Direct Link Communication I: Wired Media

Reliable Transmission

Principal methodology: ARQ (Automatic Repeat reQuest)

$\longrightarrow$ use retransmission

$\longrightarrow$ common to wired/wireless communication

- used at link layer and higher layers (e.g., TCP)

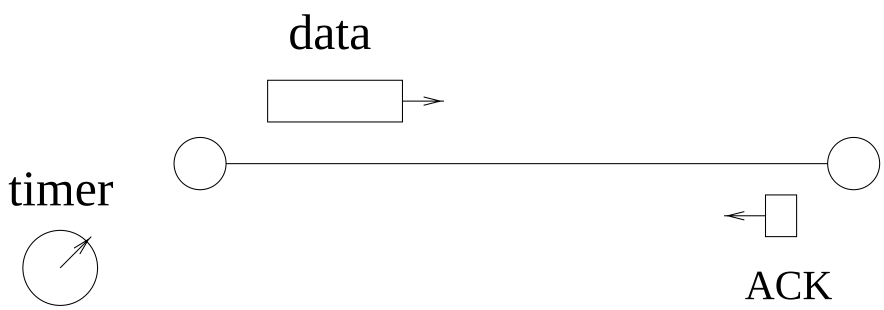
  $\rightarrow$ function duplication

- alternative: FEC

  $\rightarrow$ not assured

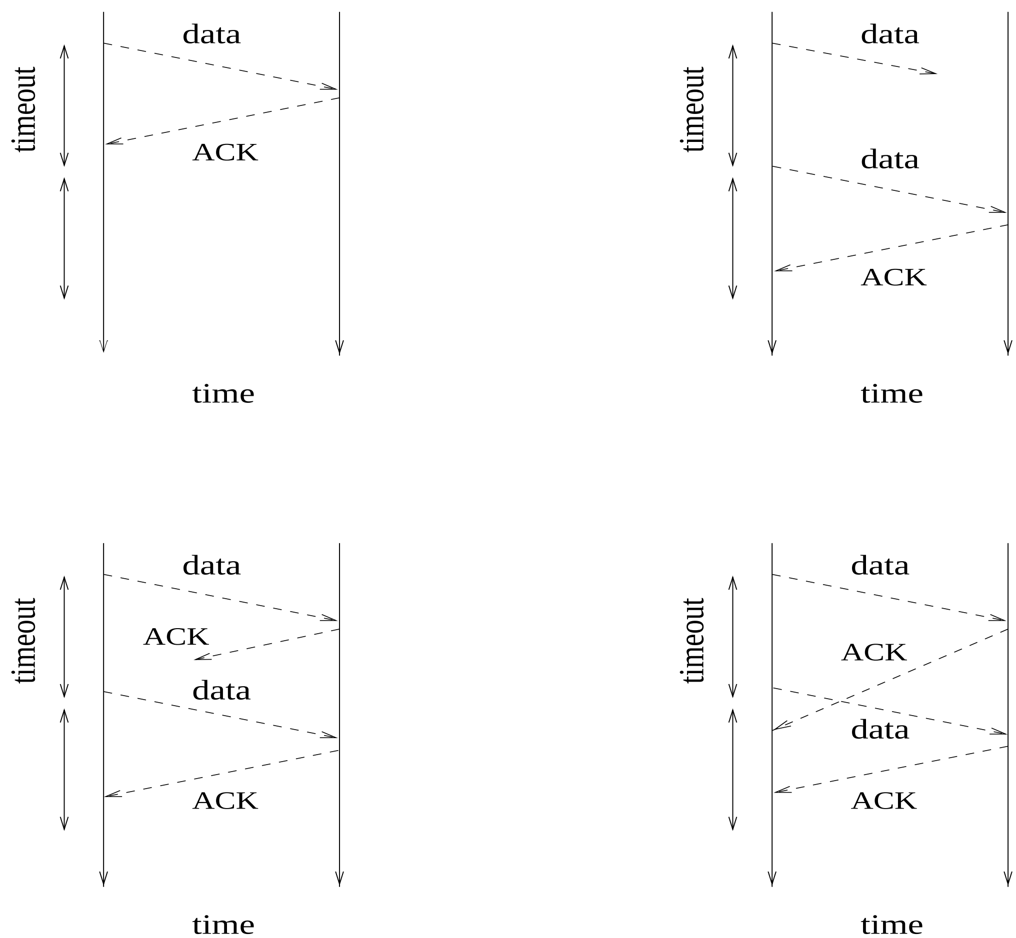
Three components:

- timer
- acknowledgment (ACK)
- retransmit



Stop-and-wait

Assumption: Frame is "lost" due to corruption; discarded by NIC after error detection.



Issue of RTT (Round-Trip Time) & timer management:

- what is proper value of timer?

  $\rightarrow$ RTT estimation

- easier for single link

  $\rightarrow$ RTT is more well-behaved

- more difficult for multi-hop path in internetwork

  $\rightarrow$ propagation latency + queueing effect

Another key problem: not keeping the pipe full.

$\longrightarrow$ delay-bandwidth product

$\longrightarrow$ volume of data travelling on the link

High throughput: want to keep the pipe full

Stop-and-wait throughput (bps):

- RTT
- frame size (bits)

$\longrightarrow$ throughput = frame size / RTT

Ex.: Link BW 1.5 Mbps, 45 ms RTT

- delay-bandwidth product:

  $\longrightarrow$ 1.5 Mbps $\times$ 45 ms = 67.5 kb $\approx$ 8 kB

- if frame size 1 kB, then throughput:

  $\longrightarrow$ $1024 \times 8/0.045 = 182$ kbps

  $\longrightarrow$ utilization: only 182 kbps/1500 kbps = 0.121

  $\longrightarrow$ note: same as 1 kB / 8 kB

Solution: increase frame size

- brute increase of frame size can be problematic
- send blocks of data, i.e., sequence of frames

Sliding window protocol

Issues:

- Shield application process from reliability management chore

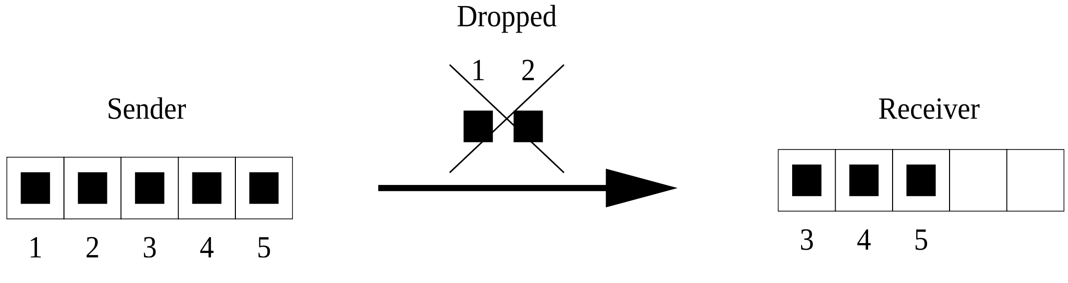
  → exported semantics: continuous byte stream

  → simple application view: e.g., **`read`** system call

- Both sender and receiver have limited buffer capacity

  → efficiency: space-bounded computation

  → task: "plug holes & flush"



Simple solution when receiver has infinite buffer capacity:

- sender keeps sending at maximum speed
- receiver informs sender of holes
- sender retransmits missing frames

$\longrightarrow$ sender's buffer capacity?

More complex solution due to bounded space:

- Sliding window technique

  → send blocks of packets

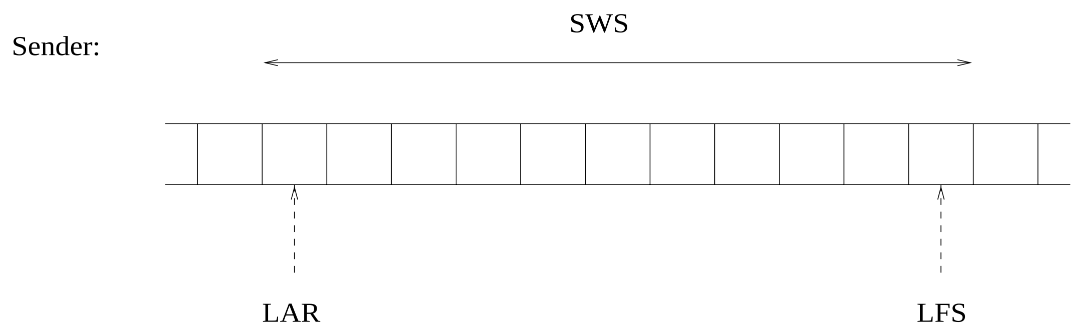
  → repair holes with positive/negative ACK

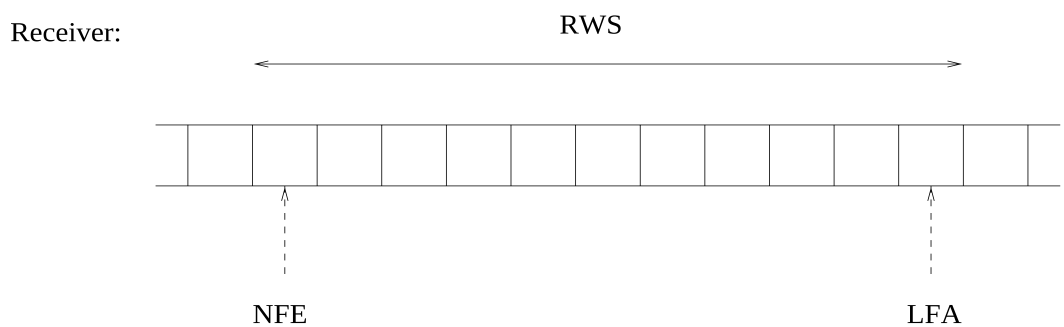
- Flow control & congestion control

  → sending too much is counterproductive

  → regulate sending rate

Set-up:





- *SWS*: Sender Window Size (sender buffer size)
- *RWS*: Receiver Window Size (receiver buffer size)
- *LAR*: Last ACK Received
- *LFS*: Last Frame Sent
- *NFE*: Next Frame Expected
- *LFA*: Last Frame Acceptable

Assign sequence numbers to frames.

$\longrightarrow$ IDs

Maintain invariants:

- $\text{LFA} - \text{NFE} + 1 \leq \text{RWS}$
- $\text{LFS} - \text{LAR} + 1 \leq \text{SWS}$

Sender:

- Receive ACK with sequence number X
- Forwind LAR to X
- Flush buffer up to (but not including) LAR
- Send up to $\text{SWS} - (\text{LFS} - \text{LAR} + 1)$ frames
- Update LFS

Receiver:

- Receive packet with sequence number Y
- Forwind to (new) first hole & update NFE

  $\longrightarrow$ NFE need not be $Y + 1$
- Send cumulative ACK (i.e., NFE)
- Flush buffer up to (but not including) NFE to application
- Update LFA $\leftarrow$ NFE + RWS $-$ 1

ACK variants:

- piggyback
- negative ACKs
- selective ACKs

Sequence number wrap-around problem:

$$\text{SWS} < (\text{MaxSeqNum} + 1)/2.$$

$\longrightarrow$ note: stop-and-wait is special binary case

Multi-Access Communication

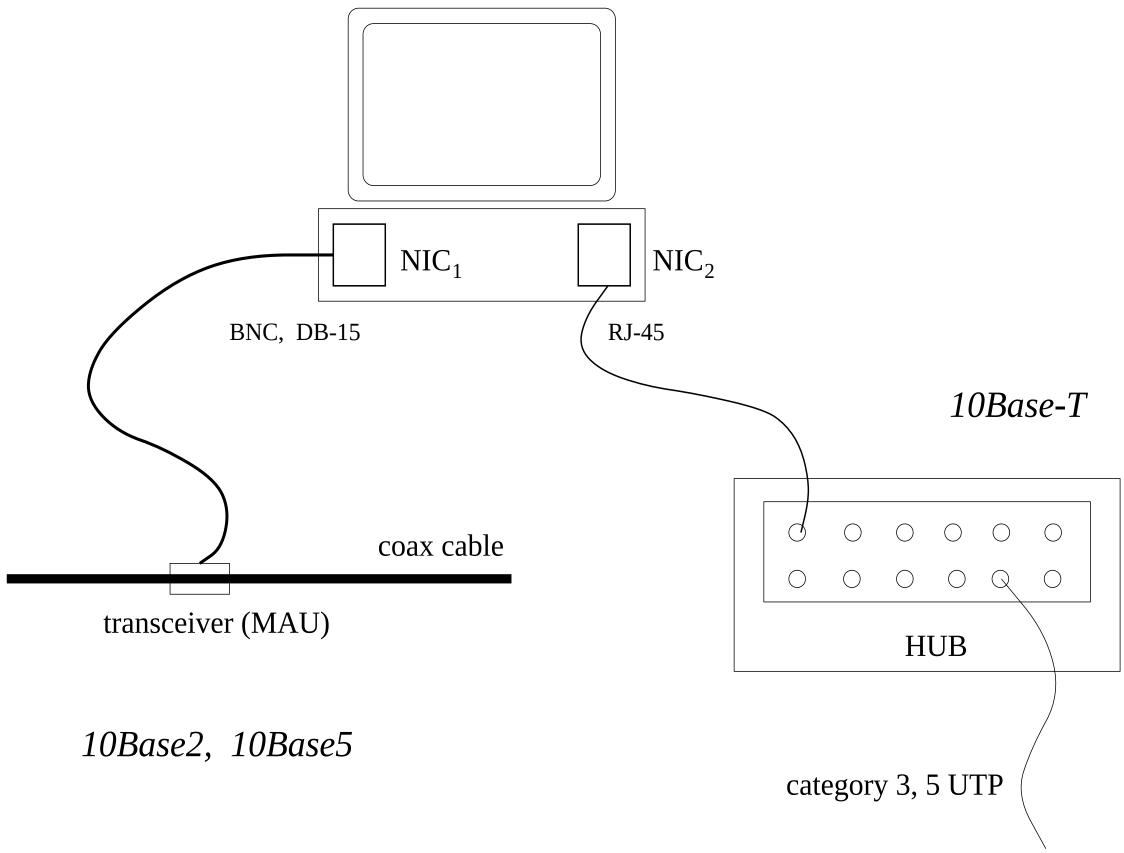
Ethernet and CSMA/CD

$\longrightarrow$ copper, fiber media

Types:

- 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

Connectivity example:



⟶ physical network: bus (old) vs. switch (new)

⟶ multihomed/singlehomed

⟶ unique Ethernet address per NIC

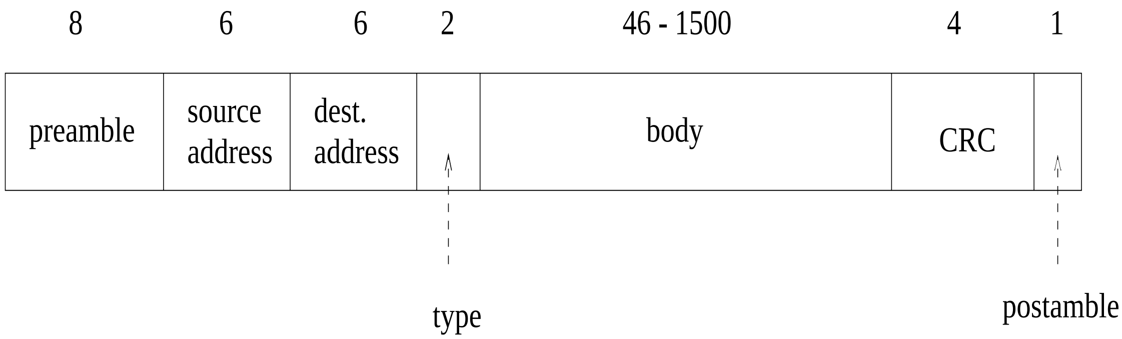
Segments can be hooked up by repeaters, bridges, gateways, hubs or switches.

- maximum of 2 (4 for IEEE 802.3) repeaters between two hosts; 1500 m
- for Fast Ethernet, 2 repeater hops

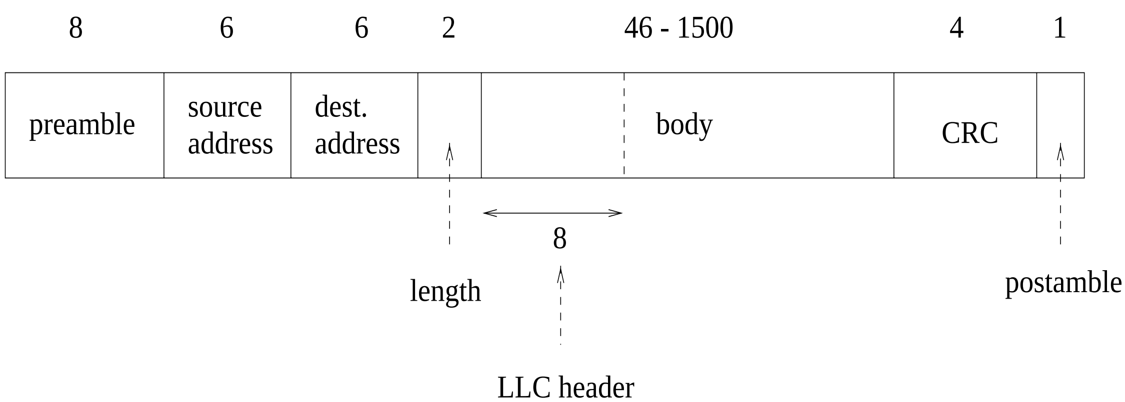
High-bandwidth Ethernets have *shorter* network diameter.

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

DIX Ethernet frame:



IEEE 802.3 Ethernet frame:



⟶ IEEE 802.2 LLC (Logical Link Control)

⟶ optional feature of LLC: ARQ

Encoding: Manchester

Addressing:

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

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

- accepts only own frame address
- accepts all frames: promiscuous mode

  $\longrightarrow$ NIC feature

MAC (Medium Access Control) approaches:

$\longrightarrow$ i.e., multiplexing problem

- broadband: FDM, CDMA
- baseband: TDM, multiple access

  → why not just use TDM?

Multiple access for baseband:

- Time slots are available for grab

  → like "on-demand" TDM
- To grab is to send

  → speak first (e.g., TV talk shows)
- If ≥ 2 users grab at the same time, slot becomes junk

  → collision

Benefits of multiple access:

- When not too many users, faster response time

  → don't need to go through registration & reservation phase (TDM)

  → registration: admission control

- Decentralized

  → no central coordinator

  → simple; self-organization

Drawbacks of multiple access:

- When many users, degraded response & throughput

  → collision wastes slots, i.e., bandwidth

- Lack of quality of service assurances

  → "you get what you get"; a form of best effort

  → problematic for real-time traffic, e.g., telephony