ECE 435 – Network Engineering
Lecture 20

Vince Weaver

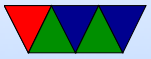
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

15 November 2018

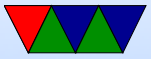
Announcements

- HW#8 was due
- HW#9 will be posted
- Will update you on project topics



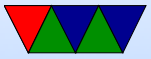
Ethernet History

- Proposed by Bob Metcalfe in 1973
  (went on to found 3Com)
- Metcalfe, Boggs, Thacker, and Lampson listed on patent
- Inspired by ALOHAnet, a wireless network in Hawaii, allow users on various islands to connect to server on Oahu
- Various competing local networks, Ethernet won in the end

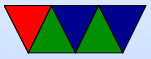


Token Ring (Ethernet Competitor)

- Guaranteed Deterministic Delivery (vs Ethernet: best effort)
- Dates to 1970s
- Standardized by IBM, 1984, IEEE 802.5 (note, not RFC)
- 4Mbps, eventually shielded twisted pair, eventually 16Mbps, 100Mbps and 1Gbps
- 3-byte frame passed around gives permission to transmit
- More complex, no crossover cable (direct connect two machines),

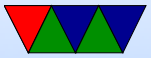


- Supports multiple identical MAC addresses
- Deterministic time to get to transmit
- Frames can have different access priorities
- Empty token passed around. If data to transmit, put in. Then passes around until it gets to receiver, removed, and back to passing empty token. When gets back to originator it knows it has been received.
- Token Bus, GM, IEEE802.4 (withdrawn) like ring, but virtual ring. Needed to know neighbors to pass token. Guaranteed worst case transmit time.



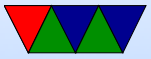
Why did Ethernet win?

- Other competitors: FDDI, ATM, DQDB.
- Why did it win? Simpler and thus cheaper.
- Why simpler?
  No priority mechanism, no QoS, no central control
- Could use cheaper twisted pair cable
- Token ring cards generally a lot more expensive than Ethernet



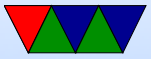
The Ethernet Progression

- Low speed (3Mbps) $\rightarrow$ High speed (40Mbps)
- Shared media $\rightarrow$ dedicated media
- LAN $\rightarrow$ WAN



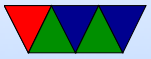
Ethernet History

- 1972 – experimental 3Mbps
- 1981 – DIX (DEC/Intel/Xerox) ver 1 (10Mbps)
- Standardized in 1981
- 1982 – DIX ver 2



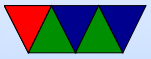
Thick Ethernet

- 1983 – IEEE 802.3/10BASE5
- "Thick Ethernet", up to 500m often yellow or orange (standard suggests yellow)
- Looks like garden hose.
- Vampire Tap, AUI connector, drill into cable, at 2.5m intervals (to avoid reflections)
- Terminated on each end One bad connection could ruin for all



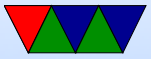
Thin Ethernet

- 1985 – 10BASE2
- "thin net", thinner connections, BNC connectors, T connectors (185m, rounded up to 200m)
- 50 ohm terminator, grounding loops, one and only one must be tied to ground.
- How to detect network problem? Send pulse, look for echo



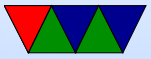
Ethernet Naming

- Naming: Speed/BROAD,BASE,PASS/PHY
- Almost all is baseband (narrow frequency, vs broadband).
- PHY originally was distance could travel (in 100m) but now medium type.



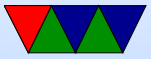
10BASE-T

- 1990
- 10BASE-T – twisted pair (Cat3) , needed hub
- 1993 – 10BASE-F – fiber

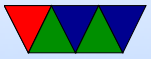


Faster Ethernet

- We will talk about this in more detail later
- 1995 – 100BASE-T, 100BASE-TX 4B5B MLT-3 cat5 two twisted pairs
- 1997 – Full-duplex
- 1998/1999 – 1000BASE-TX PAM-5, four twisted pairs, can transfer in both directions on one pair using DSP/echo cancellation
- 2006 – 10GBASE-T
- 2010 – 40G and 100G

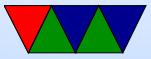


- 2017 – 400GB



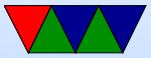
Ethernet MAC

- CSMA/CD "Carrier sense multiple access with collision detection"
- First senses cable (how?)
- If busy, waits
- Sends. If collision, jams the cable aborts transmission, waits random back off time before retrying.
- Exponential backoff. Randomly choose time from 0 to



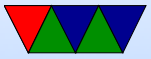
$2^k - 1$ where k is number of tries (capping at 10). Time slot is 512 bits for 10/100, 4096 for 1Gbs

- on newer full-duplex links no need for carrier sense and collision detection not needed



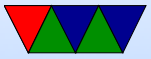
Ethernet Collisions

- In order to work properly, twice round-trip time needs to be less than time needed to transmit minimal (64-byte) frame, otherwise not possible to notice collision in time and frame loss
- This limits network size to collision domain
- Bits wasted is not bad, collision often caught in the preamble

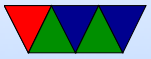


Manchester Encoding

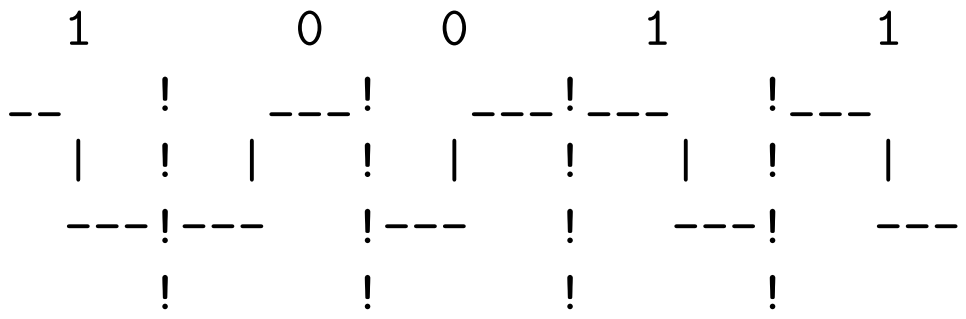
- Does not use 0V for 0 and 5V for 1.
  Why? Idle is 0, so how can you tell how many zeros at beginning of signal?
- Could use +1V/-1V, but still would need way to sync signal on long runs of 0 or 1
- Manchester encoding
  - 1 is high to low transition.
  - 0 is low to high transition.
  - Always a transition in the middle of an interval.



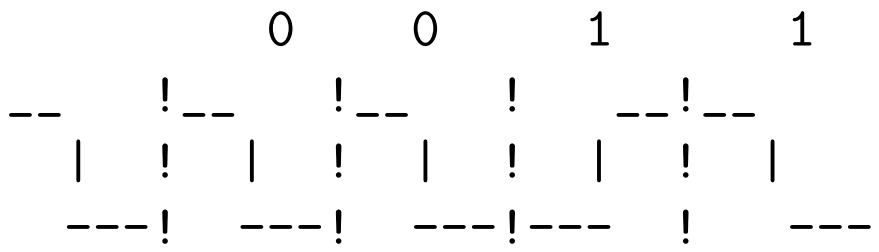
- Disadvantage, need twice as much bandwidth
- Differential Manchester
  - transition at start of interval means 0
  - lack of transition means 1
  - Still transition in the middle
  - More complex but better noise handling
- Ethernet uses Manchester, Token Ring uses differential Manchester
- Ethernet high 0.85V and low -0.85V

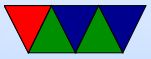


Manchester

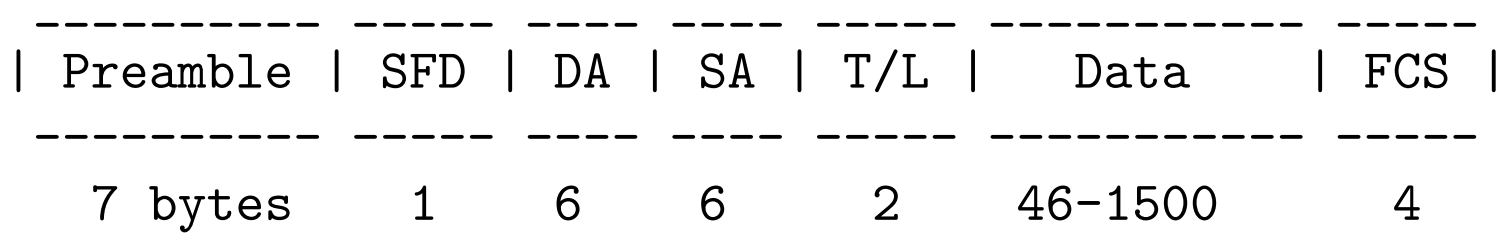


Differential Manchester

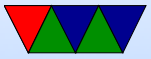




Ethernet frame layout

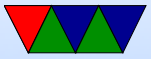


- Preamble is fixed 1010...1010 in transmission order (LSB least significant bit first)
  On original Ethernet this was 10MHz 6.4us pulse used to synch clocks The PHY might do other things (100BASE-X uses 4B/5B stuff, so different pattern)
- SFD - indicates the start of the frame with value



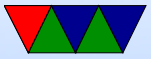
10101011 in transmission order
(original Ethernet declared 8 bytes of same pattern, but on modern first 7 bytes might be different)

- DA = 48 bit destination MAC address (see later)
- SA = 48 bit source MAC address
- T/L: Originally type field. 802.3 makes it length of *data* field (not length of frame). Later in 1997 802.3 approved as type too, so dual meaning. How tell difference? Since cannot be longer than 1500, any value bigger than 0x600 (1536) is type. 0x0800 = IPv4, 0x86dd = IPv6, 0x0806 = ARP



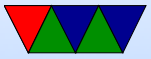
How tell length if type? Detect end, or checksum (this is most common usage)
How tell type if length? Will have 802.2 header

- Data – data from 46 to 1500 bytes
  Why limit 1500B? because RAM was expensive in 1978. If smaller than 46 bytes padded. Makes sure checksum works. Also if too short, could be done transmitting before a collision can be detected (light travel to furthest node and back)
- FCS – a 32-bit CRC code. Somewhat complicated, magic number at end 0xc704dd7b If incorrect FCS,



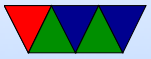
silently drops. How can we do this? Up to upper protocol (say TCP/IP) to figure out if need to resend. Makes things simple. No need to wait for ACKs.

- Frame size is variable. Often first two fields are excluded and said that Ethernet packets are between 64 and 1518 bytes long



MAC Address

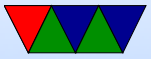
- 6-byte address
  - First 3 bytes the OUI (organization unique identifier)
  - Next 3 bytes supposed to be a unique ID
- Ethernet packets put on the wire least-significant bit first (as if shifted right out of a shift register)
- Multicast if the "first" bit (meaning 0x1, not 0x80) is set in the first octet
- Broadcast if all bits set ff:ff:ff:ff:ff:ff



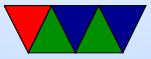
ARP – address resolution protocol

- On local network, how do we find MAC address if we know IP?
  Hard-code in /etc/ethers?
  Request somehow?

- ARP (RFC826)
  - Device first checks ARP cache to see if already knows
  - Otherwise, broadcasts to ff:ff:ff:ff:ff:ff "who has this IP"

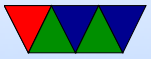


  - Device reply with its IP and MAC (unicast)
  - These are cached
  - Timeout in case you reassign
  - ARP announcement: can broadcast when your address changes so they can update (gratuitous ARP)
  - Other optimizations(?)
- Used for many higher protocols, but not IPv6 which uses NDP (Neighbor Discovery Protocol)
- Security: ARP spoofing



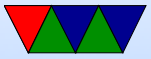
RARP/BOOTP

- Some cases need to do RARP (Reverse ARP) (RFC 903) have own MAC, find IP (netbooting is common reason)
- ARP packets nor forwarded, so extension called BOOTP that allowed network booting.
- BOOTP automated by DHCP.



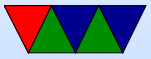
Naming note

- Why IEEE standards start with 802
  Next available? Also co-incidentally first meeting was Feb. 1980
  For example, IEEE floating point is IEEE 754 but first meeting was not April 1975



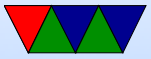
Ethernet Transmission

- Break data into frame
- In half-duplex CSMDA/CD senses carrier. Waits until channel clear
- Wait for an inter-frame-gap (IFG) 96 bit times. Allows time for receiver to finish processing
- Start transmitting frame
- In half-duplex, transmitter should check for collision.
  Co-ax, higher voltage than normal
  For twisted pair, noticing signal on the receive while



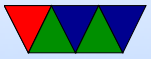
transmitting

- If no collision, then done
- If collision detected, a *jam* signal is sent for 32-bits to ensure everyone knows. Pattern is unspecified (can continue w data, or send alternating 1s and 0s)
- Abort the transmission
- Try 16 times. If can't, give up
- Exponential backoff. Randomly choose time from 0 to $2^k - 1$ where k is number of tries (capping at 10). Time slot is 512 bits for 10/100, 4096 for 1Gbs
- Wait the backoff time then retry

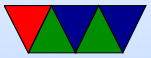


Ethernet Receiving

- Physical layer receives it, recording bits until signal done. Truncated to nearest byte.
- If too short (less than 512 bits) treated as collision
- If destination is not the receiver, drop it
- If frame too long, dropped and error recorded
- If incorrect FCS, dropped and error recorded
- If frame not an integer number of octets dropped and error recorded
- If everything OK, de-capsulated and passed up

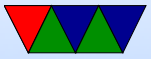


- Frame passed up (minus preamble, SFD, and often crc)
- Promiscuous mode?



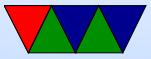
Maximum Frame Rate

- 7+1 byte preamble 64-byte frame, IFG of 12 bytes between transmissions. equals 672 bits. In 100Mbps system 148,800 frames/second



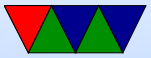
Full Duplex MAC

- Early Ethernet was coaxial in a bus
- Twisted pair has replaced this, usually in a hub/or switch star topology
- 10BASE-T and 100BASE-TX pair for transmit or receive
- inefficient. Since point to point, why do you need arbitration?
- Full-duplex introduced in 1997. Must be able to



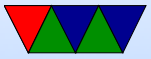
transmit/receive w/o interference, and be point to point.

- Full duplex effectively doubles how much bandwidth between. Also it lifts the distance limit imposed by collision detection



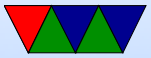
Ethernet Flow Control

- Flow control is optional
- In half duplex a receiver can transmit a "false carrier" of 1010..10 until it can take more.
- Congested receiver can also force a collision, causing a backoff and resend. Sometimes called force collision
- Above schemes called "back pressure"
- For full duplex can send a PAUSE frame that specifies how much time to wait.

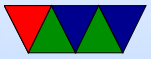


Fast Ethernet (100MB)

- 10MB not fast enough! What can we do?
  - FDDI and Fibrechannel (fast optic-ring), too expensive
  - Can we just multiply all speeds by 10? Or else come up with some completely new better thing?
  - IEEE decided to just keep everything same, just faster
  - The other group went off and made 802.12 100BaseVG (which failed)
- 802.3u 1995
- 100BASE-TX most common



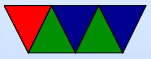
- Bit time from 100nsec to 10nsec
- Uses twisted pair/switches, no coax
  - To use cat3 100BASE-T4 wiring needed 4 twisted pair and complex encoding, no Manchester, ternary
  - To use cat5 wiring 100BASE-TX. Two twisted pair, one to hub, one from.
- Often split between MAC (media access controller) and PHY (physical interface). Card configures the PHY via the MII (media independent interface)
  Goal was you could have same card but interchangeable PHY (twisted pair, fiber, etc). 4bit bus



Interface requires 18 signals, only two can be shared if multiple PHY

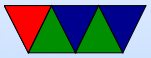
So RMII (reduced) was designed. Clock doubled, only 2-bit bus. Fewer signal wires.

- 100BASE-TX:
  - 2 pairs. One pair 100MB each direction, full duplex, 100m distance
  - Raw bits (4 bits wide at 25MHz at MII) go through 4B/5B encoding clocked at 125MHz. (DC equalization and spectrum shaping)
  - Then NRZI encoding (transition at clock if 1, none if



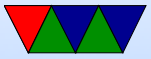
0).

- TX then goes through MLT-3 encoding (-1,0,+1,0. Transition means 1, no transition means 0) 31.25MHz, much like FDDI



Router vs Hub vs switch

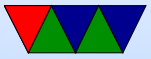
- Hub all frames are broadcast to all others
  Bandwidth is shared (only say 100MB for all)
- Switch – direct connection, no broadcast. Has to be intelligent. Each point to point connection full bandwidth.
  no collisions. Internally either own network to handle collisions, or else just buffer RAM that can hold onto frames until the coast is clear.
- Multi-speed hubs?



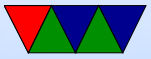
When 10/100MB first came out, cheap hubs could only run at 10MB or 100MB. But switches *really* expensive. They had a compromise 10/100MB hub that internally had a hub for both then a mini-switch to bridge the gap.

- Direct Ethernet connection. Need a special loopback cable?

  Modern cards can detect direct connect and swap the wires for you
- Router will move frames from one network to another
- Lights. How many ports? Uplink ports?
- Power over Ethernet

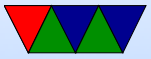


- Method B: In 10/100 Base T, only of the 4 pairs in Cat5 used. So send voltage down spare pairs
- Method A: send DC voltage down with the signals floating on top
- Original 44 VDC, 15.4W
- POE+ 25W
- Need special switch to send power, and device on other end has to support it.



Gigabit Ethernet

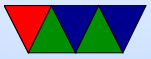
- Two task forces working in 1998/1999
- 802.3z 1998 (fiber), 802.3ab 1999 (copper)
- Could still use hub, problem was the CSMA/CD restriction.
  - About 200m for 100Mbps.
  - For Gb would have been 20m which is not very far.
  - Carrier extension: hardware transparently pads frames to 512 bytes
    Wasteful, 512 bytes to send 64 bytes of data



- Frame bursting: allow sender to sends sequence of multiple frames grouped together
- Better solution is just use full duplex
- 1000Base-SX (fiber)/LX (fiber)/CX (shielded)/T (cat 5), more
- Fiber
  - No Manchester, 8B/10B encoding. chosen so no more than four identical bits in row, no more than six 0s or six 1s

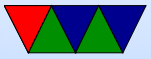
    need transitions to keep in sync

    try to balance 0s and 1s? keep DC component low so



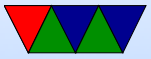
can pass through transformers?

- 1000BASE-T
  - 5 voltage levels, 00, 01, 10, 11, or control. So 8 bits per clock cycle per pair, 4 pairs running at 125MHz, so 1GBps
  - simultaneous transmission in both directions with adaptive equalization (using DSPs), 5-level pulse-level modulation (PAM-5) [technically 100BASE-TX is PAM-3]. Diagram? looks sort of like a sine wave as cycle through the voltages.
  - four-dimensional trellis coded modulation (TCM) 6dB



coding gain across the four pairs

  - Autonegotiation of speed. Only uses two pairs for this, can be trouble if pairs missing.
- Fast enough that computers at time had trouble saturating such a connection
- Jumbo Frames? 9000 byte?

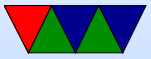


Even Faster Ethernet

http://www.theregister.co.uk/2017/02/06/decoding_25gb_ethernet_and_beyond/

- Misquote: Not sure what the network will be like in 30 years, but they will call it Ethernet.

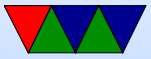
- 2.5Gb: 802.3bz (Sep 2016?)
  Like 10Gb but slower. Can't run 10Gb over Cat5e
  Power over Ethernet (for using on wireless access points)
  Power with signal overlaid on top.
  2.5Gb on Cat 5e, 5Gb on Cat6



- 10Gb: 802.3ae-2002. Full duplex, switches only
  Need Cat6a or Cat7 for links up to 100m
  Expensive. Lots of kind. 10GBASE-T, 802.3an-2006
  100m over cat6a, 55m Cat6
  additional encoding overhead, higher latency
  Tomlinson-Harashin precoding (THP), PAM15 in two-dimensional checkerboard DSQ128 at 800Msymbol/s

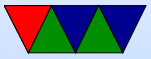
- 25Gb, 802.3by. 25GBASE-T, 50GBASE-T. Available, if copper only a few meters

- 40GB, 100GB. 802.3ba-2010, 802.3bg-2011, 802.3bj-



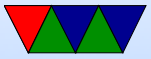
2014, 802.3bm-2015
40GBASE-T twisted pair 40GBit/s 30m. QFSP+ connectors, like infiniband

- Terabit? still under discussion



Autonegotiation

- How figure out line speed and duplex



What does your machine have

- skylake machine:

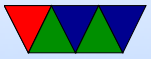
```
[   18.240021] e1000e: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: Rx/Tx
```

- Raspberry Pi:

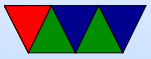
```
[   77.110505] smsc95xx 1-1.1:1.0 eth0: link up, 100Mbps, full-duplex, lpa 0xC5E1
```

- Haswell machine:

```
[    3.907651] tg3 0000:03:00.0 eth0: Tigon3 [partno(BCM95761) rev 5761100]
(PCI Express) MAC address f0:92:1c:f5:e8:f3
[    3.919115] tg3 0000:03:00.0 eth0: attached PHY is 5761
(10/100/1000Base-T Ethernet) (WireSpeed[1], EEE[0])
```

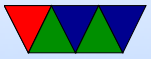


```
[    3.929838] tg3 0000:03:00.0 eth0: RXcsums[1] LinkChgREG[0] MIirq[0] ASF[1] TSOcap[1]
[    3.938174] tg3 0000:03:00.0 eth0: dma_rwctrl[76180000] dma_mask[64-bit]
[   13.758613] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[   15.404905] tg3 0000:03:00.0 eth0: Link is up at 100 Mbps, full duplex
[   15.411479] tg3 0000:03:00.0 eth0: Flow control is on for TX and on for RX
```



Linux OS Support

- When frame comes in, interrupt comes in
- Allocates `sk_buff` copies in
- Old: `net_if_rx()` interrupt, `net_rx_action()` interrupt/polling
- `net_if_receive_skb()`
- passes it to proper net level (`ip_recv()`, `ip_ipv6_recv()`, `arp_recv()`



- for send
- `net_tx_action()`
  `dev_queue_xmit()` and then deallocate `sk_buff`
- `qdisc_run()` selects next frame to transmit and calls `dequeue_skb()`