be appropriate for these?
Transmission Control Protocol

- Reliable bi-directional bytestream between processes
  - Uses a sliding window protocol for efficient transfer

- Connection-oriented
  - Conversation between two endpoints with beginning and end

- Flow control (last lecture)
  - Prevents sender from over-running receiver buffers
  - (tell sender how much buffer is left at receiver)

- Congestion control (later in term)
  - Prevents sender from over-running network capacity
TCP Delivery

Application process

Write bytes

TCP
Send buffer

Transmit segments

Segment Segment Segment

Application process

Read bytes

TCP
Receive buffer
## TCP Header Format

- Ports plus IP addresses identify a connection (**4-tuple**)

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>0-15</td>
</tr>
<tr>
<td>DstPort</td>
<td>16-31</td>
</tr>
<tr>
<td>HdrLen</td>
<td>31</td>
</tr>
<tr>
<td>SequenceNum</td>
<td>16-31</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>16-31</td>
</tr>
<tr>
<td>AdvertisedWindow</td>
<td>16-31</td>
</tr>
<tr>
<td>Checksum</td>
<td>16-31</td>
</tr>
<tr>
<td>UrgPtr</td>
<td>16-31</td>
</tr>
<tr>
<td>Options (variable)</td>
<td>16-31</td>
</tr>
<tr>
<td>Data</td>
<td>32-65535</td>
</tr>
</tbody>
</table>

---

CSE 123 – Lecture 6: Transport Protocols
TCP Header Format

- Sequence, Ack numbers used for the sliding window
  - How big a window? Flow control/congestion control determine

```
0  4  10  16  31
SrcPort | DstPort

SequenceNum
Acknowledgment
HdrLen  0  Flags  AdvertisedWindow
Checksum  UrgPtr
Options (variable)
Data
```
TCP Header Format

- Flags may be ACK, SYN, FIN, URG, PSH, RST

<table>
<thead>
<tr>
<th></th>
<th>SrcPort</th>
<th>DstPort</th>
<th>SequenceNum</th>
<th>Acknowledgment</th>
<th>HdrLen</th>
<th>Flags</th>
<th>AdvertisedWindow</th>
<th>Checksum</th>
<th>UrgPtr</th>
<th>Options (variable)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Flags</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CSE 123 – Lecture 6: Transport Protocols
Three-Way Handshake

- Opens both directions for transfer

```
Active participant (client)  Passive participant (server)
SYN, SequenceNum = x
SYN + ACK, SequenceNum = y, Acknowledgment = x + 1
ACK, Acknowledgment = y + 1
+data
```
Some Comments

- We could abbreviate this setup, but it was chosen to be robust, especially against delayed duplicates
  - Three-way handshake from Tomlinson 1975

- Choice of changing initial sequence numbers (ISNs) minimizes the chance of hosts that crash getting confused by a previous incarnation of a connection

- How to choose ISNs?
  - Maximize period between reuse
  - Minimize ability to guess (why?)
TCP State Transitions

- CLOSED
  - Passive open
  - Close
- LISTEN
  - SYN/SYN + ACK
  - Send/ SYN
- SYN_RCVD
  - SYN + ACK/ACK
- SYN_SENT
  - SYN/SYN + ACK
- ESTABLISHED
  - ACK
  - Close /FIN
- FIN_WAIT_1
  - ACK
  - FIN/ACK
- FIN_WAIT_2
  - FIN/ACK
- CLOSING
  - ACK
  - Time wait
  - Timeout after two segment lifetimes
- LAST_ACK
  - ACK
- CLOSE_WAIT
  - Close /FIN
- CLOSING
  - Time wait
  - Closed
Again, with States

Again, with States

Active participant (client)

SYN_SENT

SYN, SequenceNum = x

SYN + ACK, SequenceNum = y,

ACK, Acknowledgment = x + 1

ESTABLISHED

+data

Passive participant (server)

LISTEN

SYN_RCVD

ESTABLISHED

CSE 123 – Lecture 6: Transport Protocols
For next time…

- More TCP details
- Read Ch 5.2 in P&D