15-441 Computer Networks

Lecture 5

Link-Layer (1)

Dave Eckhardt

Some slides lifted from Peter & Hui

Roadmap

- **What's a link layer?**
- **Ethernet**
- **Things which aren't Ethernet**
  - Token Bus, Token Ring, FDDI, Frame Relay
  - 802.11
  - PPP, DSL, cable modems

Hui Zhang, Dave Eckhardt

What's a Link Layer?

- **What do we have?**
  - A physical layer
    - A transmitter
    - An evil transmission mutator (aka “channel”)
    - A receiver
  - Symbols, bits, things like that
- **What do we want?**
  - Packets
  - Addresses
  - Medium sharing
  - Reliability, fairness, world peace – maybe

What's a Link Layer?

- **Encoding**
- **Framing**
- **Addressing**
- **Error detection**
- **Reliability (assured delivery), flow control**
  - Details deferred until transport layer lecture
  - Saltzer, Reed, & Clark: End-to-End Arguments in System Design
    - Build features into lower layers only when ***provably necessary***
    - Link-layer reliability necessary ***only rarely***

What's a Link Layer?

- Encoding
- Framing
- Addressing
- Error detection
- ~~Reliability (assured delivery), flow control~~
- Medium-access control (MAC)

Hui Zhang, Dave Eckhardt

Link-layer Layers

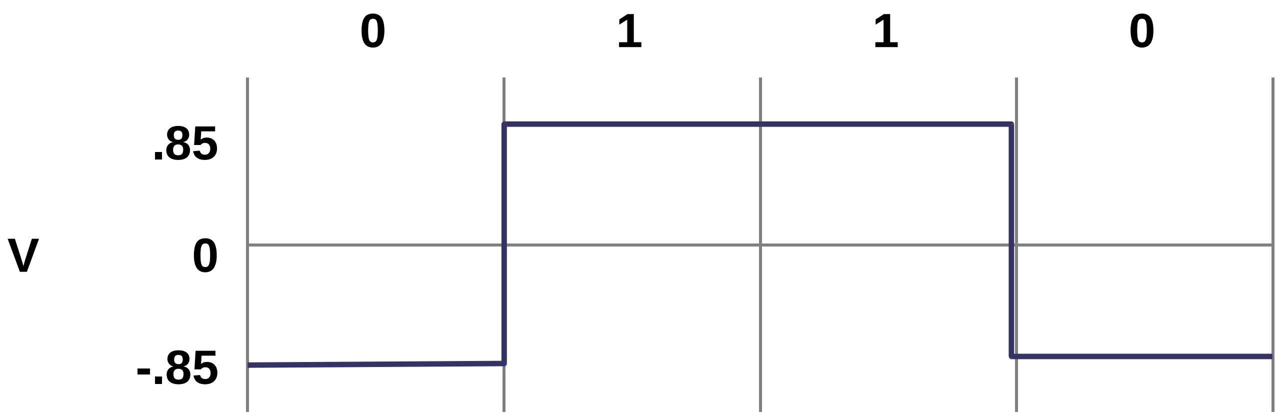
- **ISO OSI RM says "link layer" is a single thing**
- **IEEE 802 committee says it has two "sub-layers"**
  - LLC = Logical Link Control
    - Framing, addressing, error detection, ...
  - MAC = Medium Access Control
    - One bullet on previous slide
    - A whole "sub-layer" by itself
      - » Fruitful for academic papers, Ph.D. theses

Encoding

- **Physical layer transmits symbols**
  - Assume 0/1
  - Often low-voltage/high-voltage
- **Not all data streams are created equal**
  - Too many 1's or 0's in a row is bad
  - Clock recovery (determining symbol boundaries)
  - Automatic level discrimination (which voltage is 0 vs. 1)
- **Solution – transmit a "healthy mix" of 1's and 0's**

Hui Zhang, Dave Eckhardt

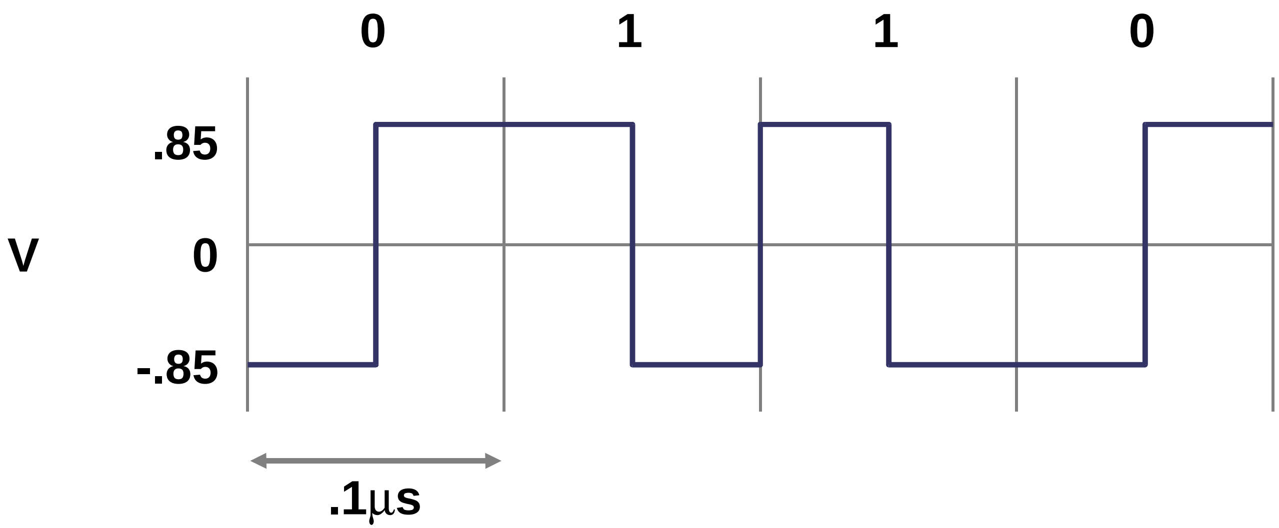
NRZ (Non-Return to Zero)



- High level for 1, low for 0 (the “obvious” thing)
- Long run of 1's or 0's confuses receiver
- Ok if runs cannot be long (RS-232: 11 bits/character)

Hui Zhang, Dave Eckhardt

Manchester Encoding



- **Positive transition for 0, negative for 1**
- **Transition every cycle for reliable clock**
  - cost: 2 transition times per bit
- **DC balance has good electrical properties**

Hui Zhang, Dave Eckhardt

4B/5B Encoding

- **NRZI line code**
- **Data coded as *symbols* of 4 data bits $\Rightarrow$ 5 line bits**
  - 100 Mbps uses 125 MHz.
  - Less bandwidth than Manchester encoding's 2X
- **Each valid symbol has at least two 1s**
  - Frequent transitions.
- **16 data symbols, 8 extra symbols used for control**
  - Data symbols: 4 data bits
  - Control symbols: idle, begin frame, etc.
- **Example: FDDI**

Hui Zhang, Dave Eckhardt

Framing

- **01011101000101010001111110001101010101010001101 01**
  - How many frames (packets) was that?
  - Where did each start and end?
  - Which bits belonged to no frame (idle link)?
- **Some techniques:**
  - Mutate bit stream (“special sequence”)
  - External information
    - Radio: “carrier sense” - lots of energy means “in a frame”
    - Out of band delimiters (e.g. 4B/5B control symbols)
    - Synchronous transmission (e.g., SONET)

Hui Zhang, Dave Eckhardt

Bit Stuffing

- **Mark frames with special bit sequence**
  - SDLC uses 1111111 (7 1's in a row)
- **Problem: that pattern will appear in data**
- **Sender rule**
  - If you've transmitted 6 1's in a row
    - Transmit 0 (“stuff a 0”), then next user bit
- **Receiver rule**
  - If you've received 6 1's in a row...
    - If you receive a 0, ignore it (it was stuffed), clear “6 1's” flag
    - If you receive a 1, declare end-of-frame
- **data 0001110011111100 sent as 000111001111110100**

Hui Zhang, Dave Eckhardt

Byte Stuffing

- **Same basic idea as bit stuffing**
- **Used when underlying layer is byte-oriented**
- **IBM RJE/NJE: DLE (“data-link escape”) byte**
  - DLE EOT means “end of packet”
  - DLE DLE means “user data contained a DLE”
- **Sender rule – to send user's DLE, send DLE DLE instead**
- **Receiver rule**
  - If you receive DLE and then:
    - A second DLE: store one DLE into user's buffer
    - An EOT: frame is done
- **Which one is more efficient?**

Link Types

- **Some links are "private"**
  - RS-232: two machines, one at each end
  - USB: one "upstream" node, one "downstream"
    - Computer and peripheral are *really* different
- **Some links are shared**
  - Radio links are *inherently* shared
    - There's only one ionosphere!
  - Some links share for price reasons
    - Primeval Ethernet: many stations, one long cable
    - Apple's "Localtalk" serial protocol, Corvus Omninet, etc.

Addressing

- **Give each "station" (node) a name**
  - Unique – on that link
- **Stations can ignore packets for other stations**
  - Use fast cheap stupid hardware, leave CPU for game
- **Data can be addressed to station *groups***
  - Multicast: "All Quake players"
  - Broadcast: "everybody"

Hui Zhang, Dave Eckhardt

Addressing Options

- **Dynamic – Appletalk**
  - Pick an address at random
  - Broadcast “Is anybody else here node 77?”
- **Static – Ethernet**
  - Every adaptor has factory-assigned number
  - 48 bits, two parts
    - 24 high-order bits sold by IEEE to manufacturer
    - 24 low-order bits assigned at factory
  - Special addresses
    - FF:FF:FF:FF:FF:FF = “everybody”

Error Detection – Why?

- **Physical layer *lies* to us**
  - Sometimes it tells us 0 when sender transmitted 1
- **Physical layers lie in different ways**
  - Some invert occasional bits
  - Some corrupt long bursts of bits
  - Some invert bits with equal probability
  - Some like to turn 1's into 0's (more than 0's into 1's)
- **Processing garbage frames can be expensive**
  - Processor load
- **Corrupted truths can be painful lies**
  - “I'm busy, wait 10 milliseconds” ⇒ 100000 milliseconds...

Error Detection – How?

- **Basic idea**
  - Send "checksum" along with each frame
  - Sender computes checksum from data, appends to frame
  - Receiver computes checksum on incoming data
    - Drop frame if computed != received
- **What's a "checksum"?**
  - May really be a sum (e.g., "parity")
  - Internet protocols use 16-bit 1's-complement sum
  - Ethernet uses "CRC-32" (polynomial division's remainder)
- **Link-level checksums designed to catch noise**
  - Not deliberate attacks – need *cryptographic* checksum

Parity

- **Parity = XOR "sum" of bits**
  - $0 \oplus 1 \oplus 1 = 0$
- **Parity provides *single error detection***
  - Sender provides ***code word*** and ***parity bit***
  - Correct: 011,0
  - Incorrect: 011,1
    - Something is wrong with this picture – ***but what?***
- ***Cannot* detect multiple-bit errors**

Outline Reminder

- Encoding
- Framing
- Addressing
- Error detection
- ~~Reliability (assured delivery), flow control~~

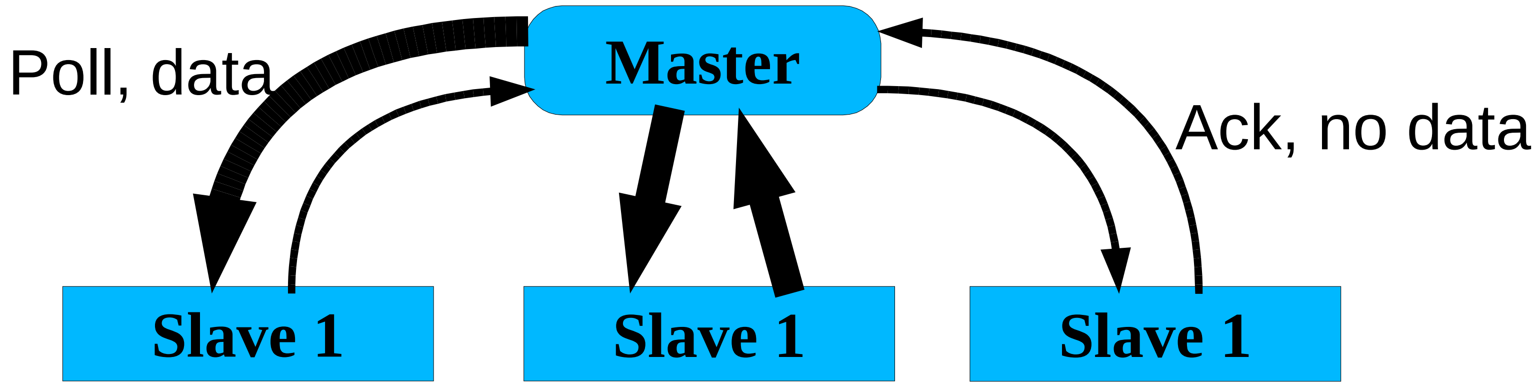
☞ Medium-access control (MAC)

Hui Zhang, Dave Eckhardt

Medium Access Control

- **Basic idea**
  - Who gets to transmit next?
- **Goals**
  - If only one station wants to transmit, it gets entire link
  - If multiple stations want to transmit, throughput is shared
    - Common case: shared equally
  - Try to avoid “master allocator node” (single point of failure)
- **Approaches**
  - Taking turns (polling, token-passing)
  - Random access (ALOHA)
  - Spread spectrum (FH, DS)

Polling



- **One “master”, many “slaves”**
  - Master: foreach(slave)
    - Send “poll” frame to slave's address
      - Include one data frame to slave, if available
    - Slave returns acknowledgement (“ack”) frame
      - Include one data frame for master, if available

Hui Zhang, Dave Eckhardt

Polling

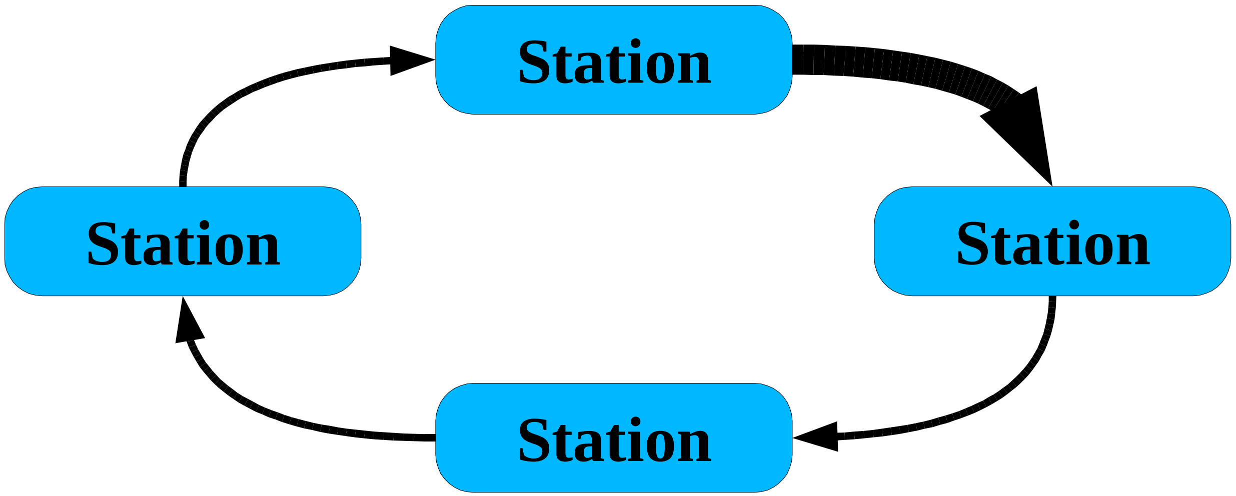
- **Problems**
  - Slaves can't talk to each other directly
    - Can “relay” through master, but inefficient
  - Polling idle slaves wastes time
  - If master dies, nobody can communicate
- **Well, that's dumb!**

Polling

- **Problems**
  - Slaves can't talk to each other directly
    - Can “relay” through master, but inefficient
  - Polling idle slaves wastes time
  - If master dies, nobody can communicate
- **Non-stupid example – IBM mainframe terminals**
    - Slaves don't want to talk to each other
    - Polling idle slaves is wasteful
      - – But humans type slowly; link is mostly idle anyway
    - Ok for master to die (it's the mainframe)
- **Dallas Semiconductor "1-wire" sensor net**

Token Passing

- **Basic idea**
  - Polling master forces slaves to take turns in order
  - Why not let the “slaves” take turns themselves?



Hui Zhang, Dave Eckhardt

Token Passing

- **Taking Turns**
  - A station transmits one or more frames
  - Then passes “transmit token” to next station
  - If no frames are queued, immediately pass token
- **Data Flow**
  - Frames flow around the ring to all stations
  - Receiver sets “I saw it” bit as frame flows by
  - Frame deleted when it flows back to sender

Token Passing

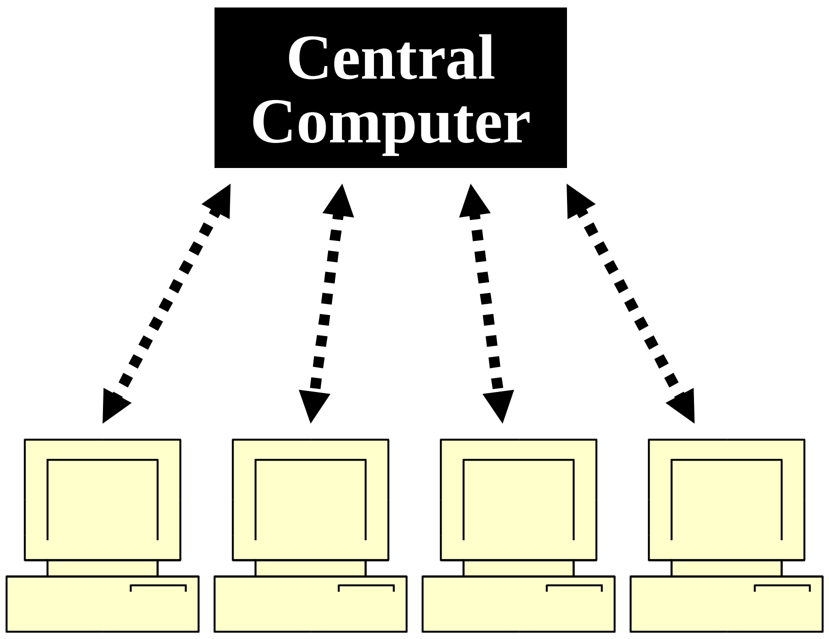
- **Performance**
  - Bound time each station may hold transmit authority
  - Yields fairness among busy stations
  - Provides simple bound on “waiting time to transmit frame”
- **Issues**
  - Any station failure breaks the system
  - Where does the “transmit token” come from?
    - When you power on a network, there isn't one...
    - "There can be only one..."
    - Distributed election protocol!

Medium Access Control

- **MAC-approaches reminder**
  - Taking turns (polling, token-passing)
  - ☞ Random access (ALOHA)
  - Spread spectrum (FH, DS)

ALOHA

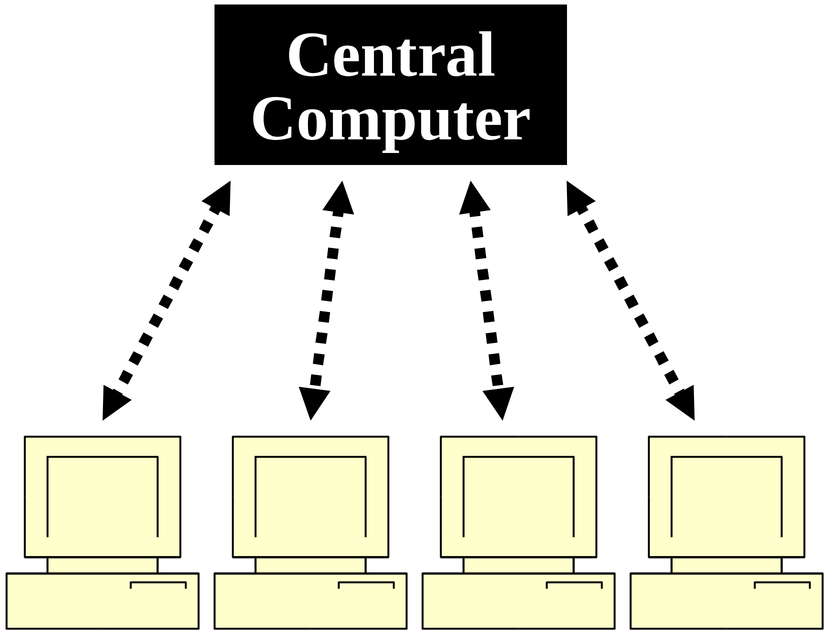
- **Hawaii, 1970...**
  - One university mainframe
  - Multiple campuses
    - On multiple ***islands***
    - Leased phone lines costly
    - Radio?
- **ALOHA system design**
  - Downlink channel
    - Scheduled by mainframe
  - Uplink channel - shared
    - Who transmits when?



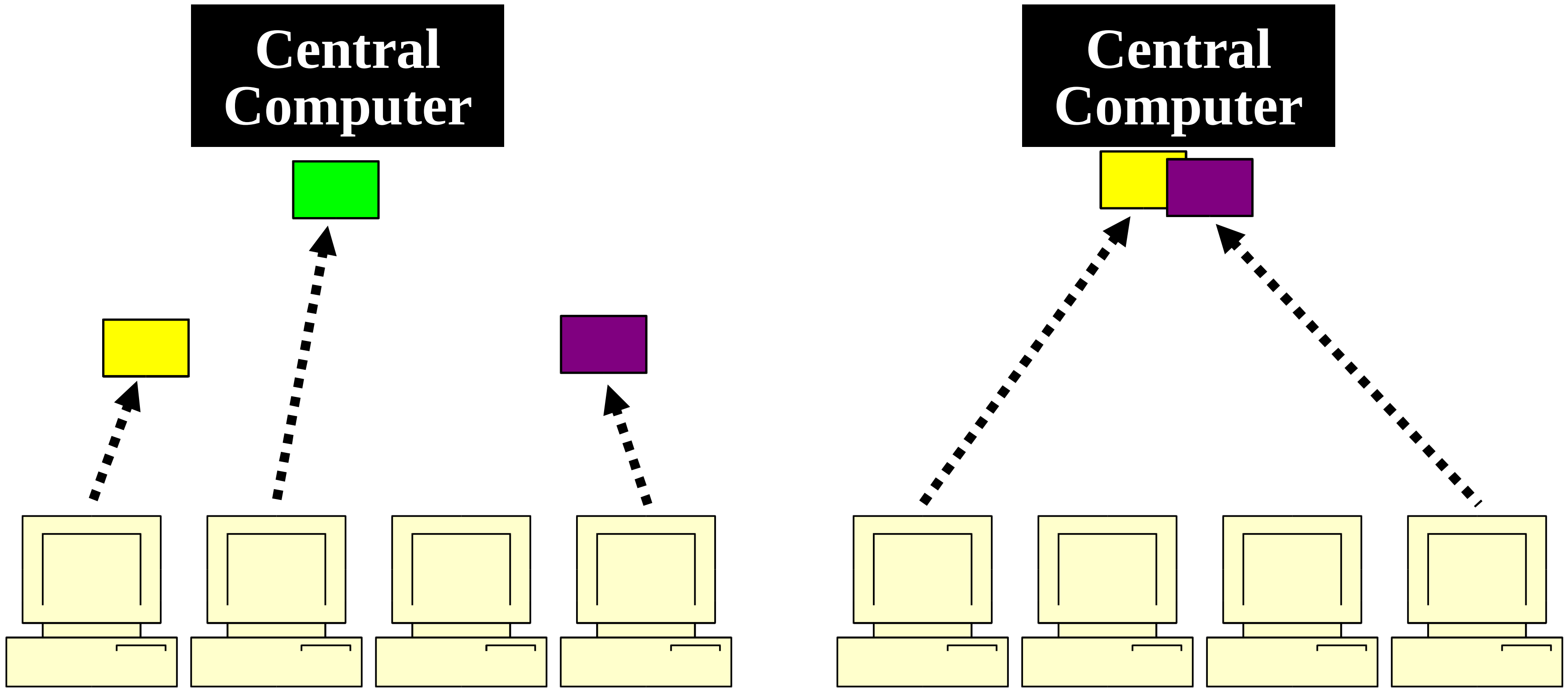
Hui Zhang, Dave Eckhardt

ALOHA

- **KISS – Keep It Simple, Stupid!**
  - Transmit when you have data
  - Expect ACK after "2τ"
  - Retransmit after random time
- **Why wouldn't a packet be ACK'd??**
  - Radio problem
    - Tune system to be "rare"
  - *Collisions!*
    - Uplink access is *random*



ALOHA



Hui Zhang, Dave Eckhardt

ALOHA

- **How bad are collisions?**
- **Famous analytical result: maximum link efficiency 18%**
  - Two assumptions...
    - Every node always wants to transmit
    - Infinitely many nodes
  - ...neither true for original system
- **What's so challenging about collisions?**
  - No synchronization among transmitters, so overlaps are random
  - ***<u>Any</u>*** overlap (even 1 bit) will probably destroy ***<u>two</u>*** packets
    - Now two stations go silent for a long time

Slotted ALOHA

- **Slots – a quick fix**
  - Choose a system-wide packet size
  - Downlink channel provides global clock
  - Uplink nodes transmit only on “slot” boundaries
  - Now station #3 collides with station #1 XOR station #2
    - Collision ⇒ retransmit in next slot with probability p
      - » (tunable system constant)
  - System throughput doubles to 37%
    - Recall: usually >> 37%
- **Properties of slotted ALOHA extensively studied**
  - Throughput, fairness, delay bounds

Medium Access Control

- **MAC-approaches reminder**
  - Taking turns (polling, token-passing)
  - Random access (ALOHA)
  - ☞ Spread spectrum (FH, DS)
    - a.k.a. "***really*** random access"

Hui Zhang, Dave Eckhardt

Spread Spectrum

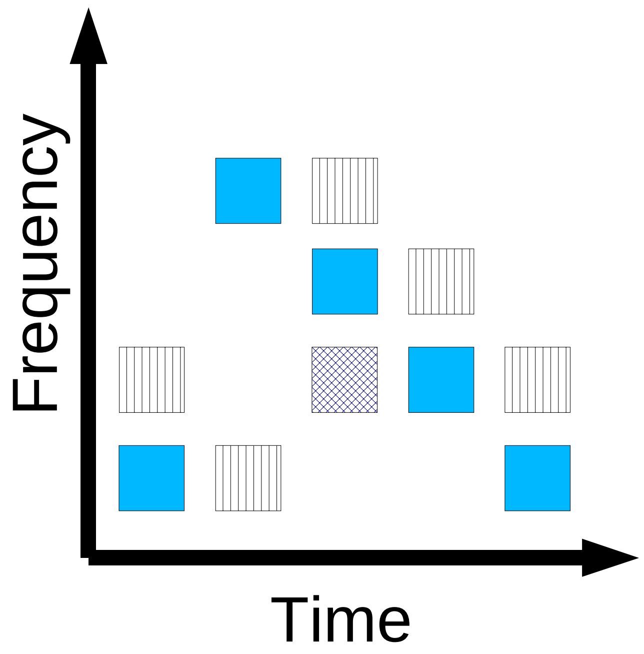
- **Basic idea**
  - Randomness worked out ok.
  - Maybe ***more*** randomness would be better?
    - – Everybody transmits *whenever they want*
- **Collisions? Who's afraid of collisions?**
  - Send every bit multiple times
  - Some copies of every bit will collide with somebody else
  - As long as a majority survive, it doesn't matter
    - – (Tricks allow you to get by with a minority, not a majority)

Spread Spectrum

- **Frequency-Hopping Spread Spectrum (FHSS)**
  - Invented by a Hollywood movie actress (more or less)
  - Transmit each copy of a bit on a different radio channel
- **Direct-Sequence Spread Spectrum (DSSS)**
  - Everybody shares one big fat radio channel
  - Use big fat digital signal processing to sort things out

Spread Spectrum – FHSS

- **Challenges**
  - Hopping schedule must avoid collisions
  - Frequency-agile radios expensive
  - Must "vote" on each bit value
- **Strengths**
  - "Learn" bad frequencies
  - Anti-jamming
  - Stealth



Spread Spectrum – Direct Sequence

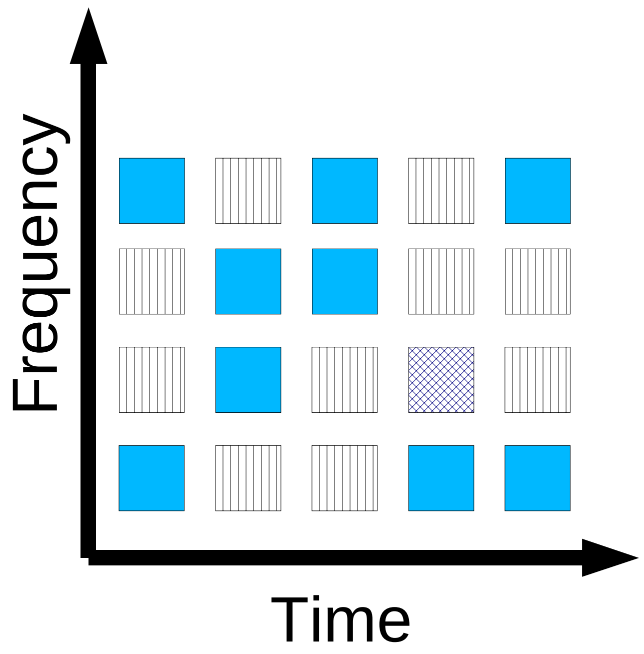
- **Basic idea**
  - "Multiply" bit stream by a "pseudo-noise" sequence
  - Data 0110 times PN 111000 =
    - 000111 111000 111000 000111
  - Increased bit rate widens bandwidth‡ of signal
  - Receiver gets N copies of each bit
    - Some have been flipped due to collision
    - Vote

‡ Yes, we mean bandwidth

Hui Zhang, Dave Eckhardt

Spread Spectrum – DSSS

- **Challenges**
  - Wide-band radio can be expensive
  - Longer PN requires faster DSP
  - Hard to explain
    - – This picture is imperfect
- **Strengths**
  - Moore's Law
  - Anti-jamming
  - Stealth



Hui Zhang, Dave Eckhardt

Spread Spectrum – Summary

- **Synchronization**
  - Sender, receiver must agree
    - – FH – hop schedule & starting time
    - – DS – PN sequence & starting time
- **Smooth overload**
  - As channel load goes over 100%
    - – ...voting errors increase
    - – ...probably some people disconnect voluntarily
    - – ...no sudden "out of slots, no more users" wall
      - » As with TDM, FDM, etc.

Spread Spectrum vs. ISO OSI RM

- **Is SS a physical-layer technology or a MAC technology?**
  - Getting SS to work right involves ***adaptive power control***
    - Transmit gain is clearly a physical-layer function
  - Uh-oh, it nicely blurs the boundary

Link Layer – Summary

- **Link-layer design**
  - Framing, Addressing, Error detection(/correction), MAC
  - Each has several options
  - Good design
    - – Identify a popular "market", pick a solution per problem
    - – Same basic concepts used over and over
- **Medium Access Control**
  - Great generator of Ph.D. theses
  - Fun with distributed algorithm design!
  - Small changes vastly increase system performance
  - Chaos: taking turns ⇒ random access ⇒ spread spectrum

Hui Zhang, Dave Eckhardt