Announcements

- Homework 8 due now
- Final project
- Demo in lab + solitaire player mode due Wed/Thur
- We’ll try to provide some feedback regarding implementation plans

Last Time

- Reviewed PageRank
- Learned about the transport layer
- In between applications and TCP
- Adds a port number
- Learned about UDP and TCP

Today’s Outline

- Continue discussing TCP
  - A “heavier-weight” but reliable alternative to UDP
  - Much more commonly used than UDP in Internet applications
**UDP Recap**

- Provides port numbers
- Guarantees:
  - If your data arrives, it has not been corrupted (optional)
- Does not guarantee:
  - Order of packets, delivery, uniqueness
  - "Send and pray" mentality
  - Connect, send, disconnect (no "on-going" conversation)

**TCP Recap**

- Transmission control protocol
- Provides reliable, ordered delivery of a stream of bytes from a process/program on one computer to another process/program on another computer
- Invented by Vint Cerf and Bob Kahn in 1974
- Handles congestion and flow control (we won't discuss in detail)
  - Congestion control - make sure sender doesn't overrun network
  - Flow control - make sure sender doesn't overrun receiver

**NORMAL TCP DATA TRANSMISSION**

- If you had a lazy mailman who occasionally "lost" letters, what would you do?
- Registered mail with return rcpt
- TCP notifies sends when their packets are delivered with an ACKNOWLEDGEMENT (ACK) packet
- Unlike UDP, TCP connections are like long-lived conversations
- How does receiver know that acknowledgement successfully gets delivered?

**Sequence and Acknowledgment Numbers**

- Loss Case
- No Loss Case
Problems with Acknowledgements?

- Do they reduce our transfer rate?
- We spend a lot of time waiting around for ACKs to arrive...

NORMAL TCP DATA TRANSMISSION?

“Stop-and-wait” is wasteful! We probably have more bandwidth to work with than one packet’s worth of data...

What can we do instead?

Maintaining Transmission Efficiency

We want to “keep the pipe full!”

How can we avoid retransmitting lost ACKs?
Cumulative Acknowledgments

Acknowledge the highest sequence number seen. That way we avoid having to retransmit lost ACKs.

What happens when there actually is a loss? Rather than waiting for a timeout, let 3 duplicate ACKs indicate a loss.

Byte Number Sequencing

Flow control — Each side of a TCP connection specifies a Receive Window size (number of bytes OS is willing to reserve before requiring an ACK).

Slow Segments == Confused Conversations

Timer expires. Retransmission of GET /index.html 1

GET /main.html 1
The Connection Start 3-Way Handshake

- SYN (seq # = x)
- Ack x+1 & SYN (seq # = y)
- GET /index.html x+1
- Ack y+1
- Ack x+2
- Contents of index.html (seq = y+1)

TCP Segment Format

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Sequence (Segment) Number</td>
<td>Acknowledgment Number</td>
</tr>
<tr>
<td>Hdr Len</td>
<td>Flags</td>
</tr>
<tr>
<td>Error Check</td>
<td>Urgent Pointer</td>
</tr>
</tbody>
</table>

Flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nothing)</td>
<td>Here’s some data</td>
</tr>
<tr>
<td>SYN</td>
<td>Connect</td>
</tr>
<tr>
<td>FIN</td>
<td>Disconnect</td>
</tr>
<tr>
<td>ACK</td>
<td>I got segment #__</td>
</tr>
<tr>
<td>RESET</td>
<td>I’m confused and want to hang up</td>
</tr>
<tr>
<td>URG</td>
<td>By the way, ...</td>
</tr>
<tr>
<td>PUSH</td>
<td>Send data now</td>
</tr>
</tbody>
</table>

Connection Termination

- FIN (seq # = z)
- Ack w+1
- FIN (seq # = w)
- Ack z+1 & FIN (seq # = w)
Today we discussed TCP

- TCP provides a reliable alternative to UDP
- TCP sits “above” IP
- TCP segments are encapsulated in IP data field
- TCP avoids congestion
- Just about everything we use on the Internet uses TCP