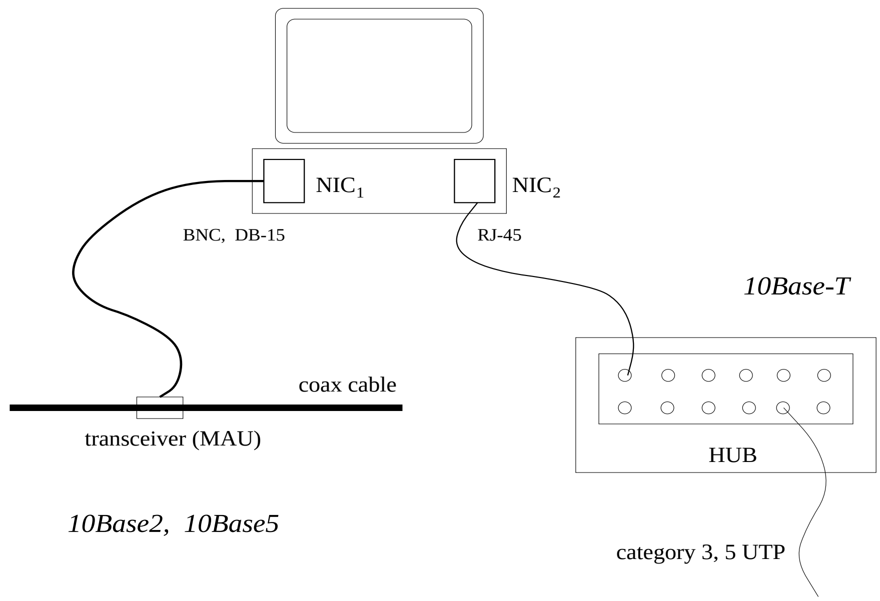
Ethernet

⟶ copper, fiber

Types (some just historical):

- 10Base2 (ThinNet): coax, segment length 200 m, 30 nodes/segment
- 10Base5 (ThickNet): coax, segment length 500 m, 100 nodes/segment
- 10Base-T: twisted pair, segment length 100 m, 1024 nodes/segment
- 100Base-T (Fast Ethernet): category 5 UTP, fiber (also 100VG-AnyLAN)
- Gigabit & 10 Gbps Ethernet: fiber, category 5 UTP
- 100 Gbps Ethernet

Connectivity example (stone age):



- single-homed vs. multi-homed
- unique 48-bit Ethernet address per NIC
- physical network: bus vs. hub vs. switch

  → very old vs. old vs. not-so-old

  → today: switched Ethernet

High-speed Ethernets have shorter network diameter

- 2500 m for 10 Mbps Ethernet
- 200 m for 100 Mbps Ethernet
- even shorter for 1 Gbps Ethernet

  → unless fully switched (later discussion)

⟶ distance limitations: due to Ethernet protocol

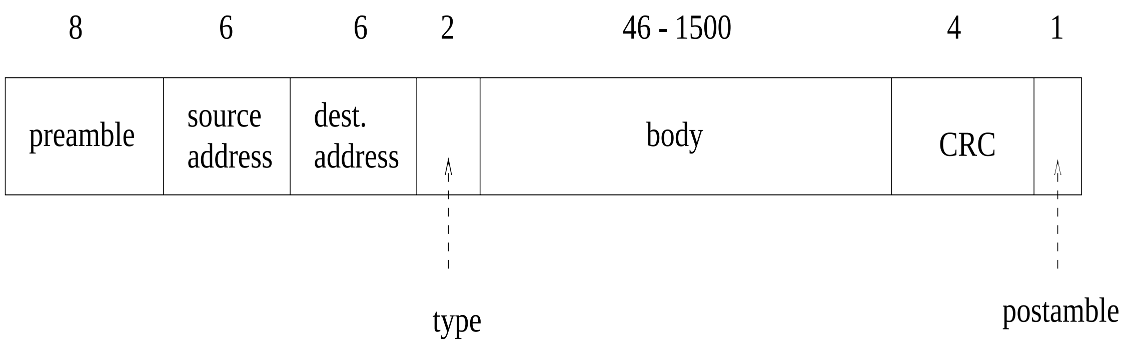
⟶ creates complications for long-haul

⟶ e.g., tens, hundreds, or thousands of miles

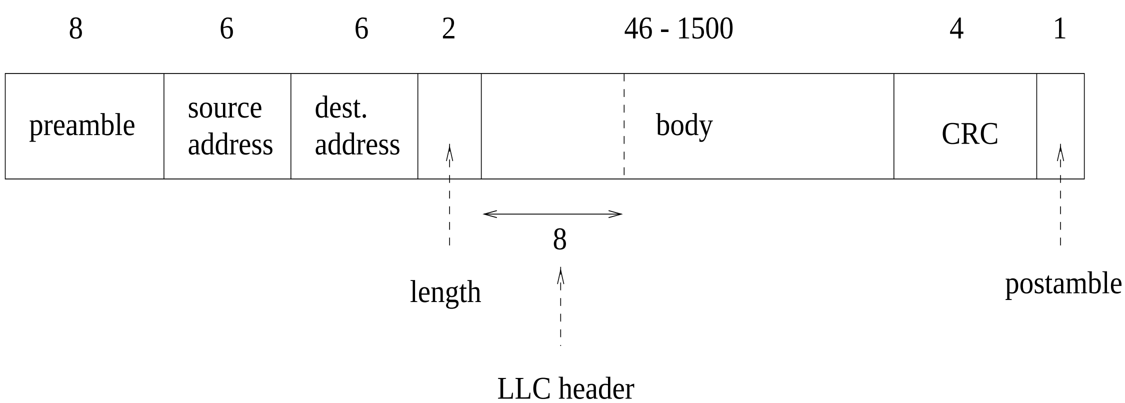
⟶ 1, 10, 100 Gbps: tier-1 backbone speeds

⟶ also multiples of 1 and 10 Gbps

DIX Ethernet frame:



IEEE 802.3 Ethernet frame:



$\longrightarrow$ IEEE 802.2 LLC (Logical Link Control)

$\longrightarrow$ two Ethernet types co-exist (802.3 dominant)

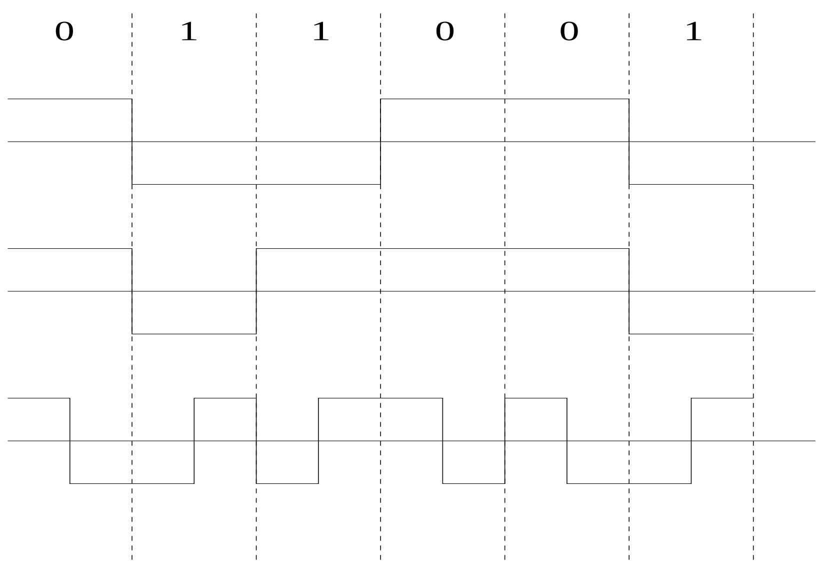
Digital transmission of digital data:

→ Ethernet uses baseband transmission

Using square waves to represent bits

→ methods and issues

- NRZ-L (non-return to zero, level)
- NRZI (NRZ invert on ones)
- Manchester (biphase or self-clocking codes)



Trade-offs:

- NRZ codes—long sequences of 0's (or 1's) causes synchronization problem; need extra control line (clock) or sensitive signalling equipment.
- Manchester codes—synchronization achieved through self-clocking; however, achieves only 50% efficiency vis-à-vis NRZ codes.

4B/5B code

Encode 4 bits of data using 5 bit code where the code word has at most one leading 0 and two trailing 0's.

0000 $\leftrightarrow$ 11110, 0001 $\leftrightarrow$ 01001, etc.

$\longrightarrow$ at most three consecutive 0's

$\longrightarrow$ efficiency: 80%

Encoding: Manchester

$\longrightarrow$ note: Ethernet is baseband

Addressing:

- 48 bit unique address
- point-to-point
- broadcast (all 1's)

Receiver: Ethernet NIC accepts frames with "relevant" address.

- accepts only own frame address

  $\rightarrow$ default

- accepts all frames: called promiscuous mode

  $\rightarrow$ can set with root privilege

  $\rightarrow$ useful for traffic monitoring/sniffing

Ethernet MAC protocol: CSMA/CD

- MA (Multiple Access): multiple nodes are allowed simultaneous access

  → just send

- CS (Carrier Sense): can detect if some other node is using the link

  → rule: if busy, wait until channel is not busy

  → works well in small areas: why?

- CD (Collision Detection): can detect if collision due to concurrent transmission has occured

  → rule: if collision, retry later

  → key question: when is later?

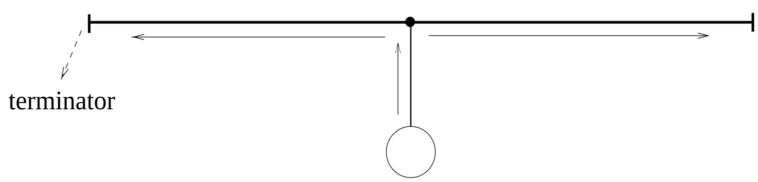
  → collision detection: more difficult in wireless environments

Collision detection mechanism:

Bi-directional signal propagation

→ terminator absorbs signal: prevent bounce back

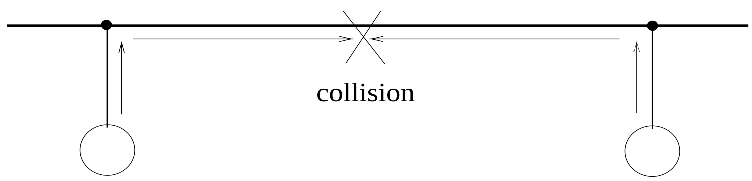
→ can hear different signal from one transmitted



Collision: 2 stations

→ while transmitting data frame, hears collided signal

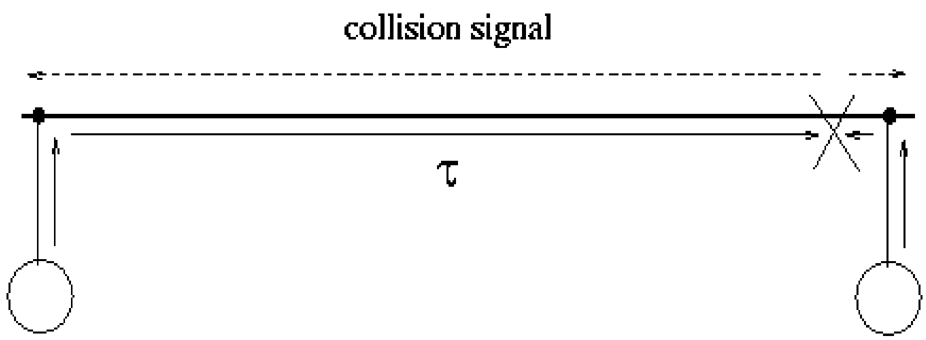
→ data frame cannot be too small



→ meet in the middle: best-case

→ why?

Worst-case collision scenario:



$\longrightarrow$ τ: one-way propagation delay

- sender needs to wait 2τ sec before detecting collision

  $\rightarrow$ time for echo to bounce back

- for 2500 m length, 51.2 μs round-trip time (2τ)
- enforce 51.2 μs slot time
- at 10 Mbps, 512 bits: minimum frame size

  $\rightarrow$ assures collision detection

  $\rightarrow$ wireless collision detection: why more difficult?

Transmit at least 512 bits

$\longrightarrow$ $6 + 6 + 2 + 46 + 4 = 64$ B $= 512$ bits

When to retry upon collision: use exponential backoff

1. Wait for random $0 \leq X \leq 51.2$ μs before 1st retry
2. Two consecutive collisions: wait for random $0 \leq X \leq 102.4$ μs before 2nd retry
3. Three consecutive collisions: wait for random $0 \leq X \leq 204.8$ μs before 3rd retry
2. i consecutive collisions: wait for $0 \leq X \leq 2^{i-1}\, 51.2$ μs before next attempt
3. Give up if $i > 16$

$\rightarrow$ a form of stop-and-wait

$\rightarrow$ what's the ACK?

$\rightarrow$ guaranteed reliability?

$\rightarrow$ why exponential backoff?

CSMA/CD Throughput

$\longrightarrow$ approximate analysis in simplified setting

Assumptions:

- time is slotted

  $\rightarrow$ slot duration: 2τ

- k hosts; each host transmits with probability p at every slot

  $\rightarrow$ transmission behavior among hosts independent

  $\rightarrow$ transmission behavior across slots independent

New performance metric: utilization (ϱ)

$\longrightarrow$ fraction of total bandwidth utilized

$\longrightarrow$ $0 \leq \varrho \leq 1$

$\longrightarrow$ small ϱ: large wastage

In slotted CSMA/CD:

$\longrightarrow$ fraction of usefully used slots

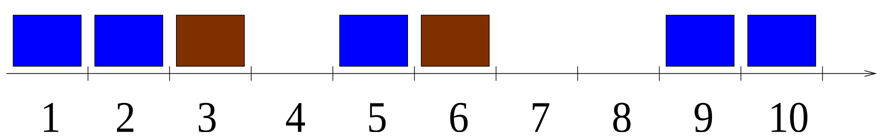
$\longrightarrow$ what are "uselessly used" slots?

Ex.: snapshot of baseband channel over 10 time slots

→ blue: successfully transmitted frames

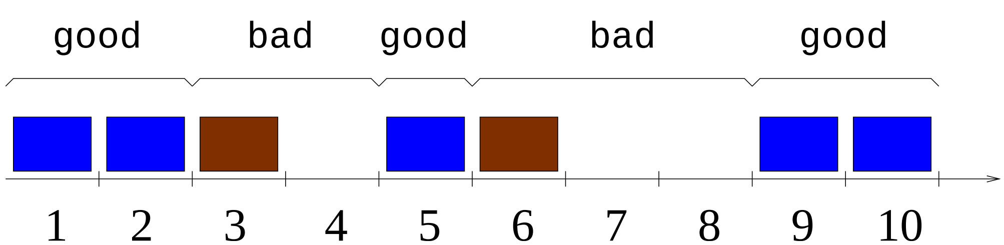
→ brown: collided frames

→ utilization ϱ?



One more viewpoint:

⟶ note: useful and useless "periods" alternate



In the long run,

$$\varrho = \frac{E[\text{good}]}{E[\text{good}] + E[\text{bad}]}$$

$\rightarrow$ avrg. length of adjacent "good" and "bad" periods

$\rightarrow$ formula holds under mild conditions

Next: estimate $E[\text{good}]$ and $E[\text{bad}]$

Not difficult to show:

$$\begin{aligned} \varrho &= \frac{E[\text{good}]}{E[\text{good}] + E[\text{bad}]} \\ &\approx \frac{1}{1 + BL/F} \end{aligned}$$

where

B: bandwidth (bps)

L: length of wire (meters)

F: frame size (bits)

What does the formula say?

For example, if B is increased, what must be done to maintain high utilization?

Today: switched Ethernet

- not bus anymore but switch

  → contention moved from bus to "single point"

  → switch: a computer

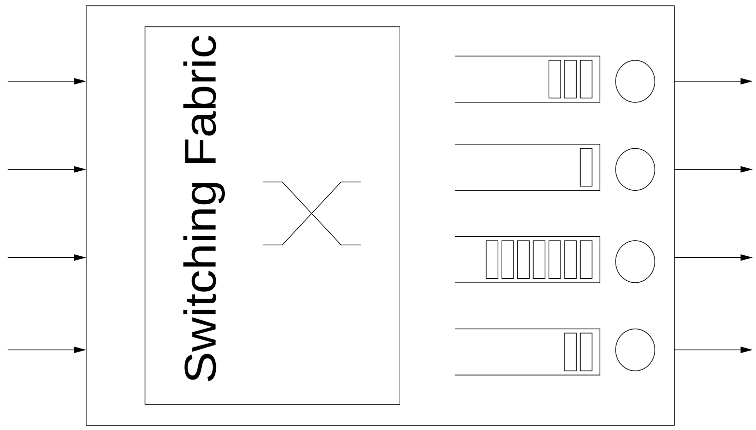
- Ethernet frames are logically scheduled

  → buffering, who goes first (FIFO, priority)

- no more physical collision

  → instead: buffer overflow

Diagram of 4-port switch:



→ output buffered switch

→ switches: both input and output buffers

→ switching fabric: hardware

→ functions: pure hardware, firmware, processes in OS

→ e.g., Cisco's router OS: IOS (Internet OS)

Note: a switch has nothing to do with CSMA/CD

→ it's not a shared bus medium with physical collisions

→ what does "switched" Ethernet mean?

Issue of backward compatibility:

- Ethernet switch emulates CSMA/CD

  → interoperate with legacy systems

  → host's CSMA/CD NIC card cannot tell difference

  → as if connected to a bus

- upon buffer overflow: send collision signal

  → switch emulates collision

  → transparent to legacy NIC

  → facilitates incremental deployment

Internet: new technology must respect legacy

→ otherwise deployment is difficult

→ key requirement of any practical solution

Long distance Ethernet: e.g., 1000Base-LX

$\longrightarrow$ what about length limit of CSMA/CD?

Medium-haul GigE/10GigE (802.3ae): 500m, 5km, 40km

- solution: disable CSMA/CD

  → switch-to-switch: disable at both ends

  → purely point-to-point link

  → backward compatibility: not an issue anymore

- flow control

  → send pause frame to prevent buffer overflow

QoS: IEEE 802.3p

$\longrightarrow$ frame tagging conveys priority

$\longrightarrow$ priority classes supported at switches

$\longrightarrow$ useful for VoIP (voice-over-IP)

Note: today's Ethernet is a hybrid mix of switch, CSMA/CD, short- and long-distance LAN

$\longrightarrow$ would not have been designed this way

$\longrightarrow$ result of legacy-respecting incremental changes