Lower-level networking issues

11/4/15
Questions on chat client?
Would it be good for me to do the select lab in class?

A. Absolutely yes
B. 
C. 
D. 
E. Absolutely not
Announcement

• For Friday: Read Chapter 4 (UDP)
Numeric representations

(Section 5.1 in the text)
Endianness

• How do we send a 4-byte number (aka an int)?

```c
int val = 0x01020304;  //int to send
send(socket, &val, 4);  //send 4 bytes
```

• The byte at &val is sent first; which is it?
  • 0x01, (called “big-endian”)
  • 0x04, (called “little-endian”; includes x86)
  • or something else?
How do we deal with endianness?

A. Standardize on one format; all numbers sent over the network use that format. The OS converts as necessary

B. Standardize, but with the application responsible for converting

C. Each system sends in its own format, but messages indicate the sender’s numeric format. The OS uses this to convert to receiver’s format

D. Assume that incompatible systems won’t communicate

E. Assume that numbers don’t need to go over the network
How do we deal with endianness?

A. Standardize on one format; all numbers sent over the network use that format. The OS converts as necessary

B. Standardize, but with the application responsible for converting

C. Each system sends in its own format, but messages indicate the sender’s numeric format. The OS uses this to convert to receiver’s format

D. Assume that incompatible systems won’t communicate

E. Assume that numbers don’t need to go over the network
What about floating point numbers?
OSI reference model

- **Application**: Specific applications: http, ssh, mail, ftp, ...
- **Presentation**: Manages data structures instead of bits (format translation) Not really used
- **Session**: Turns connections into sessions (allows pausing and resuming) Not really used
- **Transport**: Break messages into packets and ensures delivery standard (Some overlap w/ data link, but for more than 1 channel)
- **Network**: Packet routing
- **Data link**: Provide line without transmission errors Error detection, flow control
- **Physical**: Raw bits over a channel
  - How many volts represent 0/1? What is duration of a bit?
Physical layer
Representing a bit

• Use different voltage levels:

1 0 1 0
Representing a bit

• Use different voltage levels:

0.85V

-0.85V

1 0 1 0
Representing a bit

• Use different voltage levels:

![Diagram showing voltage levels]

• Issue: What does the following encode?
Manchester encoding

• Use a two-part waveform for each bit:

```
1 = high first part, low second part
0 = low first part, high second part
```
What does the following encode in the Manchester encoding?

A. 11011100
B. 11001000
C. 11101100
D. 00110100
E. 00011100
What does the following encode in the Manchester encoding?

A. 11011100
B. 11001000
C. 11101100
D. 00110100
E. 00011100
Differential Manchester encoding

• Now base waveform on previous bit:

\[\begin{array}{c}
1 \text{ if was high} & 1 \text{ if was low} \\
1 = \text{ remain same at beginning} & \\
0 = \text{ switch at beginning} \\
\text{(both bits switch in the middle)}
\end{array}\]
What does the following encode in the differential Manchester encoding?

(assume high before this)

A. 10010110
B. 10001110
C. 10110010
D. 01000101
E. 01101001
What does the following encode in the differential Manchester encoding?

(assume high before this)

A. 10010110
B. 10001110
C. 10110010
D. 01000101
E. 01101001
Tradeoff with these encodings

• Advantages:
  – easier to count long strings of 0s or 1s
  – apparently better noise resistance

• Disadvantages:
  – more complicated receiving logic
  – need to switch twice as fast (causes more noise)

• Ethernet uses Manchester
Data Link Layer
Who talks when?

- Historical system: ALOHA [Abrahson et al. 1970s]

- Want to connect computers on the islands using radio (predates cable connections...)

- Broadcast medium: Everyone can hear messages, but simultaneous transmissions get garbled
Simplest possible protocol

• Transmit if you have something to say
• If message gets garbled, stop and try later
Simplest possible protocol

- Transmit if you have something to say
- If message gets garbled, stop and try later
- When will 2 messages collide (i.e. interfere)?

blue sent by app

time
Simplest possible protocol

- Transmit if you have something to say
- If message gets garbled, stop and try later

- When will 2 messages collide (i.e. interfere)?

  blue sent by app
  
  collides if red sends during or too soon before blue
Improvement: Organize time

- Divide time into “slots”
- Transmitter waits until next slot
Improvement: Organize time

• Divide time into “slots”
• Transmitter waits until next slot
collides if red decides to send during this interval
Improvement: Listen before sending

• Listen before sending to see if anyone else is transmitting ("carrier sense protocol")
• Can still have collisions due to propagation delay (time to reach farthest receiver)
How long should system wait to retransmit?

A. A specific short time (e.g. 2 message lengths)
B. A specific long time (e.g. 1,000 message lengths)
C. A time selected uniformly at random with a small maximum (e.g. 0, 1, or 2 message lengths)
D. A time selected uniformly at random with a large maximum (e.g. 0-1,000 message lengths)
E. Give up on the whole idea of retransmissions and assume no conflicts
Adaptive retransmission

• Start with small range and grow it as more collisions occur
  – 1\textsuperscript{st} collision: delay of 0 or 1
  – 2\textsuperscript{nd} collision: delay of 0, 1, 2, or 3
  – ...
  – 10\textsuperscript{th} collision: delay of 0-1023
  – 11\textsuperscript{th} collision: delay of 0-1023
  – give up after 16 consecutive collisions

• “Exponential backoff”
Networking layer: IP
IPv4 header

<table>
<thead>
<tr>
<th>Version #</th>
<th>Header Length</th>
<th>Service class</th>
<th>Packet length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to live (hops)</td>
<td>Protocol #</td>
<td>Header checksum</td>
<td></td>
</tr>
</tbody>
</table>

Source address

Destination address

Followed by 0–10 words of options
# IPv4 header

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version #</td>
<td>0</td>
</tr>
<tr>
<td>Header Length</td>
<td>4</td>
</tr>
<tr>
<td>Service class</td>
<td>8</td>
</tr>
<tr>
<td>Packet length (bytes)</td>
<td>16</td>
</tr>
<tr>
<td>Segment ID</td>
<td></td>
</tr>
<tr>
<td>Time to live (hops)</td>
<td></td>
</tr>
<tr>
<td>Protocol #</td>
<td></td>
</tr>
<tr>
<td>Header checksum</td>
<td></td>
</tr>
<tr>
<td>Source address</td>
<td></td>
</tr>
<tr>
<td>Destination address</td>
<td></td>
</tr>
</tbody>
</table>

Followed by 0–10 words of options