CSCI-1680

Physical Layer

Link Layer I

Nick DeMarinis

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

• Snowcast milestone due Friday (Feb 4) by 11:59pm EDT
  – Implementation
  – Design of server

• Office hours
  – All hours are hybrid format (except those marked “group”)
  – Join queue online, get help in-person or via zoom

• Waitlist
  – If you have a pending override code, you’re safe
  – Next batch: Tomorrow morning
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)

• Specifies the physical medium
  – Type of cable, fiber, wireless frequency

• Specifies the signaling/modulation
  – Transmitter varies *something* (amplitude, frequency, phase)
  – Receiver samples, recovers signal => gets back 1’s and 0’s

• Specifies the *encoding*
  – How we turn the signals into *meaningful* 1’s and 0’s
Why should we care?

• This is the line between electrical engineering and computer science

• Useful to know the overall limitations of a medium
  – Requirements on bandwidth, noise…
Modulation

• Modulation: how to vary a signal in order to transmit information

• Why not just a square wave?
  – Not square when bandwidth limited

• Bandwidth – frequencies that a channel propagates well
  – Signals consist of many frequency components
  – Signal characteristics for physical media are frequency-dependent
Components of a Square Wave
Approximation of a Square Wave

Graphs from Dr. David Alciatore, Colorado State University
Idea: Use Carriers

Start with a carrier frequency, modulate it to encode data:

**OOK: On-Off Keying**

1 0 1

**ASK: Amplitude Shift Keying**

1 0 1
Early IEEE 802.11 (Wifi) channel bandwidth
Early IEEE 802.11 (Wifi) channel bandwidth
How Fast Can You Send?

Encode information in some varying characteristic of the signal, but need to recover it on the other end

Nyquist’s Theorem: 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**: 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?
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: $C = B \log_2(1 + S/N)$
- $C$: channel capacity in bits/second
- $B$: bandwidth in Hz
- $S$, $N$: average signal, noise power
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 has 3KHz BW, 30dB SNR

– \( S/N = 10^{(30 \text{ dB}/10)} = 1000 \)
– \( C = 3\text{KHz} \log_2(1 + 1000) \approx 30\text{Kbps} \)
– \( M < \sqrt{1001} \approx 31 \text{ levels} \)

Signal-to-noise ratio (SNR) is typically measured in Decibels (dB)
\[ \text{dB} = 10 \log_{10}(S/N) \]
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)
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

![NRZI (non-return to zero inverted) clock signal](image-url)
Manchester Encoding

• Map 0 → 01; 1 → 10
  – Transmission rate now 1 bit per two clock cycles
• Solves clock recovery & baseline wander
• … but halves transmission rate!

![Manchester Encoding Diagram](image-url)
Can we have a more efficient encoding?

• Every 4 bits encoded as 5 *chips*
• Carefully select 16 5-bit chips we can transmit using NRZI
  – No more than one leading 0 and no more than two trailing 0s
  – *Never get more than 3 consecutive 0s*
• 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 class: link layer
- Thursday: HW1 out