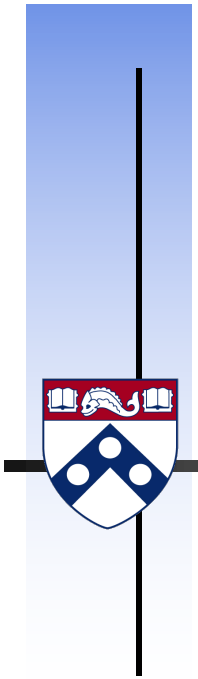


CIS 553: Networked Systems

Link Layer

February 3, 2020

Agenda

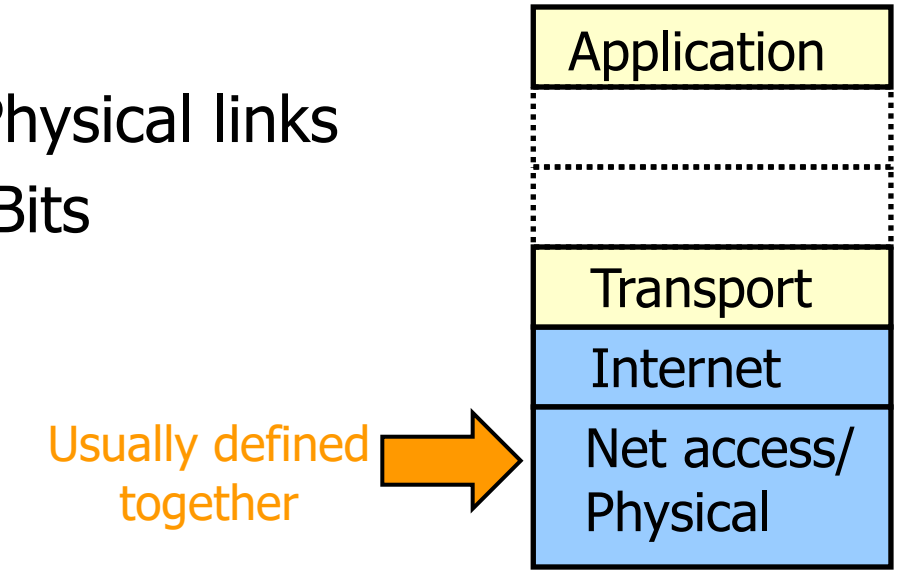
- Demultiplexing and the e2e principle ✓
- Physical Layer ✓
  - Media types ✓
  - Performance Metrics ✓
- Link Layer NEXT
  - Frame format and framing
  - Error detection
  - Medium access control
  - Switching

Review: Physical layer

- Connects: Transmitters and receivers

- Upper interface: Stream of bits
- Lower interface: N/A (photons/electrons?)

- Name of network: Physical links
- Name of message: Bits



Layer 2: Data link layer

- Connects: Physical interfaces
  - Possibly multiple
  - Usually all the same physical layer protocol
- Upper interface: A packet of data
- Lower interace: Stream of bits
- Name of network: Local Area Network (LAN)
- Name of message: Frame

Layer 2: Data link layer

- Connects: physical interfaces
  - Possibly multiple
  - Usually all the same physical layer protocol
- Upper interface: A packet of data
- Lower interace: Stream of bits
- Name of network: Local Area Network (LAN)
- Name of message: Frame

Data link layer

Provides four primary services

- Framing
  - Convert a stream of bits into messages
- Error detection and correction
  - Ensure that extracted frames haven't been corrupted
- Link access
  - Medium access control (MAC) defines when to transmit
- Reliable delivery (sometimes)

Ethernet "Frames"

- A message in Ethernet

- Preamble: 7 bytes for clock synchronization and 1 byte to indicate start of frame
- Addresses: 6 bytes
- Type: 2 bytes, higher-layer protocol (e.g., IP)
- Data payload: max 1500 bytes, min 46 bytes
- CRC: 4 bytes for error detection

Medium Access Control Address

- MAC address (e.g., 00:15:C5:49:04:A9)
  - Numerical address used within a link
  - Unique, hard-coded in the adapter when it is built
  - Flat name space of 48 bits
- Hierarchical allocation: Global uniqueness!
  - Blocks: assigned to vendors (e.g., Dell) by the IEEE
  - Adapters: assigned by the vendor from its block
- Broadcast address (i.e., FF:FF:FF:FF:FF:FF)
  - Send the frame to *all* adapters

As an aside: Promiscuous mode

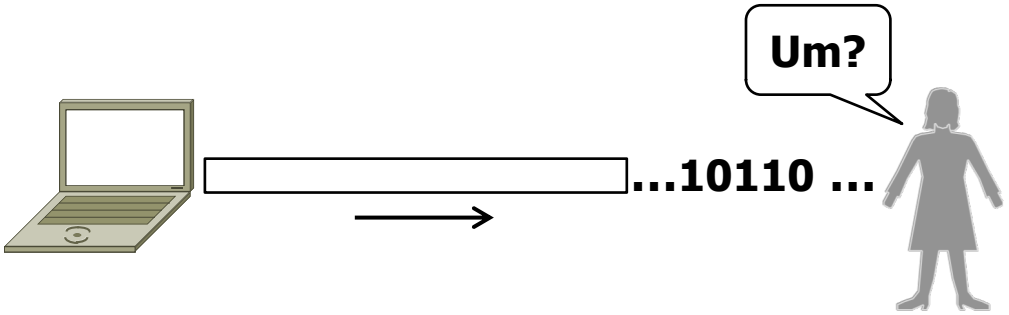
- Normal adapter: receives frames sent to
  - The local MAC address
  - Broadcast address FF:FF:FF:FF:FF:FF
- Promiscuous mode
  - Receive *everything*, independent of destination MAC
- Useful for packet sniffing
  - Network monitoring
  - E.g., wireshark, tcpdump

EtherType (demux key)

0x0800	Internet Protocol version 4 (IPv4)
0x0806	Address Resolution Protocol (ARP)
0x86DD	Internet Protocol Version 6 (IPv6)
0x8808	Ethernet flow control (PFCs)
0x88CC	Link Layer Discovery Protocol (LLDP)
0x88F7	Precision Time Protocol (PTP)
0x8906	Fibre Channel over Ethernet (FCoE)
0x8914	FCoE Initialization Protocol
0x8915	RDMA over Converged Ethernet (RoCE)

Framing frames

- Physical layer puts bits on a link
- Framing problem: how does the link layer determine where each frame begins and ends?



Framing methods

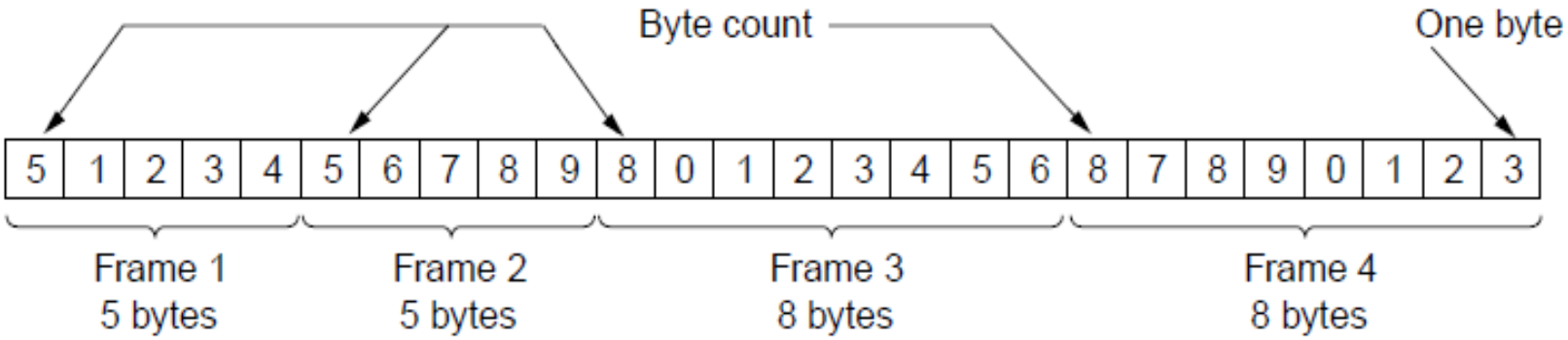
- We'll look at:
  - Byte count (motivation)
  - Byte stuffing
  - Bit stuffing
- In practice, the physical layer often helps to identify frame boundaries
  - E.g., if the signal on the wire goes flat, that's probably the end of a frame

Simple approach: byte count

- First try:
  - Let's start each frame with a length field!
  - It's simple, and hopefully good enough …

Simple approach: byte count

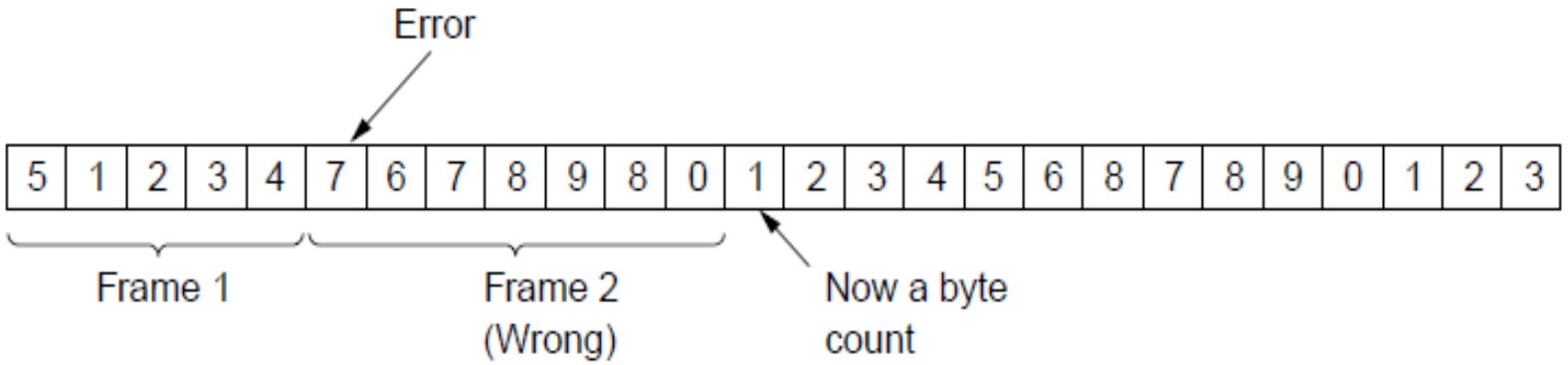
- First try:
  - Let's start each frame with a length field!



Are we done?

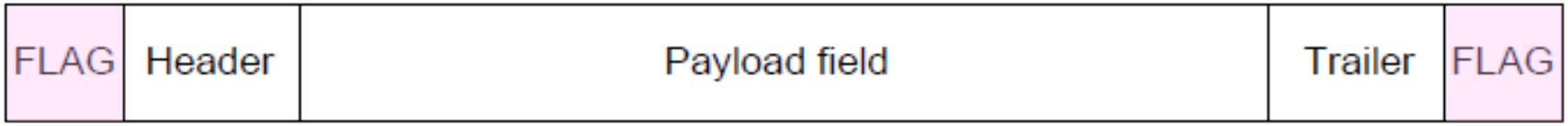
What if count is corrupted?

- We will frame the wrong bytes: framing error
  - Once framing on a link is desynchronized, it can stay that way
  - Need a method to resynchronize



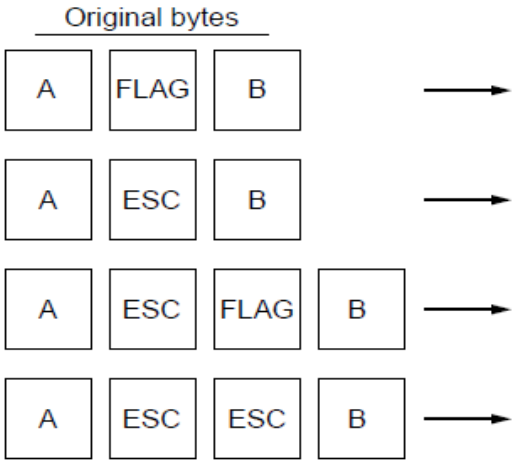
Byte stuffing

- Better idea:
  - Special FLAG byte that marks start/end of frame
  - When FLAG appears in the payload, "stuff" an escape character into the payload before the FLAG
  - Complication: must escape the escape code too!



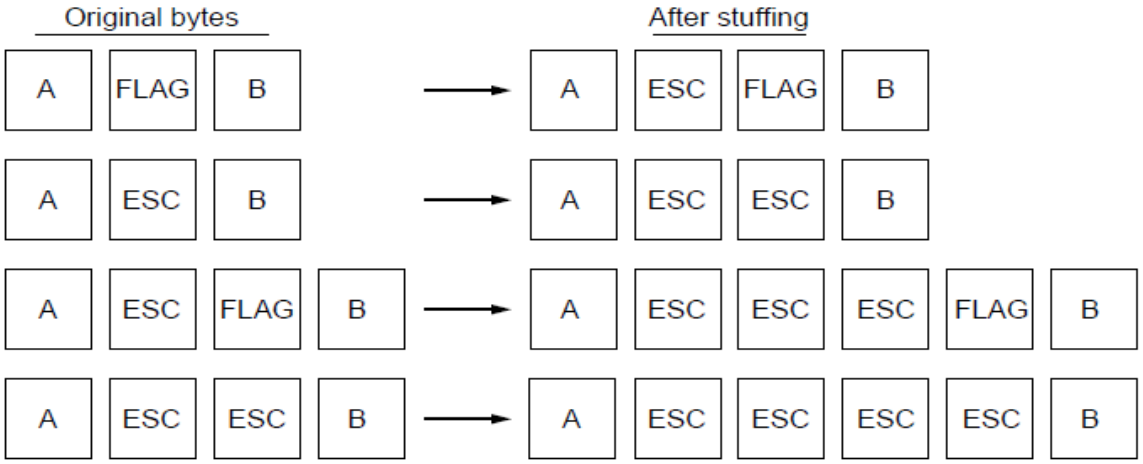
Byte stuffing examples

- Rules:
  - Replace each FLAG in data with ESC FLAG
  - Replace each ESC in data with ESC ESC



Byte stuffing examples

- Rules:
  - Replace each FLAG in data with ESC FLAG
  - Replace each ESC in data with ESC ESC



- Now any unescaped FLAG Is the start/end of a frame

Bit Stuffing

- Can stuff at the bit level too
  - Call a flag six consecutive 1s
  - On transmit, after five 1s in the data, insert a 0
  - On receive, a 0 after five 1s is deleted

Bit Stuffing Example

Data bits 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Transmitted bits with stuffing

Bit Stuffing Example

Data bits: 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Transmitted bits with stuffing: 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Stuffed bits

How does this compare with byte stuffing?

Framing and P4

- Real Ethernet frame:

- Ethernet in P4:

```
header ethernet_t {
        bit<48> dstAddr;
        bit<48> srcAddr;
        bit<16> etherType;
}
```

Agenda

- Link Layer ✓
  - Framing ✓
  - Error detection NEXT
  - Medium access control
  - Switching

Error detection and correction

- Error detection codes
  - Add check bits to the message bits to let some errors be detected
- Error correction codes
  - Add more check bits to let some errors be corrected
- Key issue is now to structure the code to detect many errors with few check bits and modest computation

Motivating Example

- A simple code to handle errors:
  - Send two copies! Error if different.

- How good is this code?
  - How many errors can it detect?

Hamming Distance

- Distance is the number of bit flips needed to change from one valid set of check bits to another

- Hamming distance of a code is the minimum distance between any pair of valid check bits

For a code of distance d+1,

up to d errors will always be detected

Simple Error Detection – Parity Bit

- Take D data bits, add 1 check bit that is the sum of the D bits
  - Sum is modulo 2 or XOR
- How well does parity work?
  - What is the hamming distance of parity bits?
  - How many errors will it detect?

Internet Checksum

- "*The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words …*" – RFC 791

- Fancy 16-bit parity

- Hamming distance = 2
- But: it handles bursts up to 16 errors

Cyclic Redundancy Check (CRC)

- Much more robust
- Based on polynomical division and finite field theory
  - Numbers represent polynomials
  - e.g, 10011010 is $x^7 + x^4 + x^3 + x^1$

Error Detection in Practice

- CRCs are widely used on links
  - Ethernet, 802.11, ADSL, Cable …
- Checksum used in Internet
  - IP, TCP, UDP … but it is weak
- Parity
  - Used infrequently