Error Detection, Multiple Access, Link Layer Addressing

CS 640, 2015-02-03

Announcements

- Quiz #1 on Thursday
- Assignment #1 due in 1 week

Outline

- Error Detection
- Access control
- Ethernet
- Link Layer Addressing

Link Layer

- Framing -- where does one packet begin and end (in a signal of bits)?
    - Discussed last week -- look for special characters (at fixed time intervals)
- Error detection -- how do we identify errors in transmission?
- Access control -- when and who sends a signal?
    - Not usually a problem in modern wired NWs
    - Still a problem in wireless NWs -- they share frequency spectrum
- Queueing -- how to we store packets while waiting to send? what happens if the queue is full?
    - We'll talk about this right before spring break
- Bridging & switching -- how do we connect multiple links?
- Addressing -- how do we identify physical endpoints?

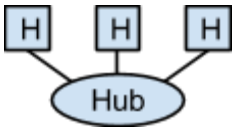
Error Detection

- ****Parity bits magic trick**
- Basic idea
    - Apply algorithm to packet (string of bits) to generate code (another string of bits)
    - Send packet and code
    - Receiver gets some packet and code (may not be correct because of errors)
    - Receiver applies same algorithm over packet it received and sees if the code it calculated matches the code it received
- ***What makes for a good error detection algorithm?***
    - Detect most errors
    - Minimal code size
    - Maybe correct errors
- Parity bits
    - Add an extra bit for each byte such that there is an even number of 1s
    - Two dimensional parity adds bits in both directions
        - Can catch all 1-, 2-, and 3-bit errors and most 4-bit errors
        - Requires 14-bits of parity for a 42-bit message

    - Disadvantages
        - Lots of overhead
        - Cannot detect more significant errors
- Checksum
    - Sum the bytes of a packet using ones-complement arithmetic
    - Advantages
        - Less overhead -- 16-bits for any length message
        - Easy to compute -- XOR 2 numbers and NOT the result
    - Disadvantage
        - Cannot detect errors which increment a word by one and decrement another word by one
    - Used at the network and transport layers
- Cyclic Redundancy Check
    - Uses modular arithmetic -- see the book for details
    - Advantages
        - Uses XOR which is easy to implement in hardware or software
        - Detects more errors than checksum
    - Used at the link layer to detect most errored packets
- Dealing with errors
    - Discard -- simple, but requires retransmission
    - Correct -- include error correcting codes; avoids retransmission, but adds overhead; more important when cost of retransmission is high (e.g., high latency link)

Access Control

- When and who sends a signal?
- Issue when multiple endpoints are in the same “collision domain”
    - Wireless -- multiple endpoints try to communicate using the same frequency spectrum
    - Half-duplex wired links -- can only send in one direction at a time
    - Wired hub -- hubs are dumb, they just repeat the signal



- Only 1 host at a time can transmit -- if 2+ hosts transmit simultaneously, the signals will collide and we’ll end up with noise
- ****How do we get hosts to take turns?***
    - Polling -- central controller asks who wants to send and tells each host when they can
    - Token passing -- pass a token host to host; host can send when it has the token
    - Random access -- try to send at a random time, and hope no other host picks the same; if you collide, pick another random time and try again
- Carrier Sense Multiple Access/Collision Detect (CSMA/CD)
    - Solution for wired networks with half
    - Required capabilities
        - Detect idle line
        - Detect collision
        - Avoid future collision (hopefully)

- Detect idle line -- physical layer has a special signal for idle
- Detect collision (assume half-duplex)
  - Host A starts transmitting; just before signal reaches host B, B starts to transmit
  - Signal from A will collide and collided signal will reach B
  - Collided signal must reach A before it has finished transmitting so it knows there was a collision

  - The following must hold: Round Trip Propagation Delay < Transmit Delay
    (2 * distance) / (2.3 * 10^8) < (size * 8) / (10^7)
    (2 / 2.3) * distance < size * 8 * 10
    distance < size * 92
  - 10Mbps Ethernet: max distance = 2500m; min packet size = 64 Bytes
    2500 < 5888
- Avoid future collision
  - Backoff for random amount of time -- multiple of transmit time for min size packet
    (64 * 8) / (10^7) = 51.2us
  - Double maximum possible backoff each time you collide
    - 1st backoff is either 0 or 51.2us
    - 2nd backoff is 0, 51.2, 102.4, or 153.6us
    - ...
  - Abort after 10 times

Ethernet

7B	1B	6B	6B	2B	46B - 1500B	4B
Preamble	Start of Frame	Dst MAC Address	Src MAC Address	Type	Payload	CRC

- Preamble -- alternating 0s & 1s; makes it easy to detect start of frame
- Start of frame -- special delimiter
- Dst/Src MAC address -- identifies dst/src hardware
  - Why do we need MAC addresses?
    - NIC can have multiple IP addresses, but only has one hardware address
    - Used to forward between intermediate hops -- IP header always specifies original src and final dst
- Type -- identifies next header; e.g., 0x0800 is IPv4, 0x86DD is IPv6, 0x0806 is ARP
- Payload -- min size is for CSMA/CD; max size is for fairness
- CRC -- used for error detection

Switches

- Connects multiple hosts without naively forwarding to all of them
- Relies on dst/src MAC addresses

Forwarding unplugged

Background

During this activity, each person will be functioning as a switch. You will be "connected" to some other switches and two hosts will be "connected" to you. In total, the network will have 5 switches (S1-S5) and 10 hosts (A-J). Packets are represented by pieces of paper, which list the source address (i.e., host letter) and destination address for the packet

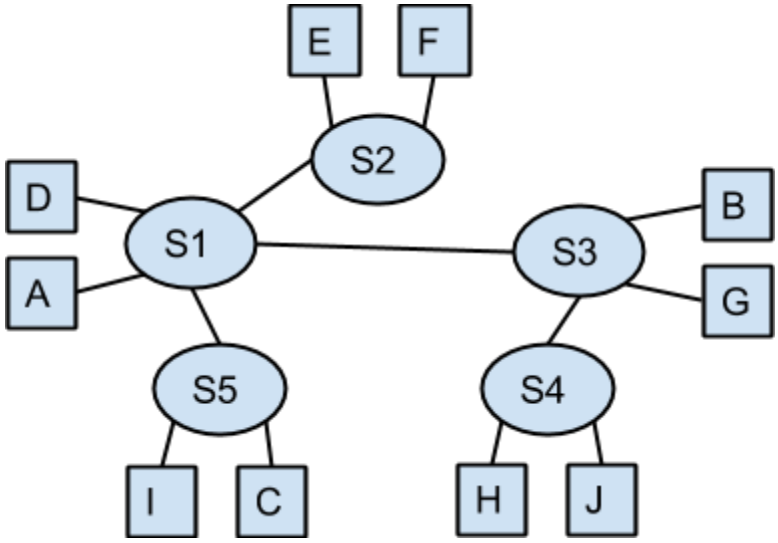
When you are functioning as a switch, there are a few rules:

1. You can only communicate with, and send packets to, directly connected switches.
2. You can duplicate a packet and send it to multiple switches.
3. You can generate and receive packets on behalf of the hosts that are connected to you

Part 1

First, discuss ideas for what algorithm each switch (i.e., each person) should execute to ensure packets reach their destination. Everyone in your group can talk at this point, but do not share information about how you are connected. In addition to selecting an algorithm, you should decide on three or four test packets that will be exchanged to see if your algorithm works; create those test packets now by creating a slip of paper for each packet and noting the source and destination of the packet on the paper.

Now, everyone in your group should function as a switch and execute the algorithm you agreed upon.



Forward your test packets from switch-to-switch (i.e., person-to-person) starting from the switch to which the source host is connected. After you have forwarded your test packets, raise your hand and we will provide a test packet to forward through your network.

Possible test packets:

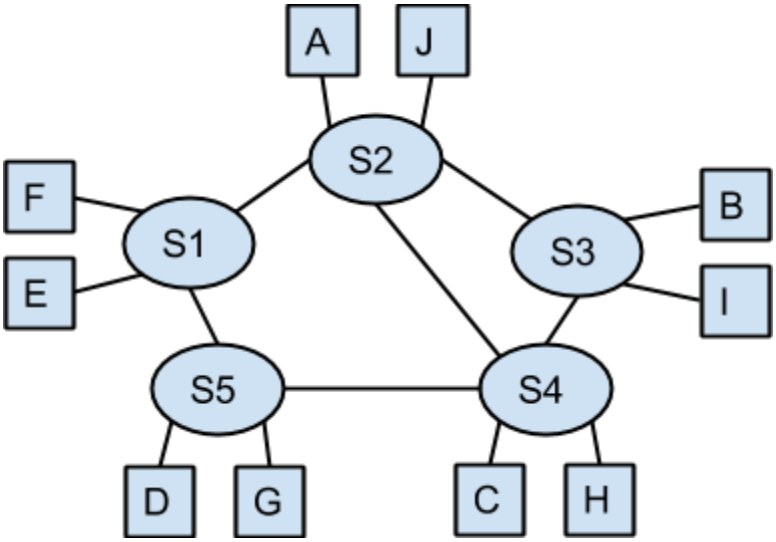
- F to G
- I to J

- H to D

Now, discuss as a group whether your algorithm worked. If not, what went wrong?

Part 2

Next, everyone in your group should function as a switch and execute a (potentially revised) algorithm. You can use the same test packets as before or agree upon some new test packets. However, the network has changed slightly, such that the following is now the setup:



After you have forwarded your test packets, raise your hand and we will provide a test packet to forward through your network.

Possible test packets:

- A to H
- D to B
- I to E

Now, discuss as a group whether your algorithm still worked. If not, what went wrong? What do you need to change? If you have time, try running this new algorithm with the same setup and test packets.