# Lecture 5: MAC Layer

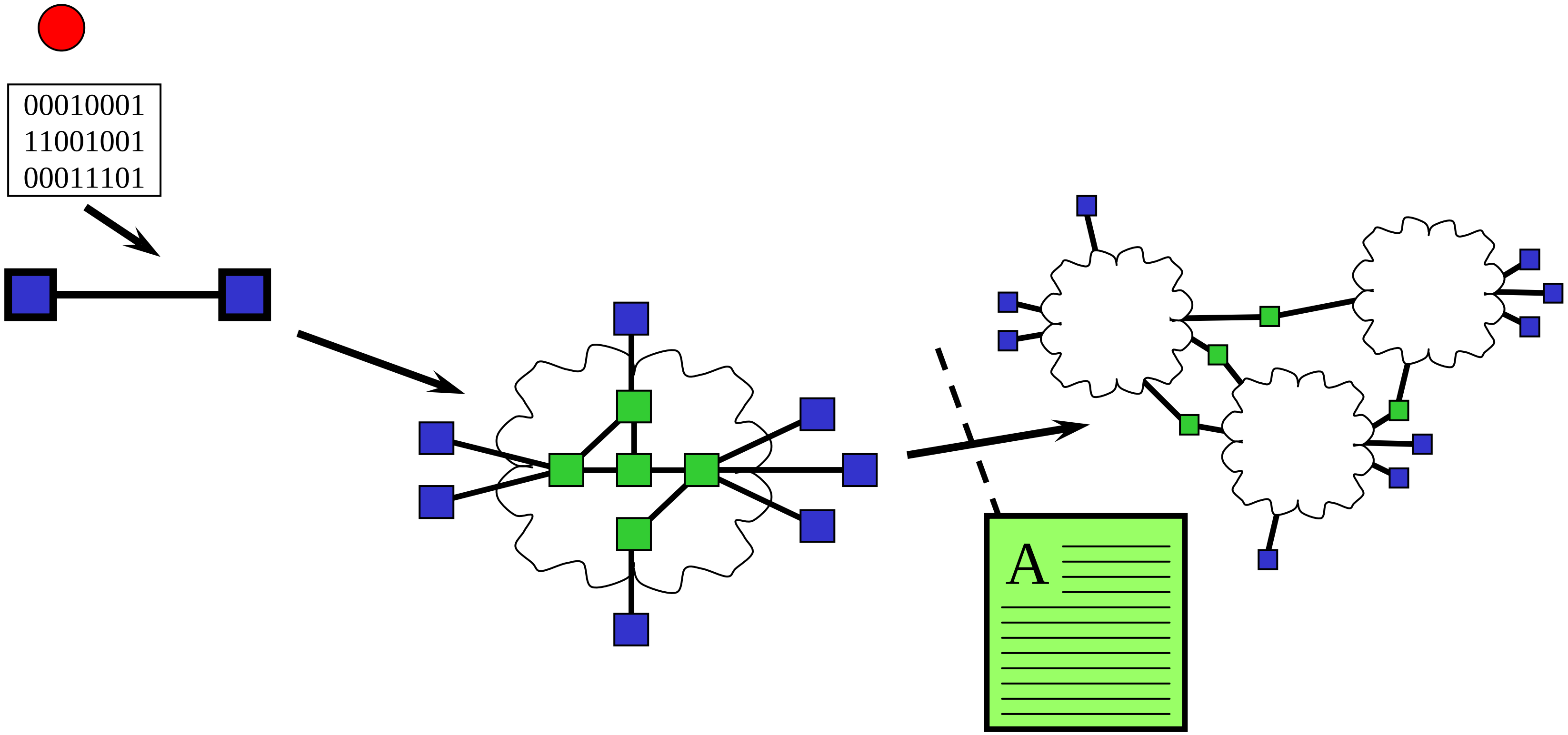
CS/ECE 438: Communication Networks

Prof. Matthew Caesar

February 17, 2010

# Where are We?

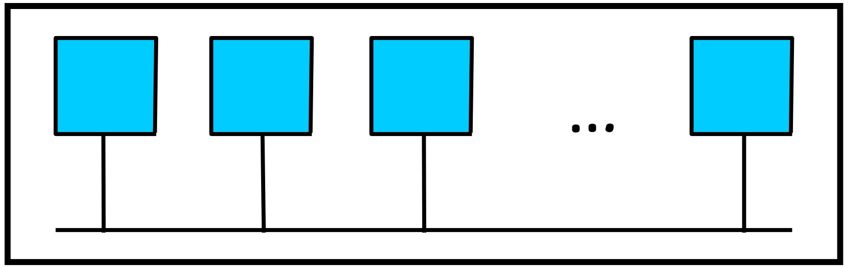
you are here




midterm is here

# Multiple Access Media

- Multiple senders on some media
  - Buses (Ethernet)
  - Radio, Satellite
  - Token Ring
- Need methods to mediate access
  - Fair arbitration
  - Good performance



# Multiple Access Media

- Typical assumptions
  - Communication needs vary
    - Over time
    - Between hosts
  - Network is not fully utilized
- Recall methods for multiplexing
  - Frequency-division multiplexing (FDM, separate bands)
  - Time-division multiplexing (TDM, synchronous time slots)
  - Statistical time-division multiplexing (STDM, time slots on demand)
- STDM most appropriate with stated assumptions

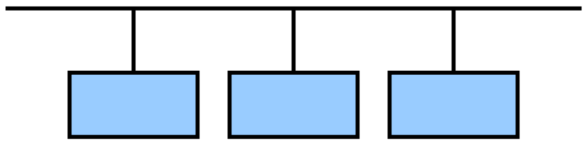
# Multiple Access Media

- Problem
  - Demands can conflict
    - Two hosts send simultaneously
  - STDM does not address problem
- Solution
  - Medium access control (MAC) algorithm

# Multiple Access Media

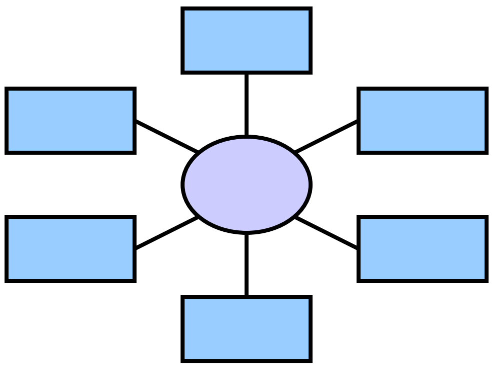
- Three solutions (of many)
  - Carrier sense multiple access with collision detection (CSMA/CD)
    - Send only if medium is idle
    - Stop sending immediately if collision detected
  - Token ring/Fiber Distributed Data Interface (FDDI)
    - Pass a token around a ring; only token holder sends
  - Carrier sense multiple access with collision Avoidance (CSMA/CA)
    - Send only if medium is idle
    - Design send algorithm to avoid collisions
- Tanenbaum Sec. 4.2 covers many others

# Types of Shared Link Networks



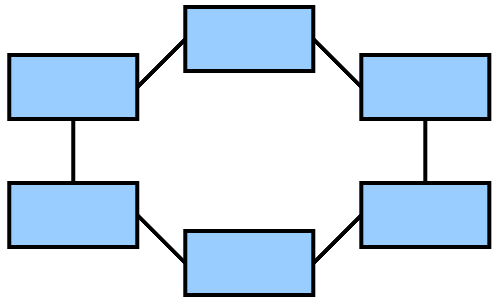
Bus Topology: Shared

Ethernet, Token Bus



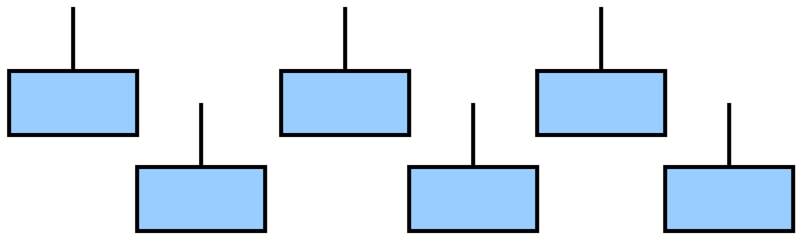
Star Topology: Active or Passive Hub

ATM



Ring Topology: Multihop

FDDI, IEEE 802.5



Wireless: Shared

IEEE 802.11

# Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

- Used by Ethernet
  - Xerox and IEEE 802.3 (10Mbps standards)
  - IEEE 802.3u (Fast Ethernet, 100Mbps standard)
  - IEEE 802.3z,ab (1Gbps Ethernet)
  - IEEE 802.3-2005/8 (10 Gbps Ethernet)
- Outline
  - Historical development
  - Topologies and components
  - MAC algorithm
  - Collision detection limitations
  - Lessons learned

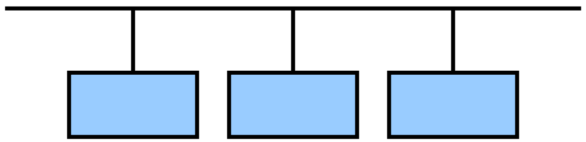
# Ethernet

- History
  - Developed by Xerox PARC, mid 70s
  - Roots in ALOHA packet radio network
  - Standardized by Xerox, DEC and Intel in 1978
  - Similar to IEEE 802.3
  - IEEE 802.3u standard defines Fast Ethernet (100 Mbps)
  - Switched Ethernet now popular
- Bandwidth
  - 10 Mbps – 100 Mbps – 1 Gbps – 10 Gbps
- Problem
  - Distributed algorithm that provides fair efficient access to a share medium

# Ethernet - CSMA/CD

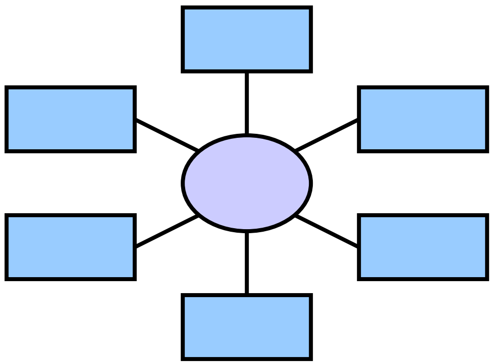
- CS – Carrier Sense
  - Nodes can distinguish between an idle and a busy link
- MA - Multiple Access
  - A set of nodes send and receive frames over a shared link
- CD – Collision Detection
  - Nodes listen during transmission to determine if there has been interference

# Ethernet Topologies



Bus Topology: Shared

All nodes connected to a wire



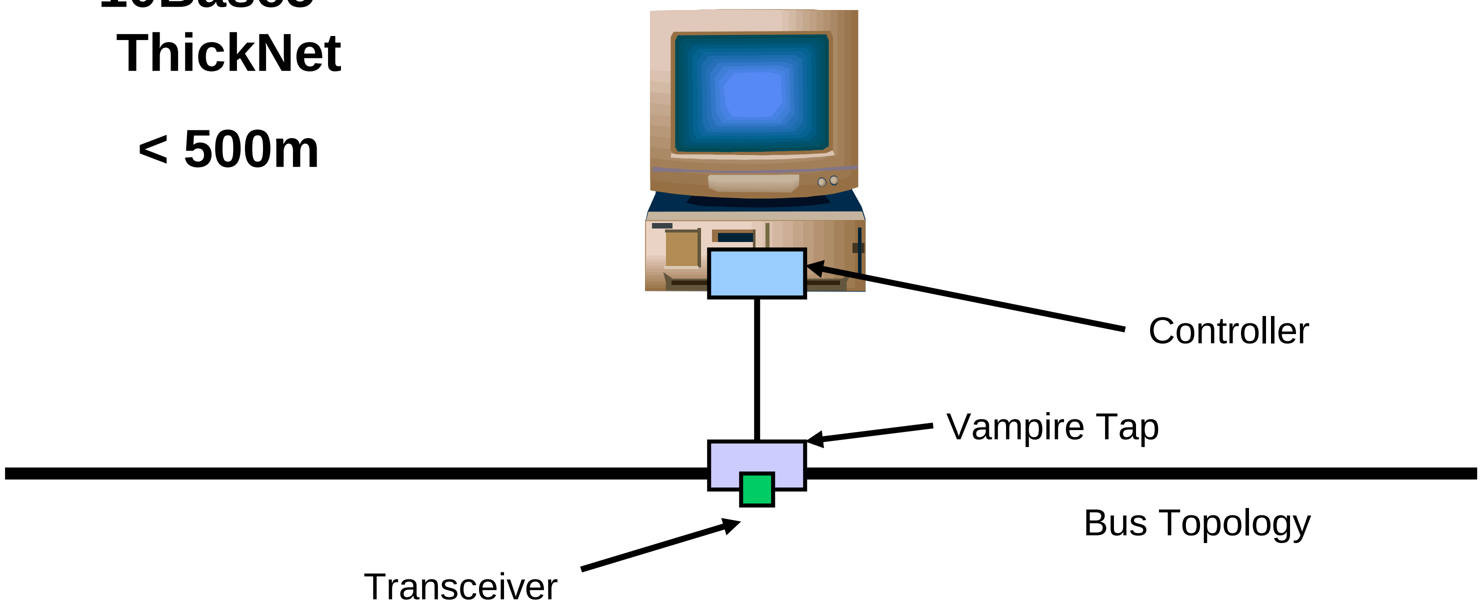
Star Topology:

All nodes connected to a central repeater

# Ethernet Connectivity

**10Base5 – ThickNet**

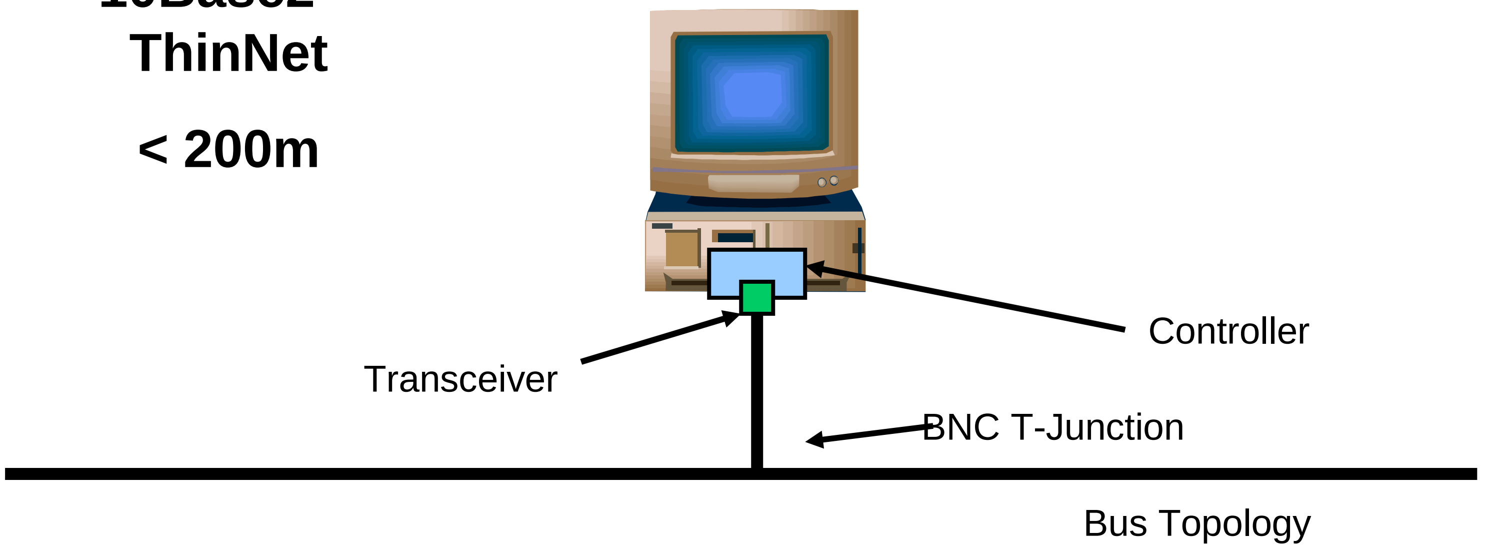
**< 500m**



# Ethernet Connectivity

**10Base2 – ThinNet**

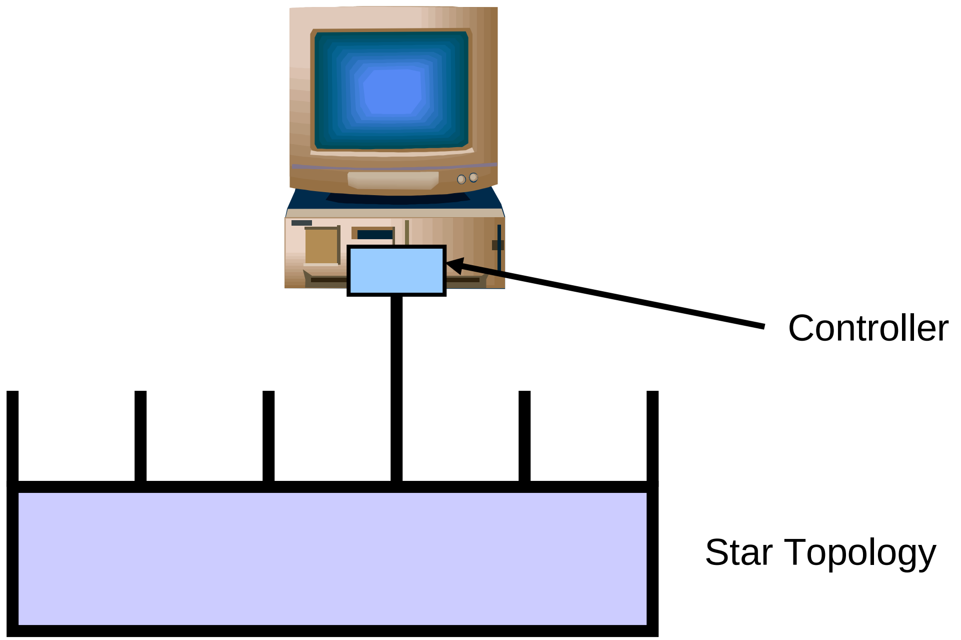
**< 200m**



# Ethernet Connectivity

**10BaseT**

**< 100m**



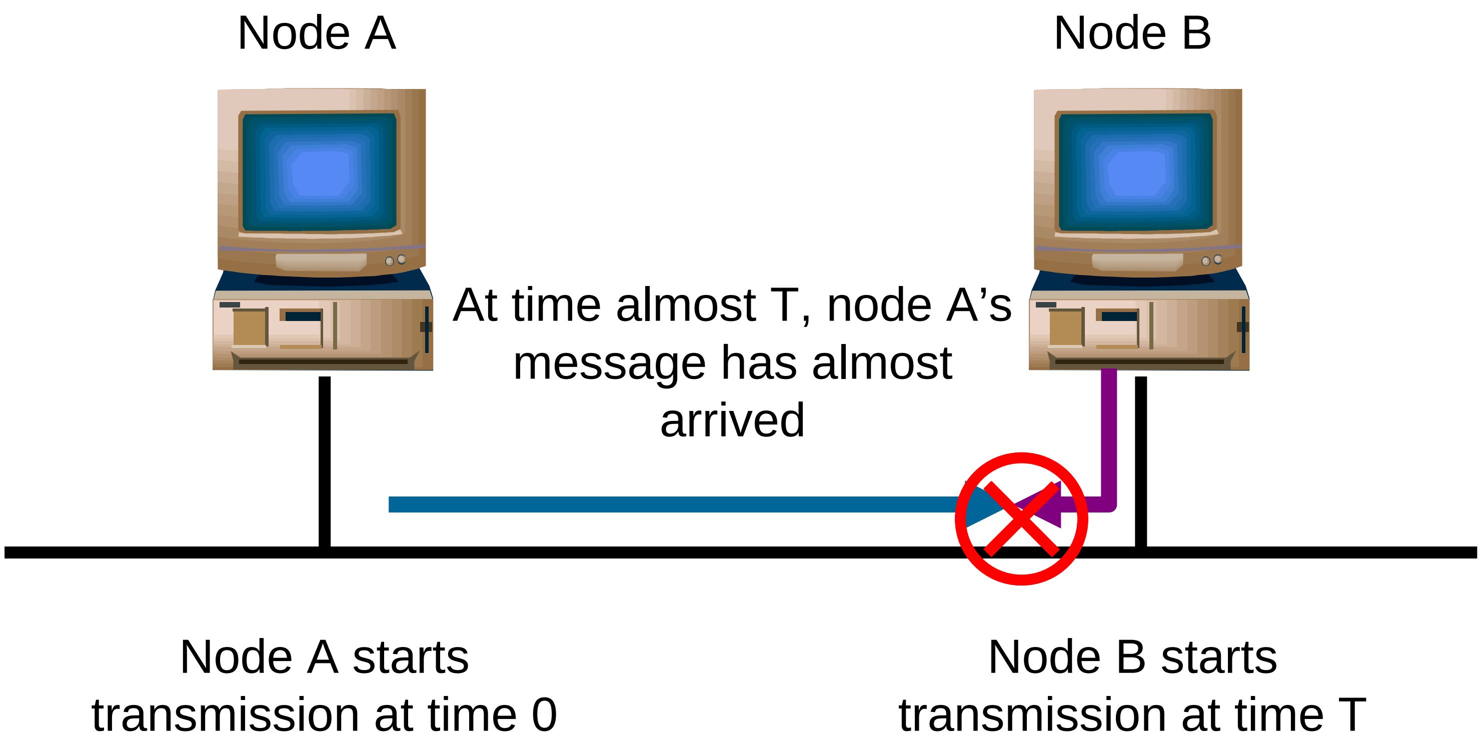
# Ethernet Specifications

- Coaxial Cable
  - up to 500m (protocol makes assumption about maximum time before you can hear from another host on the wire)
- Taps
  - > 2.5m apart
- Transceiver
  - Idle detection
  - Sends/Receives signal
- Repeater
  - Joins multiple Ethernet segments
  - < 5 repeaters between any two hosts (limits length)
- < 1024 hosts

# 10Mb Ethernet Specifications

- Broadcast (everyone receives it)
- Encoding
  - Manchester
  - 10 Mbps $\Rightarrow$ Transmission at 20Mhz
- Framing
  - Sentinel marks end of frame
  - Bit oriented (similar to HDLC)
  - Data-dependent length
- Error Detection
  - 32-bit CRC

# Ethernet MAC Algorithm

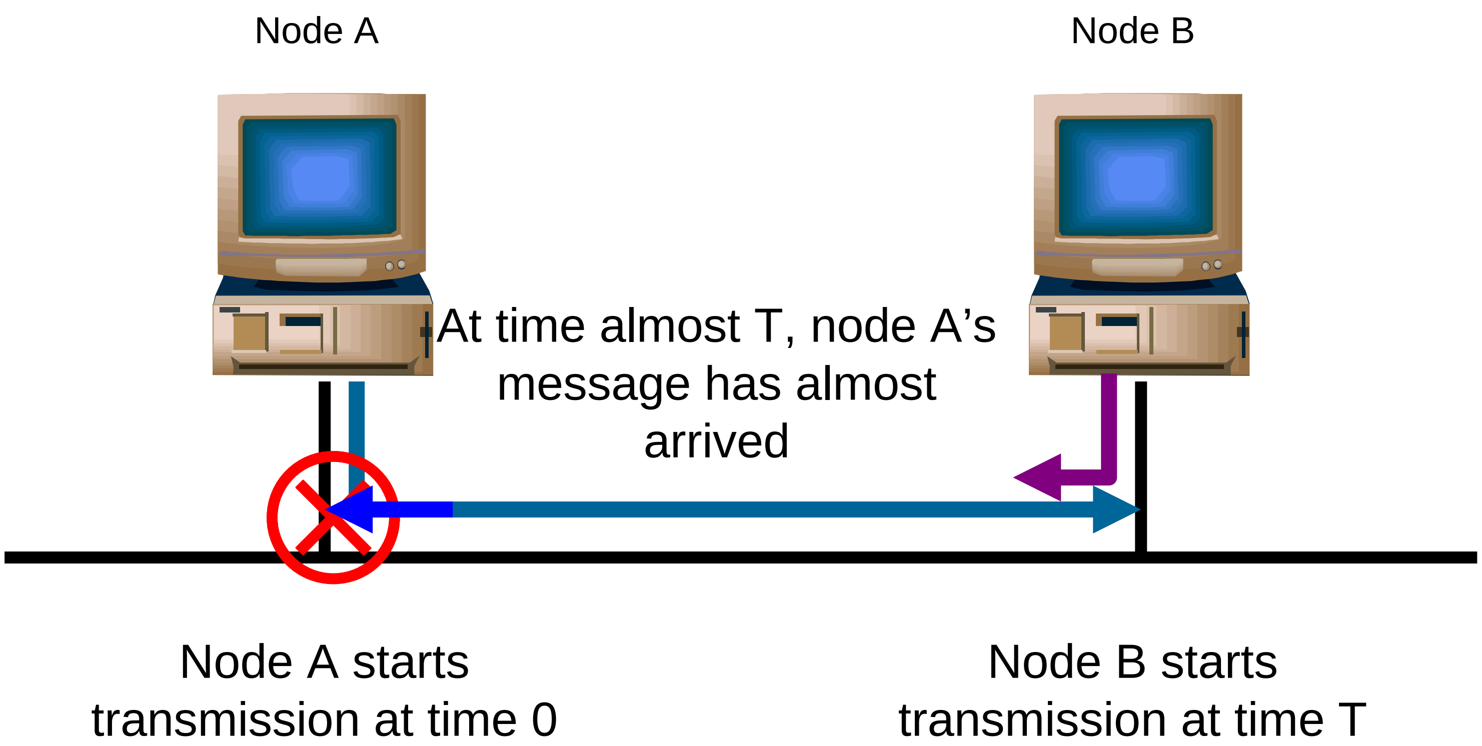



How can we ensure that A knows about the collision?

# Collision Detection

- Problem
  - How can A detect a collision?
- Solution
  - A must still be transmitting when it receives B's transmission!
- Example
  - Node A's message reaches node B at time T
  - Node B's message reaches node A at time 2T
  - For node A to detect a collision, node A must still be transmitting at time 2T

# Ethernet MAC Algorithm




At time 2T, A is still transmitting and notices a collision

# Collision Detection

- IEEE 802.3
  - 2T is bounded to 51.2μs
  - At 10Mbps 51.2μs = 512b or 64 = 512b or 64B
  - Packet length ≥ 64B
- Jam after collision
  - Ensures that all hosts notice the collision

# Ethernet MAC Algorithm

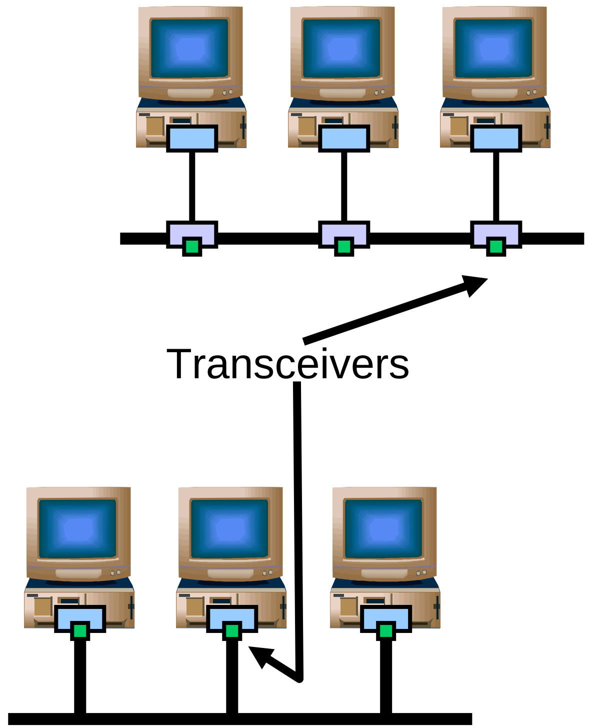
- Sender/Transmitter
  - If line is idle (carrier sensed)
    - Send immediately
    - Send maximum of 1500B data (1527B total)
    - Wait 9.6 μs before sending again
  - If line is busy (no carrier sense)
    - Wait until line becomes idle
    - Send immediately (1-persistent)
  - If collision detected
    - Stop sending and jam signal
    - Try again later

# Frame Reception

- Sender handles all access control
- Receiver simply pulls the frame from the network
- Ethernet controller/card
  - Sees all frames
  - Selectively passes frames to host processor
- Acceptable frames
  - Addressed to host
  - Addressed to broadcast
  - Addressed to multicast address to which host belongs
  - Anything (if in promiscuous mode)
    - Need this for packet sniffers/TCPDump

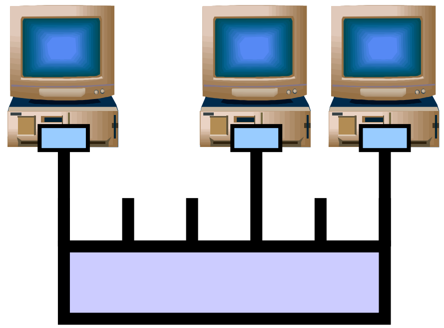
# Collision Detection Techniques: Bus Topology

- Transceiver handles
  - Carrier detection
  - Collision detection
  - Jamming after collision
- Transceiver sees sum of voltages
  - Outgoing signal
  - Incoming signal
- Transceiver looks for
  - Voltages impossible for only outgoing



# Collision Detection Techniques: Hub Topology

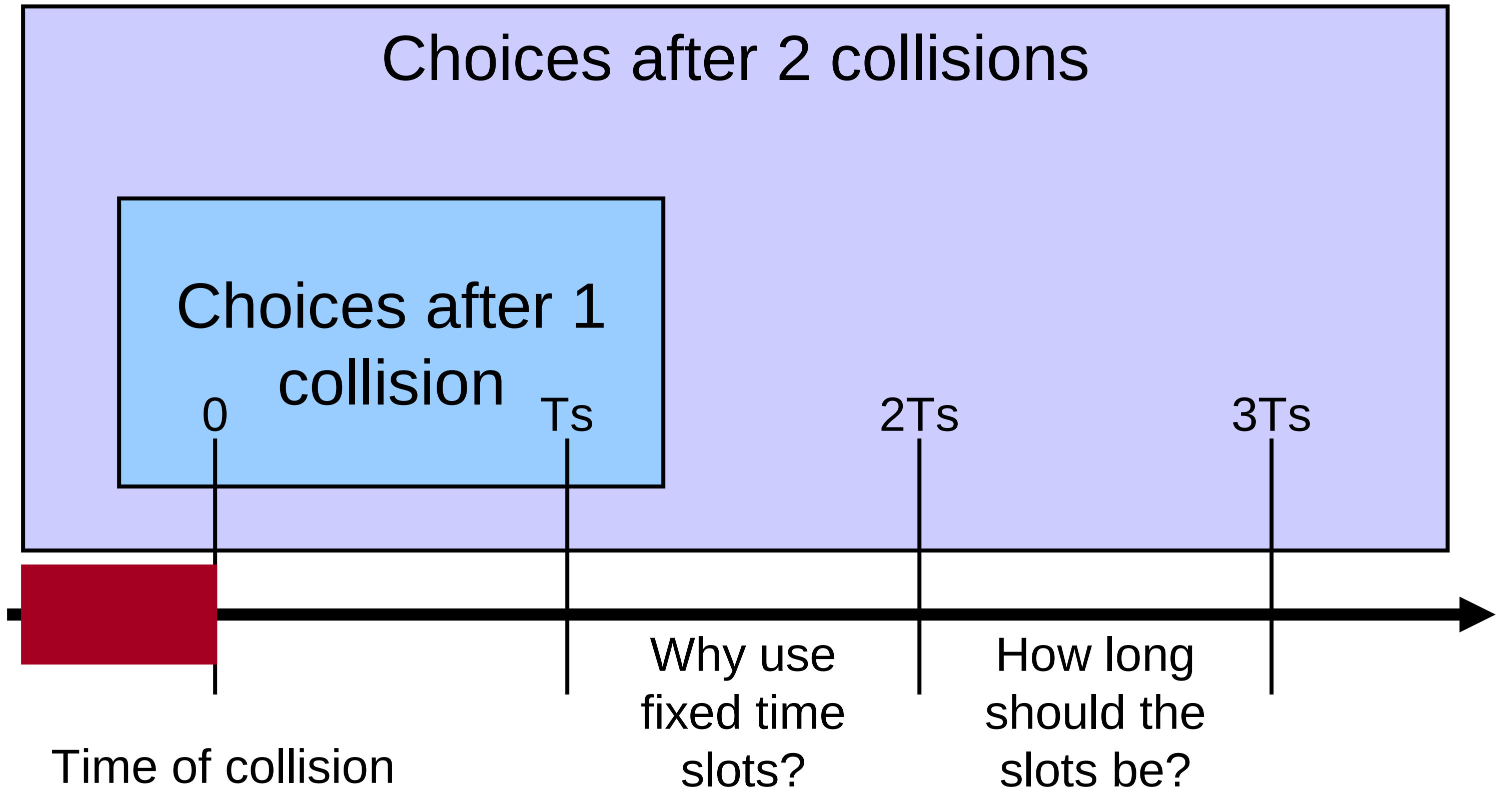
- Controller/Card handles
  - Carrier detection
- Hub handles
  - Collision detection
  - Jamming after collision
- Need to detect activity on all lines
  - If more than one line is active
    - Assert collision to all lines
    - Continue until no lines are active



# Retransmission

- How long should a host wait to retry after a collision?
- What happens if the host waits too long?
  - Wasted bandwidth
- What happens if the host doesn't wait long enough?
  - More collisions
- Ethernet Solution
  - Binary exponential backoff
    - Maximum backoff doubles with each failure
    - After N failures, pick an N-bit number
    - $2^N$ discrete possibilities from 0 to maximum

# Binary Exponential Backoff



# Binary Exponential Backoff

- For IEEE 802.3, T = 51.2 μs
- Consider the following
  - k hosts collide
  - Each picks a random number from 0 to $2^{(N-1)}$
  - If the minimum value is unique
    - All other hosts see a busy line
    - Note: Ethernet RTT < 51.2 μs
  - If the minimum value is not unique
    - Hosts with minimum value slot collide again!
    - Next slot is idle
    - Consider the next smallest backoff value

# 10Mbps Ethernet Media

| Name | Cable | Advantages | Max. Segment Length | Max Nodes on Segment |
|---|---|---|---|---|
| **10Base5** | Thick Coaxial (10mm) | Good for backbones | 500m | 100 |
| **10Base2** | Thin Coaxial (5mm) | Cheapest system | 200m | 30 |
| **10BaseT** | Twisted Pair (0.5mm) | Easy Maintenance | 100m | 1 (to hub) |
| **10BaseFP** | Fiber (0.1mm) | Best between buildings | 500m | 33 |
| Extended segments may have up to 4 repeaters (total of 2.5km) | | | | |

# 100Mbps Ethernet Media

| Name | Cable | Max. Segment Length | Advantages |
|---|---|---|---|
| **100BaseT4** | 4 Twisted Pair | 100m | Cat 3, 4 or 5 UTP |
| **100BaseTX** | Twisted Pair | 100m | Full duplex on Cat 5 UTP |
| **100BaseFX** | Fiber Pair | 100m | Full duplex, long runs |
| All hub based. Other types not allowed. Hubs can be shared or switched | | | |

# Ethernet in Practice

- Number of hosts
  - Limited to 200 in practice, standard allows 1024
- Range
  - Typically much shorter than 2.5km limit in standard
- Round Trip Time
  - Typically 5 or 10 μs, not 50
- Flow Control
  - Higher level flow control limits load (e.g. TCP)
- Topology
  - Star easier to administer than bus

# Token Ring

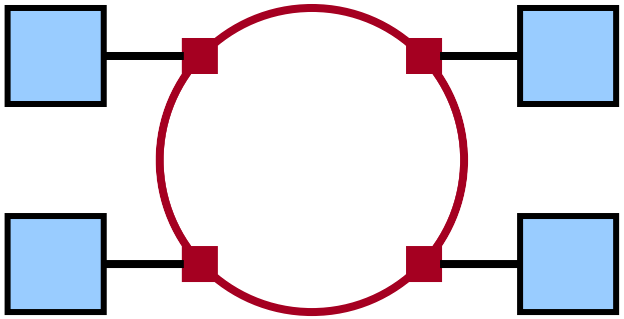
- Focus on Fiber Distributed Data Interface (FDDI)
  - 100 Mbps
  - Was (not is) a candidate to replace Ethernet
  - Used in some MAN backbones (LAN interconnects)
- Outline
  - Rationale
  - Topologies and components
  - MAC algorithm
  - Priority
  - Feedback
  - Token management

# Token Ring

- Why emulate a shared medium with point-to-point links?
- Why a shared medium?
  - Convenient broadcast capabilities
  - Switches costly
- Why emulation?
  - Simpler MAC algorithm
  - Fairer access arbitration

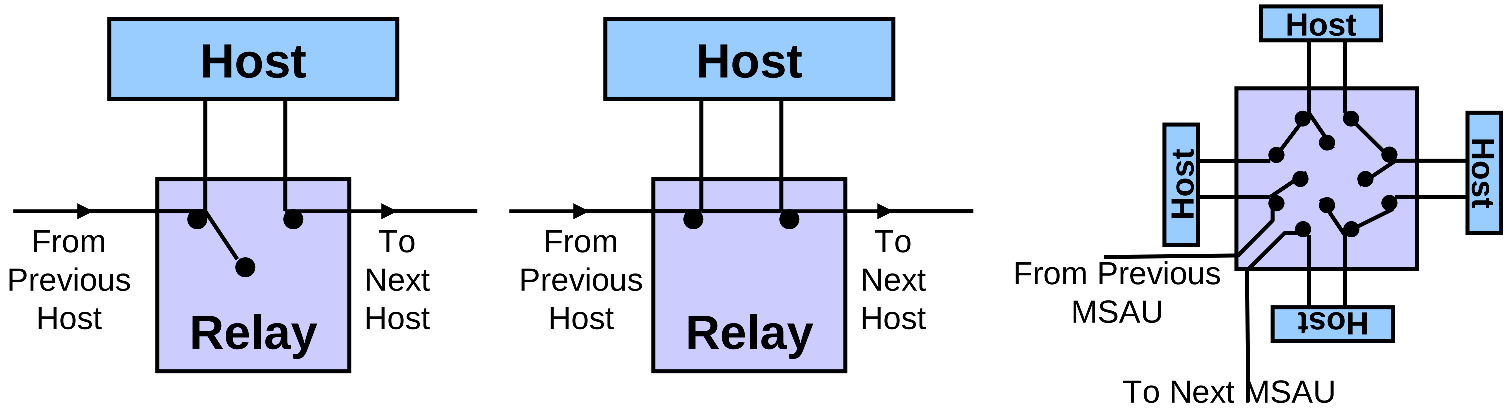
# Token Ring

- Example Token Ring Networks
  - IBM: 4Mbps token ring
  - IEEE 802.5: 16Mbps



# Token Ring: Topology and Components

- Relay
  - Single Relay
  - Multistation access units

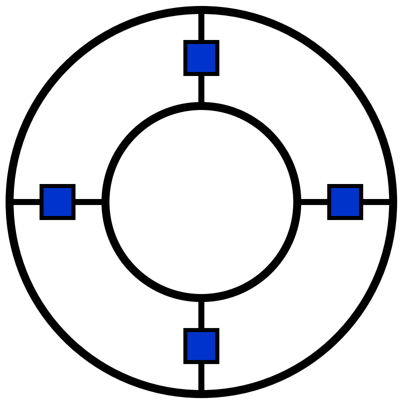


# Multistation Access Unit

- Each station imposes a delay
  - e.g., 50 ms
- Maximum of 500 Stations
- Upper limit of 100km
  - Need 200km of fiber
- Uses 4B/5B encoding
- Can be implemented over copper

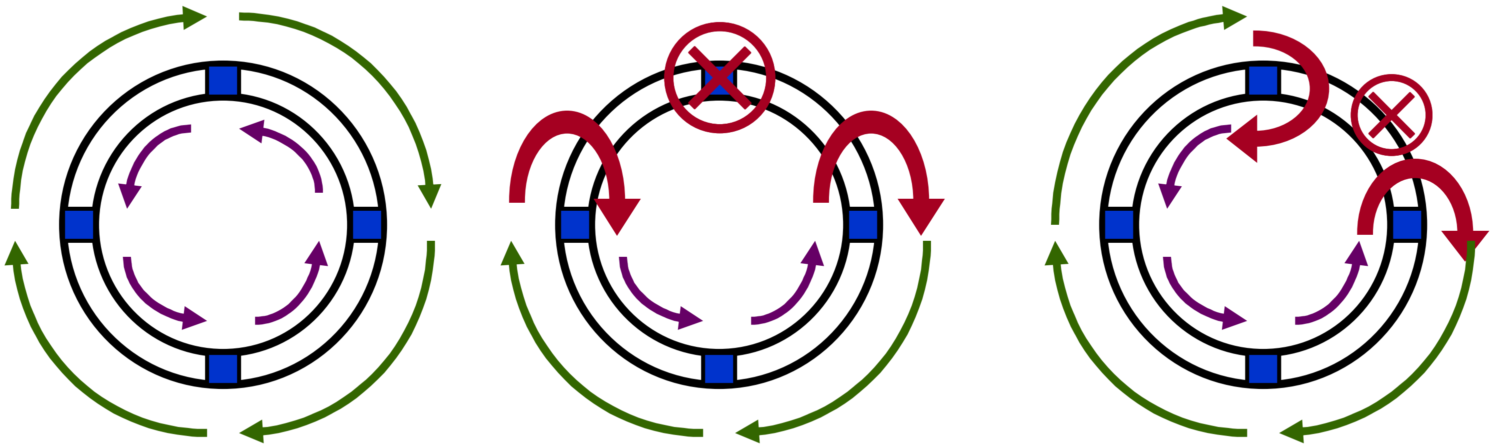
# Token Ring: Dual Ring

- Example Token Ring Networks
  - FDDI: 1000Mbps
    - Fiber Distributed Data Interface



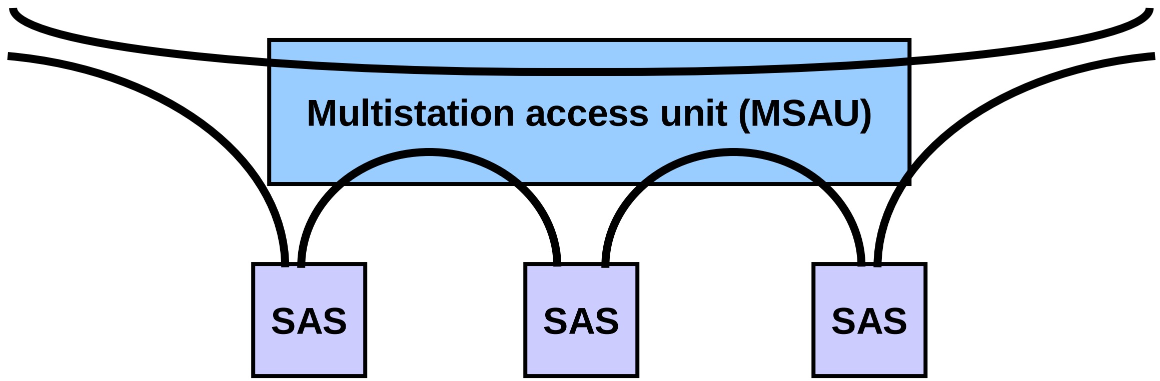
# FDDI

- Dual ring configuration
  - Self-healing
    - Normal flow in green direction
    - Can detect and recover from one failure



# FDDI: Topology and Components

- Single and Dual attachment stations
  - SAS, DAS



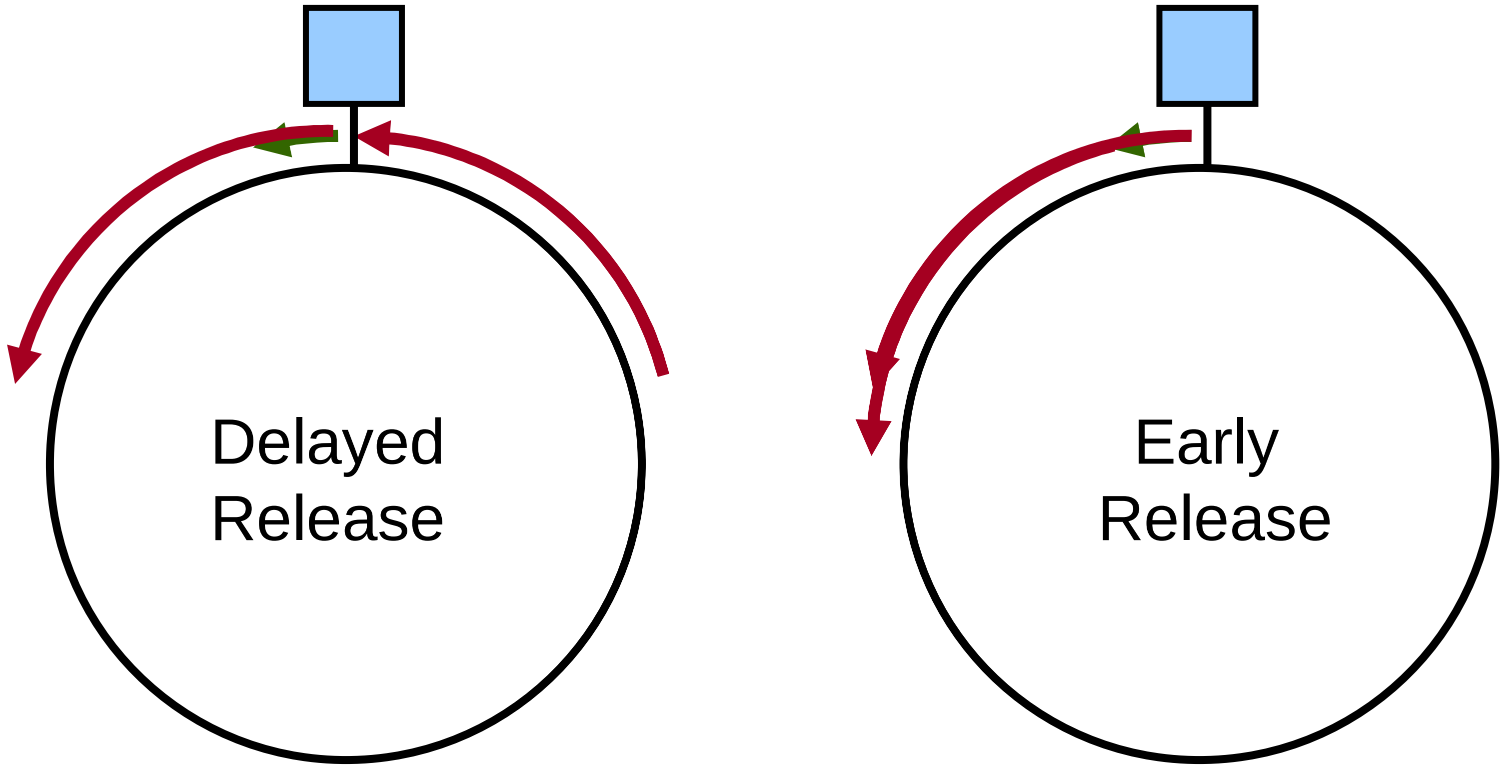
# Token Ring: Basic Concepts

- Frames flow in one direction
  - Upstream to downstream
- Token
  - Special bit pattern rotates around ring
- Stations
  - Must capture token before transmitting
  - Must remove frame after it has cycled
  - Must release token after transmitting
- Service
  - Stations get round-robin service

# Token Ring: Basic Concepts

- Immediate release
  - Used in FDDI
  - Token follows last frame immediately
- Delayed release
  - Used in IEEE 802.5
  - Token sent after last frame returns to sender

# Token Release



# Token Maintenance: IEEE 802.5

- Monitoring for a Valid Token
  - All stations should periodically see valid transmission (frame or token)
  - Maximum gap
    - = ring latency + max frame < = 2.5ms
  - Set timer at 2.5ms
    - send claim frame if timer expires

# 802.5 Reliability

- Delivery status
  - Trailer
    - A bit
      - Set by recipient at start of reception
    - C bit
      - Set by recipient on completion on reception

# 802.5 Monitor

- Responsible for
  - Inserting delay
  - Token presence
    - Should see a token at least once per round
  - Check for corrupted frames
  - Check for orphaned frames
    - Header
      - Monitor bit
        - » Monitor station sets bit first time it sees packet
        - » If monitor sees packet again, it discards packet

# Token Ring: Media Access Control Parameters

- Token Holding Time (THT)
  - Upper limit on how long a station can hold the token
  - Each station is responsible for ensuring that the transmission time for its packet will not exceed THT
- Token Rotation Time (TRT)
  - How long it takes the token to traverse the ring.
  - TRT $\leq$ ActiveNodes x THT + RingLatency
- Target Token Rotation Time (TTRT)
  - Agreed-upon upper bound on TRT

# Timing Algorithm: IEEE 802.3

- Each node measures TRT between successive tokens
  - If measured-TRT > TTRT
    - Token is late
    - Don’t send
  - If measured-TRT < TTRT
    - Token is early
    - OK to send

# Traffic Classes: FDDI

- Two classes of traffic
  - Synchronous
    - Real time traffic
    - Can always send
  - Asynchronous
    - Bulk data
    - Can send only if token is early

# Timing Algorithm: FDDI

- Each station is allocated $S_i$ time units for synchronous traffic per TRT
- TTRT is negotiated
  - $S_1 + S_2 + \ldots + S_N + \text{RingLatency} \leq \text{TTRT}$
- Algorithm Goal
  - Keep actual rotation time less than TTRT
  - Allow station i to send $S_i$ units of synchronous traffic per TRT
  - Fairly allocate remaining capacity to asynchronous traffic
  - Regenerate token if lost

# Timing Algorithm: FDDI

- When a node gets the token
  - Set TRT = time since last token
  - If TRT > TTRT
    - Token is late
    - Send synchronous data
    - Don’t send asynchronous data
  - If TRT < TTRT
    - Token is early
    - OK to send any data
      - Send synchronous data
      - Set THT = TTRT – TRT
      - If THT > 0, send asynchronous data

# FDDI Performance

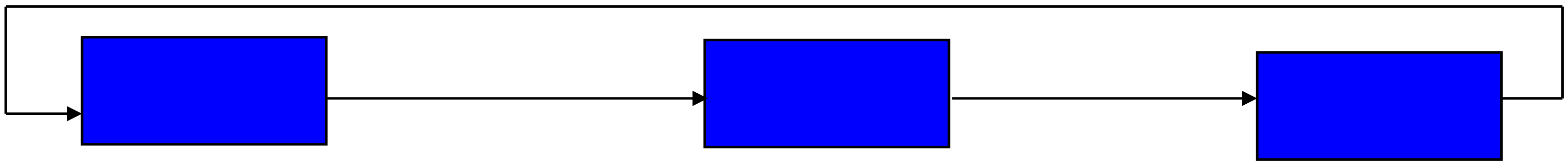
- Synchronous traffic may consume one TTRT worth of time
  - TRT > TTRT
- Worst case
  - TRT < 2*TTRT
  - Any asynchronous traffic plus RingLatency ≤ TTRT
  - Synchronous traffic < TTRT

# FDDI Performance

- Can't have two consecutive TRT = 2*TTRT
  - After a cycle with TRT = 2*TTRT, no asynchronous traffic will be sent

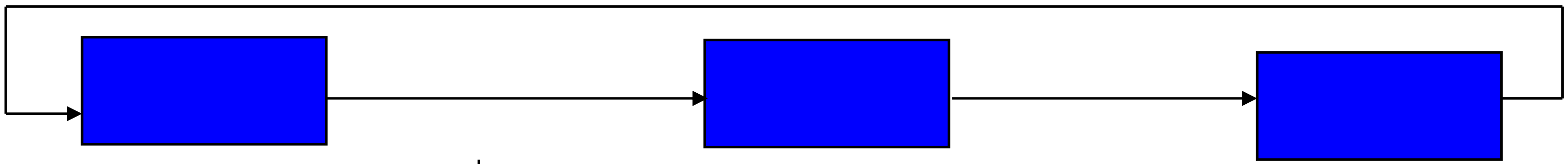
  

# FDDI example



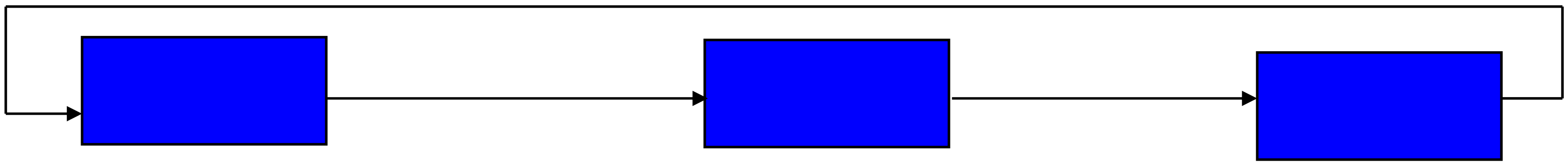
- Assume
  - RingLatency=12 µs
  - Three active stations
  - Each with $S_i$=20 µs
  - TTRT=100 µs
  - Stations have unlimited supply of asynchronous traffic.

# FDDI example



| Arrival time | TRT | s/a | Arrival time | TRT | s/a | Arrival time | TRT | s/a |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0/0 | 4 | 0 | 0/0 | 8 | 0 | 0/0 |
| 12 | 12 | 20/88 | 124 | 120 | 20/0 | 148 | 140 | 20/0 |
| 172 | 160 | 20/0 | 196 | 92 | 20/8 | 228 | 120 | 20/0 |
| 252 | 140 | 20/0 | 276 | 80 | 20/20 | 320 | 112 | 20/0 |
| 344 | 132 | 20/0 | 368 | 92 | 20/8 | 400 | 92 | 20/8 |
| 432 | 120 | 20/0 | 456 | 88 | 20/12 | 492 | 92 | 20/8 |

# FDDI example



| Arrival time | TRT | A | Arrival time | TRT | A | Arrival time | TRT | A |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 4 | 0 | 0 | 8 | 0 | 0 |
| 12 | 12 | 88 | 124 | 120 | 0 | 148 | 140 | 0 |
| 172 | 160 | 0 | 196 | 92 | 8 | 228 | 120 | 0 |
| 252 | 140 | 0 | 276 | 80 | 20 | 320 | 112 | 0 |
| 344 | 132 | 0 | 368 | 92 | 8 | 400 | 92 | 8 |
| 432 | 120 | 0 | 456 | 88 | 12 | 492 | 92 | 8 |
| 524 | 112 | 0 | 548 | 92 | 8 | 580 | 88 | 12 |
| 616 | 104 | 0 | 640 | 92 | 8 | 672 | 92 | 8 |
| 704 | 92 | 8 | 736 | 96 | 4 | 764 | 92 | 8 |
| 796 | 92 | 8 | 828 | 92 | 8 | 860 | 96 | 4 |
| 888 | 92 | 8 | 920 | 92 | 8 | 952 | 92 | 8 |
| 984 | 96 | 4 | 1012 | 92 | 8 | 1044 | 92 | 8 |
| 1076 | 92 | 8 | 1108 | 96 | 4 | 1136 | 92 | 8 |
| 1168 | 92 | 8 | 1200 | 92 | 8 | 1232 | 96 | 4 |
| 1260 | 92 | 8 | 1292 | 92 | 8 | 1324 | 92 | 8 |
| 1356 | 96 | 4 | 1384 | 92 | 8 | 1416 | 92 | 8 |
| 1448 | 92 | 8 | 1480 | 96 | 4 | 1508 | 92 | 8 |

# Token Maintenance: FDDI

- Lost Token
  - No token when initializing ring
  - Bit error corrupts token pattern
  - Node holding token crashes
- Monitoring for a valid token
  - Should see valid transmission (frame or token) periodically – within 2*TTRT
  - Maximum gap = RingLatency + MaxFrame $\leq$ 2.5ms

# Token Maintenance: FDDI

- Generating a Token (and agreeing on TTRT)
  - Execute when joining ring or suspect a failure
  - Send a claim frame that includes the node's TTRT bid
    - When receive claim frame, update the bid and forward
    - If your claim frame makes it all the way around the ring:
      - Your bid was the lowest
      - Everyone knows TTRT
      - You insert new token