Announcements

• Reminder:
  – Project 1 is due on Thursday.
Network Architecture

• General blueprints that guide the design and implementation of networks
• Goal: to deal with the complex requirements of a network
• Use *abstraction* to separate concerns
  – Identify the useful service
  – Specify the interface
  – Hide the implementation
Layering

• A result of abstraction in network design
  – A stack of services (layers)
  – Hardware service at the bottom layer
  – Higher level services are implemented by using services at lower levels

• Advantages
  – Decompose problems
  – Modular changes

```
Application
  ↓
Error Control
  ↓
Routing
  ↓
Hardware
```
Protocols

• A protocol is a specification of an interface between modules (often on different machines)

• Sometimes “protocol” is used to mean the implementation of the specification.
Example Protocol Stack

Application Programs
Request / Reply Channel  Message Stream Channel
Host-to-Host Connectivity
Hardware
Protocol Interfaces

- **Service Interfaces**
  - Communicate up and down the stack
- **Peer Interfaces**
  - Communicate to counterpart on another host
Example Protocol Graph
Encapsulation
Internet Protocol Graph

- FTP
- HTTP
- NTP
- VOIP

- TCP
- UDP

- IP

- Ethernet
- ATM
- FDDI
Open Systems Interconnection (OSI)

End Host
- Reference model – not actual implementation.

Application
- Transmits *messages* (e.g. FTP or HTTP)

Presentation
- Data format issues (e.g. big- vs. little-endian)

Session
- Manages multiple streams of data

Transport
- Process to process protocols

Network
- Routes *packets* among nodes in network

Data Link
- Packages bit streams into *frames*

Physical
- Transmits raw bits over link
Open Systems Interconnection (OSI)

**End Host**
- Reference model – not actual implementation.

**Application**
- Transmits *messages* (e.g. FTP or HTTP)

**Presentation**
- Data format issues (e.g. big- vs. little-endian)

**Session**
- Manages multiple streams of data

**Transport**
- Process to process protocols

**Network**
- Routes *packets* among nodes in network

**Data Link**
- Packages bit streams into *frames*

**Physical**
- Transmits raw bits over link
Network adapters encode streams of bits into signals.

Simplification: Assume two discrete signals—high and low.

Practice: Two different voltages on copper link or different brightness of light on fiber link.
(leads to some interesting encoding issues)
\[ \nabla \cdot \vec{E} = \varepsilon_0 \rho \]
\[ \nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \]
\[ \nabla \cdot \vec{B} = 0 \]
\[ \nabla \times \vec{B} = \mu_0 \vec{J} + \mu_0 \varepsilon_0 \frac{\partial \vec{E}}{\partial t} \]
Open Systems Interconnection (OSI)

- **End Host**
  - Reference model – not actual implementation.

- **Application**
  - Transmits *messages* (e.g. FTP or HTTP)

- **Presentation**
  - Data format issues (e.g. big- vs. little-endian)

- **Session**
  - Manages multiple streams of data

- **Transport**
  - Process to process protocols

- **Network**
  - Routes *packets* among nodes in network

- **Data Link**
  - Packages bit streams into *frames*

- **Physical**
  - Transmits raw bits over link
Framing

• Need a way to send blocks of data.
  – How does the network adapter detect when the sequence begins and ends?
  – Are there transmission errors in the data?

• *Frames* are link layer unit of data transmission
  – Byte oriented vs. Bit oriented
  – Point-to-point (e.g. PPP) vs. Multiple access (Ethernet)
A Multi-access, Bit-oriented Protocol

- Frames contain sequences of bits
  - Could be ASCII
  - Could be pixels from an image
- Frames read by many nodes
  - Address distinguishes intended recipient

- HDLC (High-level Data Link Control)
  - Begin and ending = 01111110
  - Uses bit stuffing: suffix five 1’s with a 0

```
<table>
<thead>
<tr>
<th>Begin</th>
<th>Header</th>
<th>Body</th>
<th>CRC</th>
<th>Ending</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>16</td>
<td></td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>
```

HDLC frame format
Problem: Error Detection & Correction

- Bit errors may be introduced into frames
  - Electrical interference
  - Thermal noise
- Could flip one bit or a few bits independently
- Could zero-out or flip a sequence of bits (burst error)

- How do you detect an error?

- What do you do once you find one?
Error Detection

• General principal: Introduce redundancy

• Trivial example: send two copies
  – High overheads: 2n bits to send n
  – Won’t detect errors that corrupt same bits in both copies

• How can we do better?
  – Minimize overhead
  – Detect many errors
  – General subject: error detecting codes
Simple Error Detection Schemes

- **Parity**
  - 7 bits of data
  - 8th bit is sum of first seven bits mod 2
  - Overhead: 8n bits to send 7n
  - Detects: any odd number of bit errors

- **Internet Checksum algorithm**
  - Add up the words of the message, transmit sum
  - 16 bit ones-complement addition
  - Overhead: 16 bits to send n
  - Does not detect all two bit errors
Cyclic Redundancy Check

• Reading: Wikipedia entry on CRC
• Used in link-level protocols
  – CRC-32 used by Ethernet, 802.5, PKzip, …
  – CRC-CCITT used by HDLC
  – CRC-8, CRC-10, CRC-32 used by ATM

• Better than parity or checksum
  – (e.g. 32 bits to send 12000)
• Simple to implement
Cyclic Redundancy Check (CRC)

- Consider (n+1)-bit message as a n-degree polynomial
  - Polynomial arithmetic modulo 2
  - Bit values of message are coefficients
  - Message = 10011010
  - Polynomial
    \[ M(z) = (1 \cdot z^7) + (0 \cdot z^6) + (0 \cdot z^5) + (1 \cdot z^4) + (1 \cdot z^3) +
    (0 \cdot z^2) + (1 \cdot z^1) + (0 \cdot z^0) \]
    \[ = z^7 + z^4 + z^3 + z^1 \]
Cyclic Redundancy Check

• Sender and receiver agree on a *divisor polynomial* $C(z)$ of degree $k$
  – Example $k = 3$
  – $C(z) = z^3 + z^2 + 1$
  – Coefficients are 1101

• Error correction bits are remainder of
  $$(M(z) \cdot z^k) \text{ divided by } C(z)$$

• This yields a $n+k$ bit transmission polynomial $P(z)$ that is *exactly* divisible by $C(z)$
Example CRC Calculation

Original message: \( M(z) \)

Divisor Polynomial: \( C(z) \)

Multiplication by \( z^3 \)

Remainder

...
Example CRC Calculation

\[ Z^3 \cdot \text{Original Message } M(z) = \begin{array}{c} \text{10011010} \\
\end{array} \begin{array}{c} \text{000} \\
\end{array} \]

\[ \text{Remainder} = + \begin{array}{c} \text{101} \\
\end{array} \]

\[ \text{Transmitted message } P(z) = \begin{array}{c} \text{10011010} \\
\end{array} \begin{array}{c} \text{101} \\
\end{array} \]

- Recipient checks that \( C(z) \) evenly divides the received message.
CRC Error Detection

- Must choose a good divisor \( C(z) \)
  - There are many standard choices: CRC-8, CRC-10, CRC-12, CRC-16, CRC-32
  - CRC-32: 0x04C11DB7

- All 1-bit errors as long as \( z^k \) and \( z^0 \) coefficients are 1
- All 2-bit errors as long as \( C(z) \) has three terms
- Any odd number of errors if \( (z+1) \) divides \( C(z) \)
- Any burst errors of length \( \leq k \)
CRC Implementations

- Easy to implement in hardware
  - Base 2 subtraction is XOR
  - Simple k-bit shift register with XOR gates inserted before 1’s in $C(z)$ polynomial
  - Message is shifted in, registers fill with remainder

- Example $C(z) = 1101$

![CRC Diagram](image)
Error Correction Codes

- Redundant information can be used to correct some errors.
- Typically requires more redundancy.

- Tradeoffs:
  - Error detection requires retransmission.
  - Error correction sends more bits all the time.

- Forward Error Correction is useful:
  - When errors are likely (e.g. wireless network).
  - When latency is too high for retransmission (e.g. satellite link).
Open Systems Interconnection (OSI)

End Host

Application
- Reference model – not actual implementation.
- Transmits *messages* (e.g. FTP or HTTP)

Presentation
- Data format issues (e.g. big-vs. little-endian)

Session
- Manages multiple streams of data

Transport
- Process to process protocols

Network
- Routes *packets* among nodes in network

Data Link
- Packages bit streams into *frames*

Physical
- Transmits raw bits over link
IEEE 802 network standards

The IEEE 802 committee produces standards & specifications for Local Area Networks (LAN):

- **802.3 CSMA/CD Networks (Ethernet)**
- **802.4 Token Bus Networks**
- **802.5 Token Ring Networks**
- **802.6 Metropolitan Area Networks**
- **802.11 Wireless LAN (Wifi) [Thursday]**
Ethernet (802.3)

• A standard for local area networks (LAN)

• Developed in mid-70’s at Xerox PARC
  – Descendent of Aloha, a U. of Hawaii radio packet network
  – DEC, Intel, and Xerox standard: 1978 for 10Mbps
  – IEEE 802.3 standard grew out of that

• Physical implementations:
  – 10Base5, 10BaseT, 100BaseT, 1000BaseT…
  – Speed: 10Mbps, 100Mbps, 1000Mbps, …
Ethernet Physical links

- Originally used “Thick-net” 10Base5
  - 10 = 10Mbps
  - 5 = maximum of 500 meters segments
  - Up to 4 repeaters between two hosts
    =2500m max

- More common: 10BaseT
  - 10 = 10Mbps
  - T = Twisted pair (typically Category 5),
    Maximum of 100 meter segments
  - Connected via hubs (still 2500m max)

- Today’s standards: 100BaseT, 1000BaseT
Ethernet topologies

10Base5 topology

10BaseT topology

Host

Repeater

Hub

Hub
How the ethernet works

• The Ethernet link is *shared*
  – A signal transmitted by one host reaches *all* hosts
• Method of operation: **CSMA/CD**
  – Carrier Sense, Multiple Access, with Collision Detection
• Hosts competing for the same link are said to be in the same *collision domain*
  – Good news: easy to exchange data
  – Bad news: have to regulate link access
• Protocol: *Media Access Control (MAC)*
Ethernet Addresses

• Every adapter manufactured has a unique address
  – 6 bytes (48 bits) usually written in Hex.
  – Examples: 00-40-50-B1-39-69 and 8:0:2b:e4:b1:2
  – Each manufacturer is assigned 24-bit prefix
  – Manufacturer ensures unique suffixes
Ethernet Frame Format

- Preamble – repeating pattern of 0’s & 1’s
  - Used by receiver to synchronize on signal
- Dest and Src – Ethernet Addresses
- Type – demultiplexing key
  - Identifies higher-level protocol
- Body – payload
  - Minimum 46 Bytes
  - Maximum 1500 Bytes
Addresses in an ethernet frame

• All bits = 1 indicates a broadcast address
  – Sent to all adapters

• First bit = 0 indicates unicast address
  – Sent to only one receiver

• First bit = 1 indicates multicast address
  – Sent to a group of receivers
An Ethernet Adapter Receives:

- Frames addressed to the broadcast address
- Frames addressed to its own address
- Frames sent to a multicast address
  - If it has been programmed to listen to that address
- All frames
  - If the adapter has been put into *promiscuous mode*
Ethernet Transmitter Algorithm

• If the link is idle transmit the frame immediately
  – Upper bound on frame size means adapter can’t hog the link

• If the link is busy
  – Wait for the line to go idle
  – Wait for 9.6\,\mu s after end of last frame (sentinel)
  – Transmit the frame

• Two (or more) frames may collide
  – Simultaneously sent frames interfere
Collision Detection

• When an adapter detects a collision
  – Immediately sends 32 bit jamming signal
  – Stops transmitting

• A 10MBps adapter may need to send 512 bits in order to detect a collision
  – Why?
    – 2500m + 4 repeaters gives RTT of 51.2\(\mu\)s
    – 51.2\(\mu\)s at 10Mbps = 512 bits
  – Fortunately, minimum frame (excluding preamble) is 512 bits = 64 bytes
    • 46 bytes data + 14 bytes header + 4 bytes CRC
Ethernet Collision (Worst Case)

T=0

25.6μs

51.2μs
Exponential Backoff

• After it detects 1st collision
  – Adapter waits either 0 or 51.2\(\mu\)s before retrying
  – Selected randomly

• After 2nd failed transmission attempt
  – Adapter randomly waits 0, 51.2, 102.4, or 153.6\(\mu\)s

• After n\(\text{th}\) failed transmission attempt
  – Pick k in 0 … 2\(\text{n}-1\)
  – Wait k x 51.2\(\mu\)s
  – Give up after 16 retries
    (but cap n at 10)
Ethernet Security Issues

• Promiscuous mode
  – *Packet sniffer* detects all Ethernet frames

• Less of a problem in *switched* Ethernet
  – Why?