Discussion 1: Project 1, Networking Basics

CSE 123: Computer Networks
Marti Motoyama & Chris Kanich
CAPTCHAs
Measures of success

- How fast?
  - **Bandwidth** measured in bits per second
  - Often talk about KBps or Mbps – Bytes vs bits

- How long was the wait?
  - **Delay** (one-way or round trip) measured in seconds

- How efficiently?
  - **Overhead** measured in bits or seconds or cycles or…

- Any mistakes?
  - **Error rate** measured in terms of probability of flipped bit
How long to send a message?

- Transmit time $T = \frac{M}{R} + D$
  - 10 Mbps Ethernet LAN (M=1KB)
    » $\frac{M}{R}=1\text{ms}$, $D \approx 5\text{us}$
  - 155 Mbps cross country ATM link (M=1KB)
    » $\frac{M}{R}=50\text{us}$, $D \approx 40-100\text{ms}$

- Where are the bits in the mean time?
  - In transit inside the network

- $R*D$ is called the **bandwidth delay product**
  - How many bits can be “stored” be stored in transit
  - Colloquially, we say “fill the pipe”
TCP/IP Protocol Stack

Application Layer

Transport Layer

Network Layer

Link Layer
Link Layer

Network Layer

Transport Layer

Application Layer
Putting this all together

- **ROUGHLY**, what happens when I click on a Web page from UCSD?

My computer

www.google.com

Internet

CSE 123 – Lecture 1: Course Introduction
Web request (HTTP)

- Turn click into HTTP request

GET http://www.google.com/ HTTP/1.1
Host: www.google.com
Connection: keep-alive
...
Name resolution (DNS)

- Where is www.google.com?

My computer (132.239.9.64)

Local DNS server (132.239.51.18)

What’s the address for www.google.com

Oh, you can find it at 66.102.7.104
Frame 33 (76 bytes on wire, 76 bytes captured)
Ethernet II, Src: IntelCor_0e:1b:2b (00:1f:3b:ae:1b:2b), Dst: Cisco_4a:70:00 (00:21:56:4a:70:00)
Internet Protocol, Src: 128.54.1.79 (128.54.1.79), Dst: 132.239.0.252 (132.239.0.252)
User Datagram Protocol, Src Port: 62619 (62619), Dst Port: domain (53)
Domain Name System (query)

0000 00 21 56 4a 70 00 00 0f 3b ae 1b 2b 08 00 45 00  .!VJp... ;+...E.
0010 00 3e 4f a3 00 00 80 11 e3 9b 80 36 01 4f 84 ef  :>0.3e...6.0.
0020 00 fc f4 9b 00 35 00 2a 8b d0 81 88 01 00 00 01  ......5.*
0030 00 00 00 00 00 00 00 03 77 77 77 08 66 61 63 65 62  ........w www.facebook
0040 6f 6f 6b 03 63 6d 6f 6d 00 00 01 00 01 01 oo.k.com. ....
Data transport (TCP)

- Break message into packets (TCP segments)
- Should be delivered reliably & in-order

```
GET http://www.google.com HTTP/1.1
Host: www.google.com
Connection: keep-alive
...`

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>3.257933</td>
<td>128.239.0.202</td>
<td>128.34.1.79</td>
<td>ICMP</td>
<td>Destination unreachable (Port unreachable)</td>
</tr>
<tr>
<td>33</td>
<td>3.880033</td>
<td>128.54.1.79</td>
<td>128.239.0.252</td>
<td>DNS</td>
<td>Standard query A <a href="http://www.facebook.com">www.facebook.com</a></td>
</tr>
<tr>
<td>34</td>
<td>3.888454</td>
<td>128.239.0.252</td>
<td>128.54.1.79</td>
<td>DNS</td>
<td>Standard query response A 66.220.147.33</td>
</tr>
<tr>
<td>35</td>
<td>3.889163</td>
<td>128.54.1.79</td>
<td>128.239.0.252</td>
<td>DNS</td>
<td>Standard query AAAA <a href="http://www.facebook.com">www.facebook.com</a></td>
</tr>
<tr>
<td>36</td>
<td>3.890698</td>
<td>128.239.0.252</td>
<td>128.54.1.79</td>
<td>DNS</td>
<td>Standard query response</td>
</tr>
<tr>
<td>37</td>
<td>3.959996</td>
<td>128.54.1.79</td>
<td>66.220.147.33</td>
<td>TCP</td>
<td>53792 &gt; http [SYN] Seq=0 Win=8192 Len=0 MSS=1460</td>
</tr>
<tr>
<td>38</td>
<td>3.977502</td>
<td>66.220.147.33</td>
<td>128.54.1.79</td>
<td>TCP</td>
<td>http &gt; 53792 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460</td>
</tr>
<tr>
<td>39</td>
<td>3.977671</td>
<td>66.220.147.33</td>
<td>128.54.1.79</td>
<td>TCP</td>
<td>53792 &gt; http [ACK] Seq=1 Ack=558 Win=4937 Len=0</td>
</tr>
<tr>
<td>40</td>
<td>3.981512</td>
<td>66.220.147.33</td>
<td>128.54.1.79</td>
<td>HTTP</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>42</td>
<td>4.177880</td>
<td>66.220.147.33</td>
<td>128.54.1.79</td>
<td>TCP</td>
<td>http &gt; 53792 [ACK] Seq=1 Ack=558 Win=4937 Len=0</td>
</tr>
<tr>
<td>43</td>
<td>4.178344</td>
<td>66.220.147.33</td>
<td>128.54.1.79</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>45</td>
<td>4.390254</td>
<td>128.54.1.79</td>
<td>66.220.147.33</td>
<td>TCP</td>
<td>53792 &gt; http [ACK] Seq=558 Ack=1461 Win=64240 Len=0</td>
</tr>
<tr>
<td>46</td>
<td>4.407933</td>
<td>66.220.147.33</td>
<td>128.54.1.79</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>47</td>
<td>4.417271</td>
<td>128.54.1.79</td>
<td>128.239.0.252</td>
<td>DNS</td>
<td>Standard query A static.ak.fbcdn.net</td>
</tr>
<tr>
<td>48</td>
<td>4.419163</td>
<td>128.239.0.252</td>
<td>128.54.1.79</td>
<td>DNS</td>
<td>Standard query response CNAME static.ak.facebook.com.edgesuit</td>
</tr>
<tr>
<td>49</td>
<td>4.419929</td>
<td>128.54.1.79</td>
<td>128.239.0.252</td>
<td>DNS</td>
<td>Standard query AAAA static.ak.fbcdn.net</td>
</tr>
<tr>
<td>50</td>
<td>4.421474</td>
<td>128.239.0.252</td>
<td>128.54.1.79</td>
<td>DNS</td>
<td>Standard query response CNAME static.ak.facebook.com.edgesuit</td>
</tr>
</tbody>
</table>

Frame 43 (1514 bytes on wire, 1514 bytes captured)
- Ethernet II, Src: Cisco_4a:70:00 (00:21:56:4a:70:00), Dst: IntelCor_ae:1b:2b (00:1f:3b:a:1:b:2b)
- Internet Protocol, Src: 66.220.147.33 (66.220.147.33), Dst: 128.54.1.79 (128.54.1.79)
- Transmission Control Protocol, Src Port: http (80), Dst Port: 53792 (53792), Seq: 1, Ack: 558, Len: 1460

0000 00 1f 3b ae 1b 2b 00 21 56 4a 70 00 08 00 45 00 ...:++! VJp...E.
0010 05 dc 3a 1d 40 00 f3 06 f0 7b 42 dc 93 21 80 36 ...@...{B..16
0020 01 4f 00 50 d2 20 a8 03 b6 f8 7c 64 82 09 50 18 ...O.P.....,.P
0030 33 49 13 39 00 00 48 54 50 50 3f 31 30 30 32 31 13 ...I, OK HT 1.12
0040 30 30 20 4f 4b 0d 0a 43 61 63 68 65 2d 43 6f 6e 00 OK...c ache-Con
0050 74 72 6f 6c 3a 20 71 70 72 69 76 61 74 65 5c 20 6e trl: pr iate, n
0060 6f 2d 63 61 66 65 65 2c 62 20 6f 6d 2d 73 74 6f 72 o-cache, no-stor
0070 65 2c 20 2d 65 73 74 73 2d 2d 72 65 76 61 6c 69 64 e, must-revalidated
0080 74 65 6d 0a 45 78 70 70 69 72 65 76 33 20 53 31 74 te..Expires: Sat
0090 2c 20 30 30 30 30 30 30 30 30 30 30 30 30 30 01 Jan 2000 00

File: C:\Users\kanich\Documents\123.test 38...
Packets: 702 Displayed: 629 Marked: 0
Global Network Addressing

- Address each packet so it can traverse network and arrive at host

My computer (132.239.9.64)

www.google.com (66.102.7.104)

Destination | Source | Data
|---|---|---
66.102.7.104 | 132.239.9.64 | 1 GET htt
Project Overview
Project Objective

- Goal: Facilitate communication between sender and receiver threads over a lossy, unreliable communication channel.

- Your task: Implement a version of the Sliding Window Protocol (SWP) for communication between threads
  - See Sections 2.4 and 2.5 of P&D
Recapping Threads

- Wikipedia definition: The smallest unit of processing that can be scheduled by the OS

- Multiple threads can run inside the same process and share memory
  - Protect critical sections using mutexes/condition variables/etc

- FYI, how to create a thread in C: pthread_create
  - Arguments passed in:
    - pthread data type
    - IGNORE
    - function name
    - function input parameter
Project Analogy

Sending Host

Input: “Hello World”

Physical/Wireless Tube

Receiving Host

Output: “Hello World”

Sending Thread

Shared memory

Receiving Thread
Project Analogy (cont)

- **Sender thread**: A node attempting to transmit data over a link
- **Receiver thread**: A node waiting to receive data over the link

- What problems do nodes face when transmitting data?
  - Packet losses
  - Data corruption
  - Collisions
Sender threads: Take input from command line and transmit to receivers
  - `msg <sender_id> <receiver_id> <stdin message>`
  - `msg 0 1 hello world!`

Receiver threads: Print out messages intended for them
  - `<RECV_1>[hello world!]`
Project Overview (diagram)
You will have to handle:

- Message drops
- Corrupted messages
- Message buffering/segmenting
- Multiple outstanding messages

To tackle this project:
- Read P&D Section 2.4/2.5
Resource Allocation: Queues

- Sharing access to limited resources
  - E.g., a link with fixed service rate
- Simplest case: first-in-first out queue
  - Queue/serve packets in the order they arrive
  - Drop packets when the queue is full
- Anybody hear of “Network Neutrality”?