Lecture 7: Transport Control Protocol

CSE 123: Computer Networks
Aaron Schulman
Lecture 7 Overview

- TCP Connection Management
  - Starting a TCP connection
  - Graceful, symmetric close
  - The TIME_WAIT state

- Connecting links together to form networks
  - Hubs/repeaters
  - Switching
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 Maximum Segment Size

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

Active participant (client)

SYN, SequenceNum = x

Passive participant (server)

SYN, SequenceNum = y

+data

What’s wrong here?

CSE 123 – Lecture 7: TCP
Two-way handshake?

Active participant (client)

Old SYN, SequenceNum = x

New SYN, SequenceNum = q

SYN, SequenceNum = y

Delayed old SYN

Rejected

Passive participant (server)

Client and server do not agree they are starting the same connection
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
Again, with States

---

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

+data

CSE 123 – Lecture 7: TCP
Three-Way Handshake

- Opens both directions for transfer

Active participant (client)

SYN, SequenceNum = $x$

SYN + ACK, SequenceNum = $y$

ACK, Acknowledgment = $x + 1$

Passive participant (server)

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**
  - Send/ SYN

- **SYN_RCVD**
  - SYN/SYN + ACK

- **SYN_SENT**
  - SYN + ACK/ACK
  - Timeout after two segment lifetimes

- **ESTABLISHED**
  - FIN/ACK
  - Close /FIN

- **FIN_WAIT_1**
  - ACK
  - Close /FIN

- **FIN_WAIT_2**
  - FIN/ACK

- **CLOSING**
  - ACK

- **CLOSE_WAIT**
  - Close /FIN

- **LAST_ACK**
  - ACK

- **TIME_WAIT**
  - Timeout after two segment lifetimes

CSE 123 – Lecture 6: Transport Protocols
Again, with States

Active participant (client)  Passive participant (server)

SYN_SENT

SYN, SequenceNum = x

SYN + ACK, SequenceNum = y,
Acknowledgment = x + 1

ACK, Acknowledgment = y + 1

ESTABLISHED

LISTEN

SYN_RCVD

ESTABLISHED

+data

CSE 123 – Lecture 6: Transport Protocols
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
TCP Connection Teardown

Server

FIN_WAIT_1

FIN_WAIT_2

TIME_WAIT

CLOSED

Client

CLOSE_WAIT

LAST_ACK

CLOSED

FIN

ACK

ACK

FIN

CSE 123 – Lecture 7: TCP
The TIME_WAIT State

- We wait 2*MSL (maximum segment) before completing the close
- Maximum segment lifetime is usually 60 seconds (longest period of time packet can be stuck in network)
  - Why?
- ACK might have been lost and so FIN will be resent
  - Could interfere with a subsequent connection
- Real life: Abortive close
  - Don’t wait for 2*MSL, simply send Reset packet (RST)
  - Why?
Lots of Icky Details

- Window probes
- Silly Window Syndrome
- Nagle’s algorithm
- PAWS
- Etc…

- Steven’s books “TCP/IP Illustrated (vol 1,2)” is a great source of information on this
Lots of Icky Details

- Nagle’s algorithm
- Window probes
- Silly Window Syndrome
- Nagle’s algorithm
- PAWS
- Etc…

- Steven’s books “TCP/IP Illustrated (vol 1,2)” is a great source of information on this
Jumping back down
So far, just one link

- One shared link (a bus) limits scale in terms of:
  - Distance (e.g., 2500 m for Ethernet)
  - Number of nodes (1024 nodes)
  - Performance (Capacity shared across all nodes)

- Often called a **Local Area Network (LAN)**

- A better alternative is to have multiple busses
  - Each bus is of a limited size, scale, number of hosts, etc.

- Need the ability to connect multiple busses together
  - In other words move frames from one wire to another
For next time…

- Read P&D 3-3.1