CSCI-1680
Physical Layer
Link Layer I

Rodrigo Fonseca

Based partly on lecture notes by David Mazières, Phil Levis, John Jannotti
Administrivia

- **Snowcast milestone today**
  - “Last commit before midnight”
  - Schedule your milestone meeting
Today

• **Physical Layer**
  – Modulation and Channel Capacity
  – Encoding

• **Link Layer I**
  – Framing
Layers, Services, Protocols

Application
Service: user-facing application.
Application-defined messages

Transport
Service: multiplexing applications
Reliable byte stream to other node (TCP),
Unreliable datagram (UDP)

Network
Service: move packets to any other node in the network
IP: Unreliable, best-effort service model

Link
Service: move frames to other node across link.
May add reliability, medium access control

Physical
Service: move bits to other node across link
Physical Layer (Layer 1)

- Responsible for specifying the physical medium
  - Type of cable, fiber, wireless frequency
- Responsible for specifying the signal (modulation)
  - Transmitter varies *something* (amplitude, frequency, phase)
  - Receiver samples, recovers signal
- Responsible for specifying the bits (encoding)
  - Bits above physical layer -> *chips*
Modulation

• Specifies mapping between digital signal and some variation in analog signal

• Why not just a square wave (1v=1; 0v=0)?
  – Not square when bandwidth limited

• Bandwidth – frequencies that a channel propagates well
  – Signals consist of many frequency components
  – Attenuation and delay frequency-dependent
Components of a Square Wave

Graphs from Dr. David Alciatore, Colorado State University
Approximation of a Square Wave

individual harmonics

\[ \omega_0 \]
amp = \frac{4}{\pi}

\[ 3\omega_0 \]
amp = \frac{4}{3\pi}

\[ 5\omega_0 \]
amp = \frac{4}{5\pi}

combined harmonics

\[ \omega_0 \]
amp = 0

\[ \omega_0, 3\omega_0, 5\omega_0, \ldots, \infty \omega_0 \]
Idea: Use Carriers

- Only use frequencies that transmit well
- *Modulate* the signal to encode bits

**OOK: On-Off Keying**

**ASK: Amplitude Shift Keying**
Idea: Use Carriers

- Only use frequencies that transmit well
- *Modulate* the signal to encode bits

**FSK: Frequency Shift Keying**

**PSK: Phase Shift Keying**
How Fast Can You Send?

• Encode information in some varying characteristic of the signal.

• If B is the maximum frequency of the signal

\[
C = 2B \text{ bits/s}
\]

(Nyquist, 1928)
Can we do better?

• So we can only change 2B/second, what if we encode more bits per sample?
  – Baud is the frequency of changes to the physical channel
  – Not the same thing as bits!

• Suppose channel passes 1KHz to 2KHz
  – 1 bit per sample: alternate between 1KHz and 2KHz
  – 2 bits per sample: send one of 1, 1.33, 1.66, or 2KHz
  – Or send at different amplitudes: A/4, A/2, 3A/4, A
  – n bits: choose among $2^n$ frequencies!

• What is the capacity if you can distinguish M levels?
Example

256-QAM Constellation

Amplitude

Phase
Hartley’s Law

\[ C = 2B \log_2(M) \text{ bits/s} \]

Great. By increasing \( M \), we can have as large a capacity as we want!

Or can we?
The channel is noisy!
The channel is noisy!

• Noise prevents you from increasing M arbitrarily!
• This depends on the signal/noise ratio (S/N)
• Shannon: $C = B \log_2(1 + S/N)$
  – C is the channel capacity in bits/second
  – B is the bandwidth of the channel in Hz
  – S and N are average signal and noise power
  – Signal-to-noise ratio is measured in dB = $10\log_{10}(S/N)$
Putting it all together

• **Noise limits M!**

\[2B \log_2(M) \leq B \log_2(1 + S/N)\]

\[M \leq \sqrt{1+S/N}\]

• **Example: Telephone Line**
  
  – 3KHz b/w, 30dB S/N = \(10^{(30/10)} = 1000\)
  
  – \(C = 3\text{KHz} \log_2(1001) \approx 30\text{Kbps}\)
Encoding

• Now assume that we can somehow modulate a signal: receiver can decode our binary stream.

• How do we encode binary data onto signals?

• One approach: 1 as high, 0 as low!
  – Called Non-return to Zero (NRZ)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NRZ (non-return to zero)

Clock
Drawbacks of NRZ

- No signal could be interpreted as 0 (or vice-versa)
- Consecutive 1s or 0s are problematic
- Baseline wander problem
  - How do you set the threshold?
  - Could compare to average, but average may drift
- Clock recovery problem
  - For long runs of no change, could miscount periods
Alternative Encodings

• **Non-return to Zero Inverted (NRZI)**
  – Encode 1 with transition from current signal
  – Encode 0 by staying at the same level
  – At least solve problem of consecutive 1s
Manchester

- Map 0 $\rightarrow$ chips 01; 1 $\rightarrow$ chips 10
  - Transmission rate now 1 bit per two clock cycles
- Solves clock recovery, baseline wander
- But cuts transmission rate in half
4B/5B

• Can we have a more efficient encoding?
• Every 4 bits encoded as 5 chips
• Need 16 5-bit codes:
  – selected to have no more than one leading 0 and no more than two trailing 0s
  – *Never get more than 3 consecutive 0s*
• Transmit chips using NRZI
• Other codes used for other purposes
  – E.g., 11111: line idle; 00100: halt
• Achieves 80% efficiency
### 4B/5B Table

<table>
<thead>
<tr>
<th></th>
<th>0000</th>
<th>11110</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0001</td>
<td>01001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>10100</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>10101</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>01010</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>01011</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>01110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>01111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>10010</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>10011</td>
</tr>
<tr>
<td>A</td>
<td>1010</td>
<td>10110</td>
</tr>
<tr>
<td>B</td>
<td>1011</td>
<td>10111</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
<td>11010</td>
</tr>
<tr>
<td>D</td>
<td>1101</td>
<td>11011</td>
</tr>
<tr>
<td>E</td>
<td>1110</td>
<td>11100</td>
</tr>
<tr>
<td>F</td>
<td>1111</td>
<td>11101</td>
</tr>
</tbody>
</table>
Encoding Goals

- DC Balancing (same number of 0 and 1 chips)
- Clock synchronization
- Can recover some chip errors
- Constrain analog signal patterns to make signal more robust
- Want near channel capacity with negligible errors
  - Shannon says it’s possible, doesn’t tell us how
  - Codes can get computationally expensive
- In practice
  - More complex encoding: fewer bps, more robust
  - Less complex encoding: more bps, less robust
Last Example: 802.15.4

- Standard for low-power, low-rate wireless PANs
  - Must tolerate high chip error rates
- Uses a 4B/32B bit-to-chip encoding
Questions?

Photo: Lewis Hine
Next Week

• Next week: more link layer
  – Flow Control and Reliability
  – Ethernet
  – Sharing access to a shared medium
  – Switching

• Next Thursday: HW1 out