Lecture 6: Transport Layer Protocols

CSE 123: Computer Networks
Alex C. Snoeren

HW 1 due WEDNESDAY

UCSD CSE
Overview

- Process naming/demultiplexing
- User Datagram Protocol (UDP)
- Transport Control Protocol (TCP)
  - State transitions
  - Connection teardown

How’s the homework going?
A. Already done
B. Seems doable
C. I’m scared
D. Homework? There’s a homework?
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)
UDP Delivery

Packets arrive

Ports

Message Queues

Application process

Application process

Application process

DeMux

Packets arrive

Kernel boundary
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 (receive window)
  - 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
What corresponds to the receive window size?

A. Segment size  
B. Bandwidth*delay product  
C. Send buffer space  
D. Receive buffer space
TCP Header Format

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

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>10</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>DstPort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SequenceNum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HdrLen</td>
<td>0</td>
<td>Flags</td>
<td>AdvertisedWindow</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td></td>
<td>UrgPtr</td>
<td></td>
</tr>
<tr>
<td>Options (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CSE 123 – Lecture 6: Transport Layer 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
```

CSE 123 – Lecture 6: Transport Layer Protocols
TCP Header Format

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>0</td>
</tr>
<tr>
<td>DstPort</td>
<td>4</td>
</tr>
<tr>
<td>SequenceNum</td>
<td>10</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>16</td>
</tr>
<tr>
<td>HdrLen</td>
<td>0</td>
</tr>
<tr>
<td>Flags</td>
<td>0</td>
</tr>
<tr>
<td>AdvertisedWindow</td>
<td>16</td>
</tr>
<tr>
<td>Checksum</td>
<td>16</td>
</tr>
<tr>
<td>UrgPtr</td>
<td>16</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
Connection Establishment

- Both sender and receiver must be ready before we start to transfer the data
  - Sender and receiver need to agree on a set of parameters
  - Most important: sequence number space in each direction
  - Lots of other parameters: e.g., the window size

- Handshake protocols: setup state between two oblivious endpoints
  - Need to deal with delayed and reordered packets
Two-way handshake?

Active participant (client)  Passive participant (server)

SYN, SequenceNum = x

SYN, SequenceNum = y

What’s wrong here?

+data
Two-way handshake?

Active participant (client)

Passive participant (server)

Old SYN, SequenceNum = x

New SYN, SequenceNum = q

SYN, SequenceNum = y

Delayed old SYN

Rejected

+data
Three-Way Handshake

- Opens both directions for transfer

<table>
<thead>
<tr>
<th>Active participant (client)</th>
<th>Passive participant (server)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN, SequenceNum = $x$</td>
<td>SYN + ACK, SequenceNum = $y$, Acknowledgment = $x + 1$</td>
</tr>
<tr>
<td>SYN + ACK, SequenceNum = $y$, Acknowledgment = $x + 1$</td>
<td>ACK, Acknowledgment = $y + 1$</td>
</tr>
</tbody>
</table>

+data
Connection Teardown

- Orderly release by sender and receiver when done
  - Delivers all pending data and “hangs up”

- Cleans up state in sender and receiver

- TCP provides a “symmetric” close
  - Both sides shutdown independently

Why a symmetric close?

A. Need to ensure no data loss
B. Data transfer is bi-directional
C. Both ends need to know when they can “forget” about connection
D. All of the above
TCP Connection Teardown

Web server

FIN

ACK

FIN

ACK

TIME_WAIT

CLOSED

Web browser

CLOSED

CSE 123 – Lecture 6: Transport Layer Protocols
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?)
For next time...

- Begin to discuss actual *networks*!
- Read Ch 3-3.1 in P&D
- NO CLASS MONDAY (Enjoy MLK Day)
- Homework due BEFORE NEXT CLASS