The link layer – Broadcast

To do …

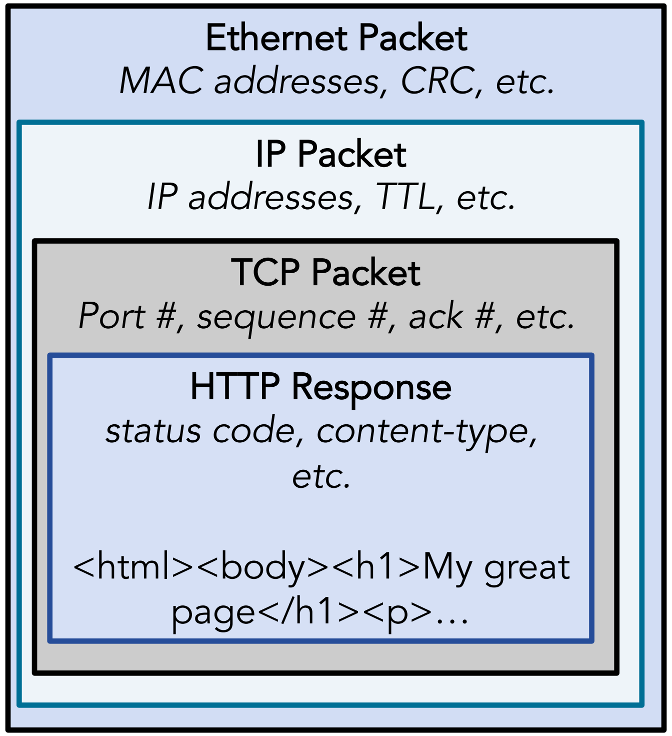
- Link layer service
- Error detection and correction
- Multiple access protocols and examples

Slides partially based on J. Kurose and K. Ross, "Computer Networking: A Top Down Approach", 7Ed., 2016

Northwestern

Each layer solves a particular set of problems

- **Link layer (layer 2) – Shares a physical channel among several transmitters/receivers**
- Network layer (layer 3)
  – Routing from source to destination, along many hops
- Transport layer:
  – Multiplexing (>1 connection / machine)
  – Ordering, Acknowledgement, Pacing
- HTTP layer:
  – Resource urls, Response codes,
  – Caching, content-types, …

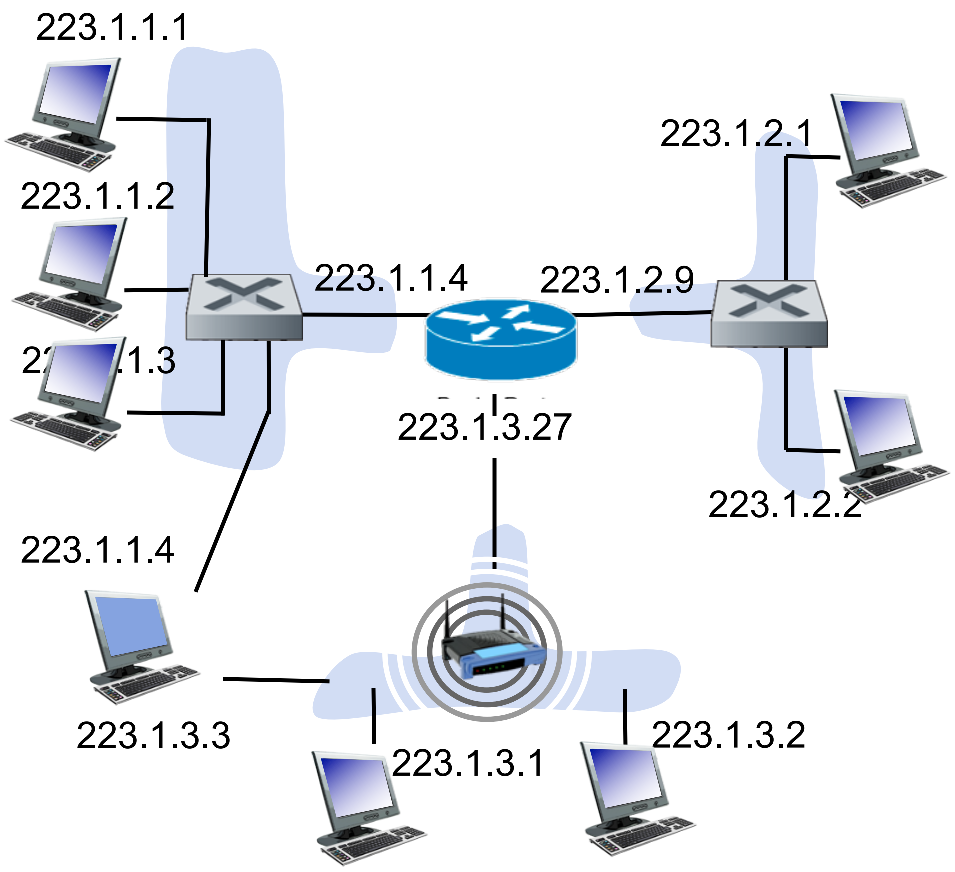


What services does it provide?

- Framing – encapsulate each network-layer datagram within a frame before transmission
- Link access – A medium access control (MAC) protocol specifies the rules by which a frame is transmitted onto the link
- Reliable delivery – Reliable across a link; as with transport, you can build it based on ack and retransmission; may make sense for error-prone links but be overhead for others
- Error detection and correction – Bit errors due to things like electromagnetic noise; can the bit errors be detected or corrected?

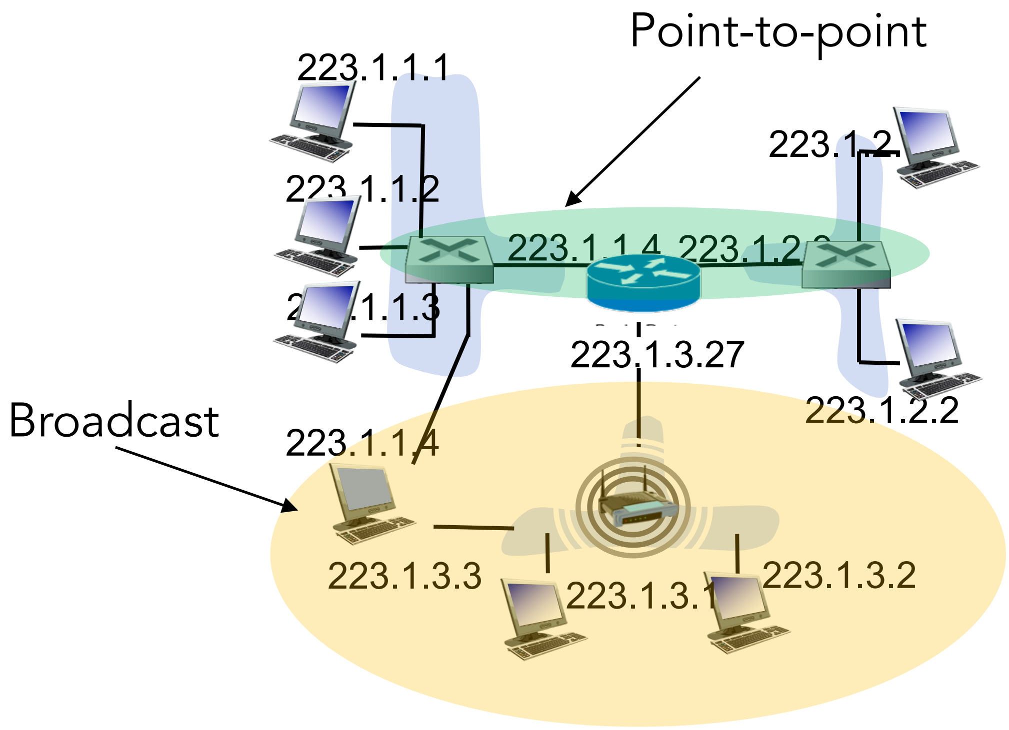
Terminology

- Nodes are hosts and routers, switches, WiFi access points
- Links are communication channels that connect adjacent nodes along communication path
  - wired links
  - wireless links
  - LANs
- On a given link, a transmission nodes encapsulates a datagram in a link-layer frame



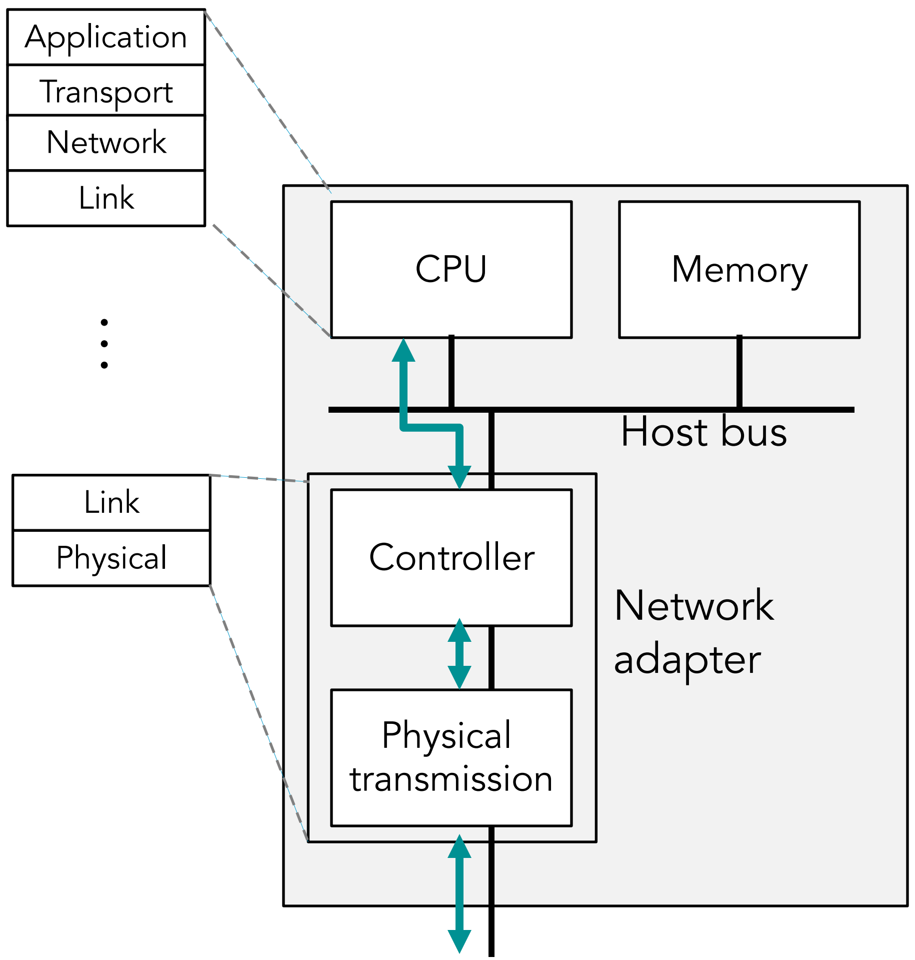
Types of link-layer channels

- There are two fundamental type of link-layer channels
  1. Broadcast, connecting multiple hosts in wireless LAN, satellite networks, hybrid fiber-coaxial cables, … many hosts connected to the same channel, so we need “multiple access protocols”
  2. Point-to-point communication, e.g., between routers or between a host and a nearby Ethernet switch



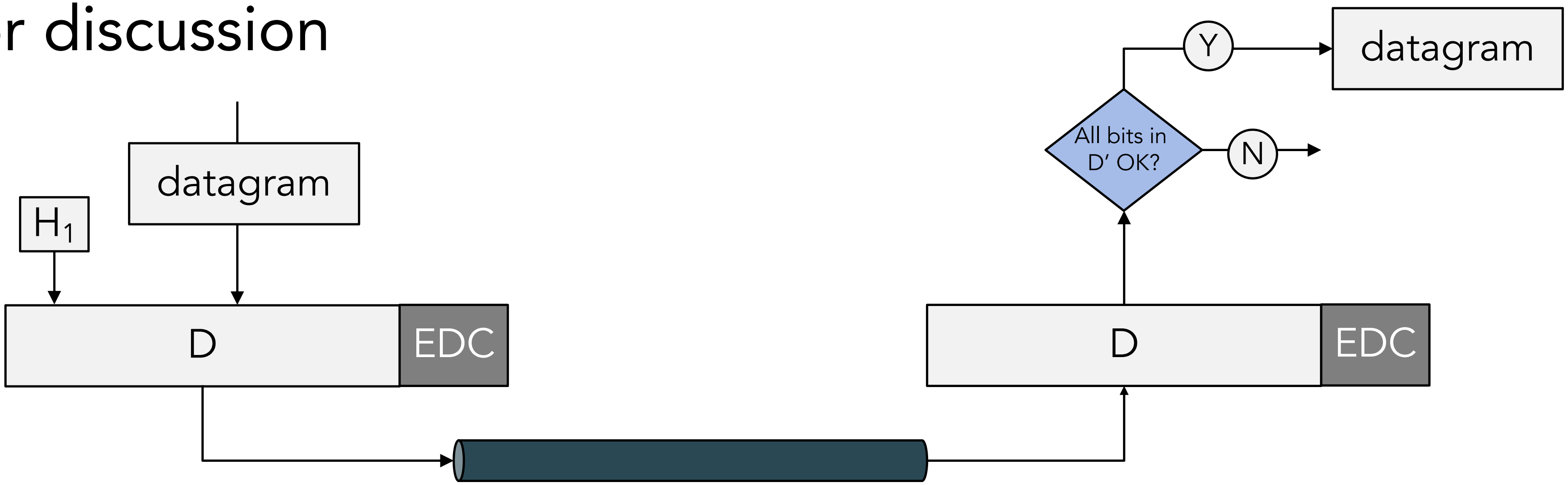
Where is it implemented? Hardware or software?

- On routers' line cards
- In hosts, software or hardware?
  – Mostly in the network adapter (or NIC), HW – a single, special-purpose chip implements many of the functions – framing, line access, error detection
  – Part in SW – assembling link-layer addressing information and activating the controller HW, responding to controller interrupts, handling error conditions, passing datagram up the stack …



Error detection and correction

- Bit-level error detection and correction is a key service
  - Wireless media are especially prone to bit-flip errors, due to noise
- Error detection and correction
  - Notice a bit error and discard the packet or fix it before delivering
    - Detect and correct – FEC or Forward Correction Error
  - In both cases, additional bits of redundant data are added
- Model for discussion



Parity

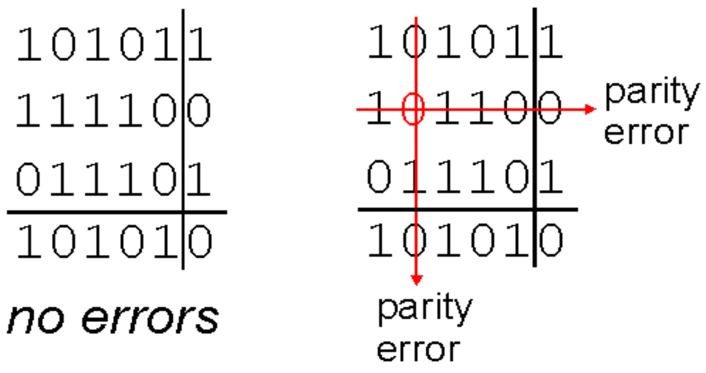
- Add a one or zero to make the total number of ones even (even parity scheme)

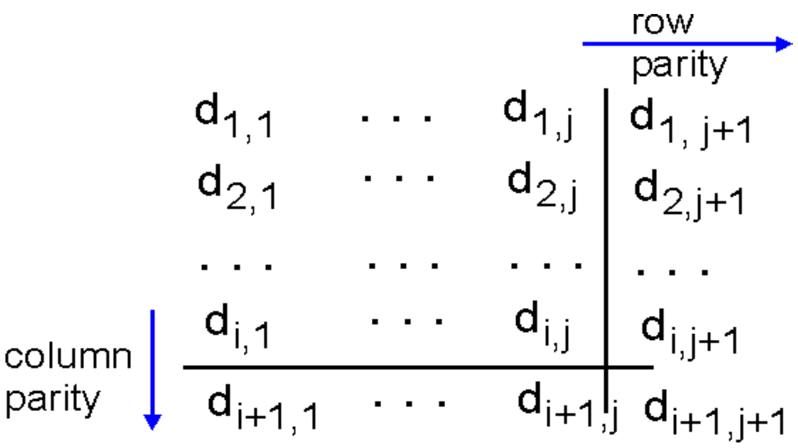
  ← d data bits → Parity bit

  10101 11110 01110 |0

  - Single-bit parity detects a single bit error: **10101 11110 01110 |0**

- Two-dimensional bit parity
  - Can detect and correct combination of two errors





Checksum and Cycle Redundancy Check

- Checksum used in IPv4, UDP, and TCP headers
  - Break the data into 16b sequences, add them up
  - Wrap the carry-out bits to the least-significant position
  - Complement
  - *Simple and fast to run in software (transport, network-layer)*

- Cycle Redundancy Check (CRC)
  - For *d* data bits (*D*) to be sent, sender choose *r* bits so that the *d* + *r* bit patter is exactly divisible by *G*, an agreed upon generator
  - Can be efficiently implemented
  - Ethernet header uses CRC-32, using a specific 32-bit generator

Medium Access Control (MAC)

- MAC is needed in broadcast links, where more than one node is sharing a single channel/medium
  - Clearly not for point-to-point links
- Collision is the fundamental problem with broadcast
  - *Think of a (at least my) family reunion!*
  - Multiple nodes try to communicate at once, interfering with each other
  - None of the colliding msgs can be received and must be retransmitted
  - Wasting time and bandwidth
- Can't be too aggressive (collision) or to polite (wasted time/bandwidth)


shared wire (e.g., cabled Ethernet)

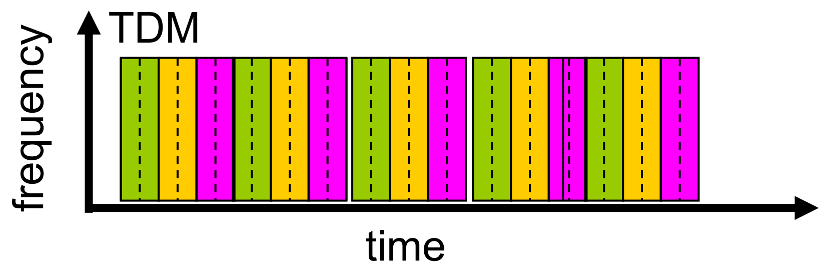

shared RF (e.g., 802.11 WiFi)

Multiple Access Protocol goals

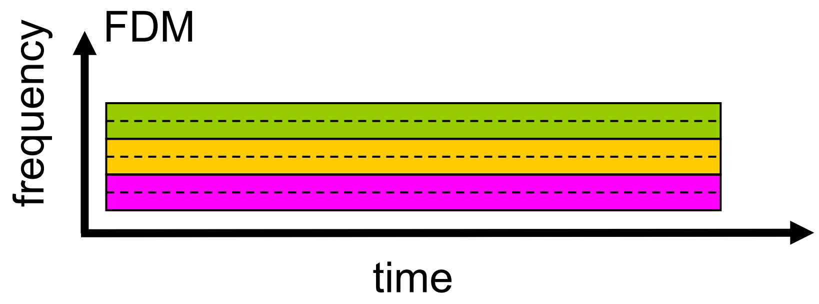
- With multiple nodes sharing a link of throughput R bps
  - Only one node communicating, should get the full bandwidth (R)
  - When N nodes have data to send, they should each get R/N bandwidth
- Protocol should be
  - Decentralized, with no single point of failure
  - It should be simple and inexpensive to implement
- Three basic classes of multiple access protocols
  - Channel partitioning
  - Random access
  - Taking turns

Channel partitioning protocols

- Time-division multiplexing
  - Eliminates conditions and is perfectly fair
  - But even if alone, a node can't get more than its R/N bps share and must wait for its turn



- Frequency-division multiplexing
  - Similarly, avoids collision and divides bandwidth fairly
  - But a node is limited to R/N even if alone



- Code division multiple access (CDMA)
  - Every node is assigned a code used to modulate the base signal (almost as assigning human languages to people in a room, signal for some, noise for the rest) – *More on this soon*

Random access protocols

- Any node can try sending immediately at full bitrate
- If there's a collision, retransmit **but** after a random delay (chosen independently)

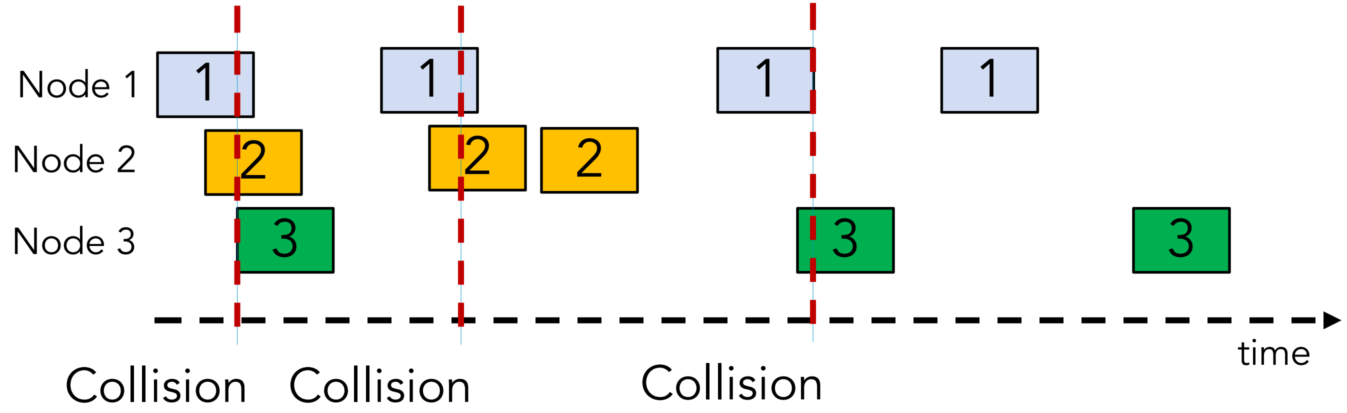
  Node 1: send *(collision)* ....send *(success)*

  Node 2: send *(collision)* .... send *(success)*

- Randomization will likely cause the two nodes to retry at different moments in the future
- Many examples, we will look at a couple
  - ALOHA
  - Carrier-Sense Multiple Access (CSMA) – Ethernet

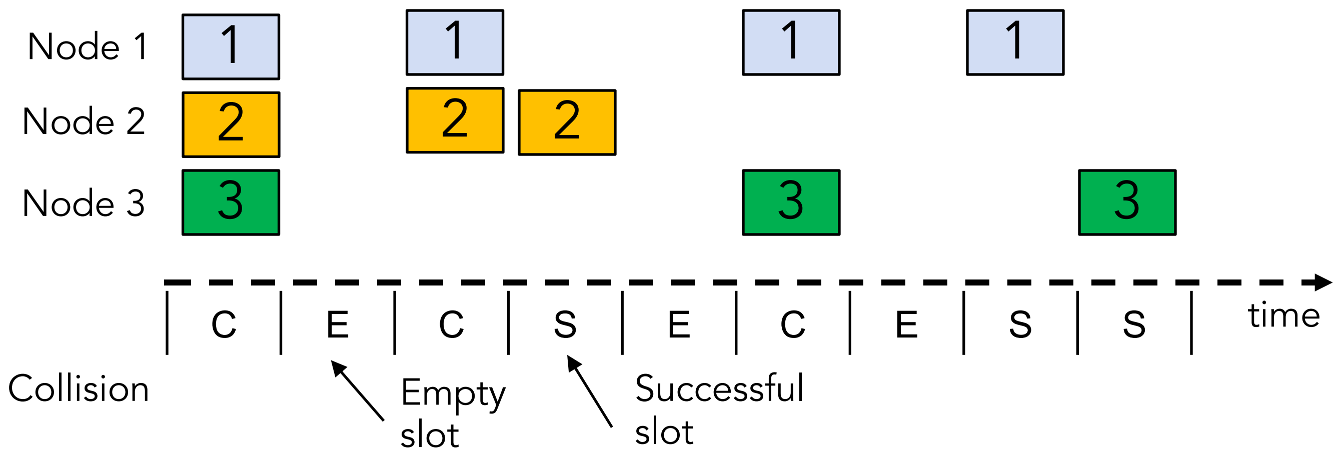
ALOHA: an early random access protocol

- 1970s for radio communication between Hawaiian islands
  - Why it matters? First example of radio packet network and the basis for Metcalfe's Ethernet
- ALOHA basics
  - (Re)Transmit the packet with probability *p* or wait for a time slot
  - A time slot = time to transmit a packet
  - No collision detection; no ACK implies collision probably occurred
  - Don't listen before broadcasting – just assume channel is free



Efficiency and a small improvement – Slotted ALOHA

- Basic ALOHA
  - If the channel is busy, we expect 18% of the peak throughput
  - Collision with other finishing/starting transmissions
- Slotted ALOHA requires time synchronization among senders
  - This lets it assign "virtual" slots of size *L/R* seconds with every segment being of *L* bits and the capacity of the link being *R bps*
  - Achieves 37% of the peak throughput



ALOHA

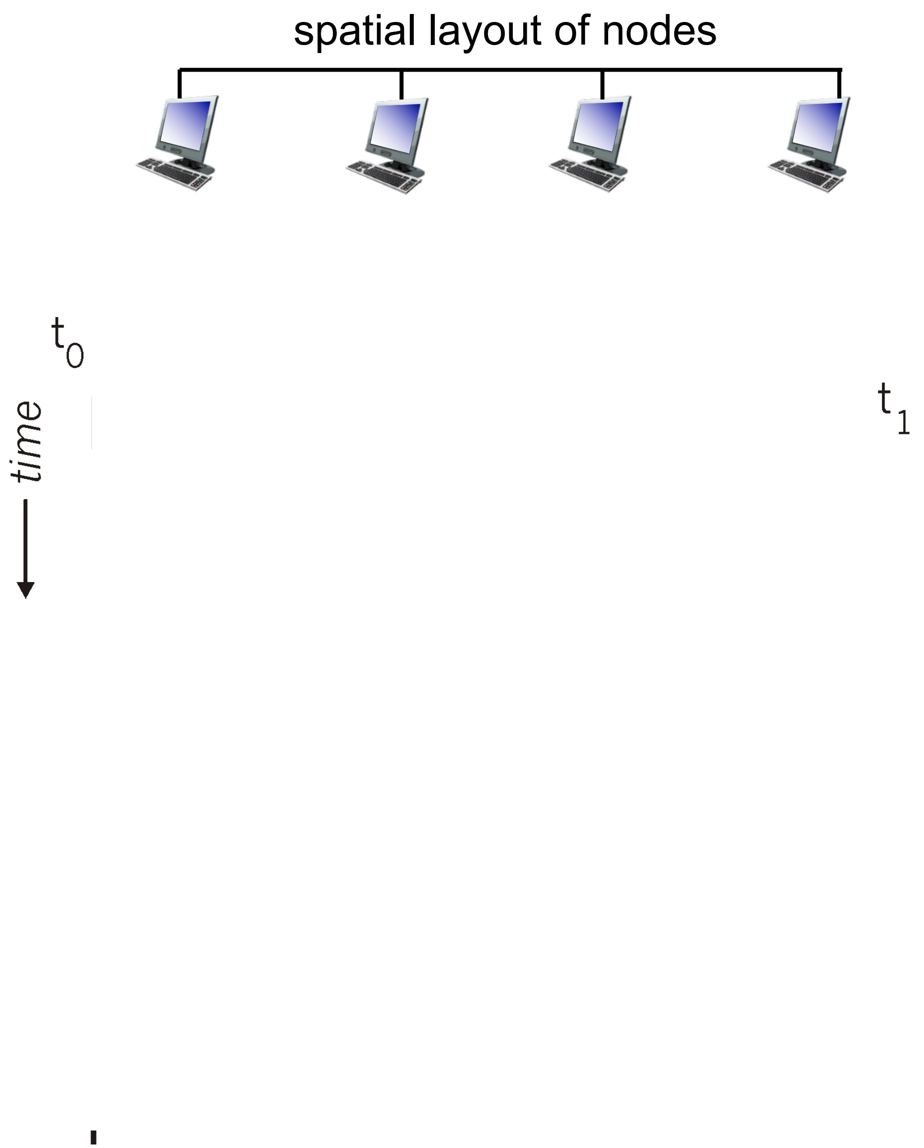
- Pros
  - One sender can use full bitrate, unlike channel partitioning
  - Decentralized, each node detects collision and independently decides when to retransmit
  - Simple
- Cons
  - With multiple nodes, collisions are possible
  - Link may be idle while waiting
  - May interrupt another sender simply because didn't listen first
  - So, poor throughput when busy

Carrier Sense Multiple Access (CSMA)

- In ALOHA, a node’s decision to transmit is independent of what other nodes are doing
  - Doesn’t matter if another one is transmitting
  - Doesn’t stop if another one starts to interfere
- Humans (*typically*) do better
  - Listen before speaking – or carrier sensing in networking
  - If someone else begins talking at the same time, stop talking – collision detection
- Rules are embodied in a family of protocols CSMA and CSMA/CD (with Collision Detection)

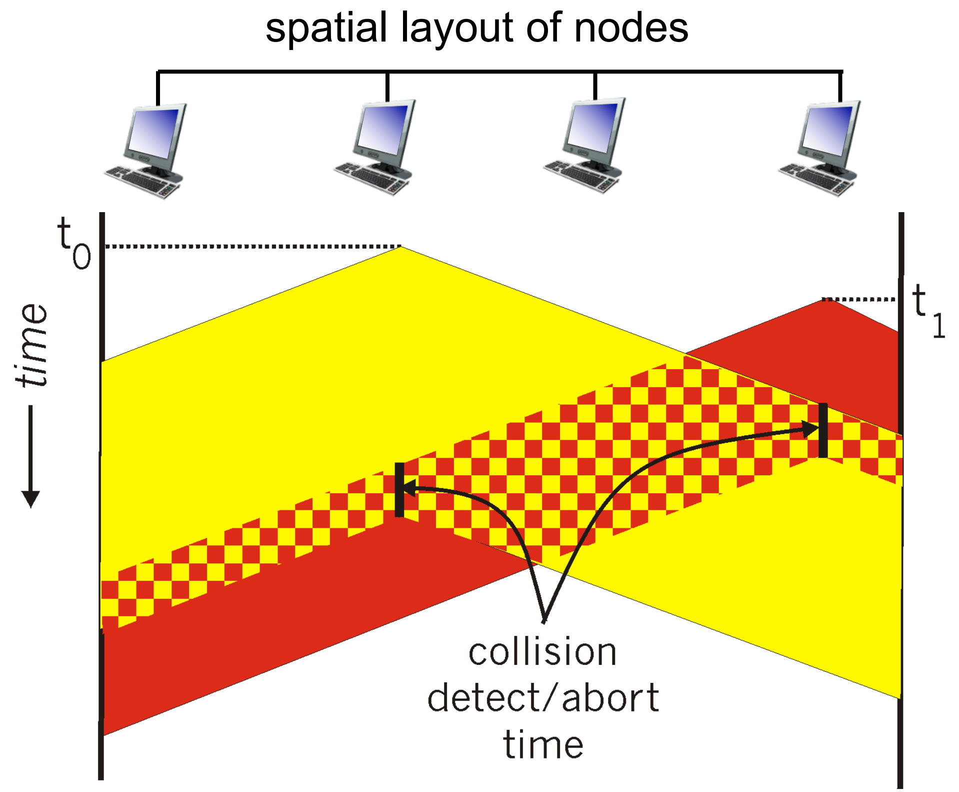
Collisions are unavoidable

- Why *carrier sensing* is not enough to prevent collisions?
- Propagation delay in the channel delays carrier sensing observations
  - A node's knowledge of channel state is always slightly out-of-date
- The longer the propagation delay, the higher collision rate



Collision detection

- Reduces the channel-time wasted by collisions
- How?
  - Measure channel signal
  - If energy is greater than transmission energy, there must be some extra signal
- Works well for wired channels
- For wireless, received radio signals are much weaker than transmitted signals, so collision detection is harder



CSMA/CD steps

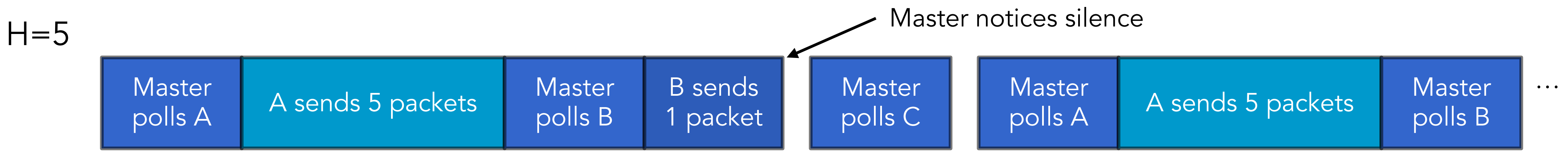
- Carrier Sensing: listen to the channel before sending,
  - If it's busy, then wait
- Collision Detection: listen while transmitting,
  - Abort transmission immediately if another transmission is heard, try after waiting for a random interval of time … *but how long?*
- Binary exponential backoff determines that
  - If frame has collided *n* times, choose a random value *K* in $[0, (2^n-1)]$, and waits for *K* * the time needed to send a frame (for Ethernet, *K**512/bitrate, and *n* cannot grow past 10)
  - Exponential backoff resets for each new packet

Binary exponential backoff example

- A node attempts to transmit a frame for the fist time and detects collision
- Chooses between K = 0 or 1 with equal probability
  - K = 0, begin sensing immediately
  - K = 1, wait for time to send 512 bits, 512 μsec on a 100Mbps Ethernet
- After a second collision, *K* is chosen from {0,1,2,3}, after three collisions from {0,1,2,3,4,5,6,7} …
- *New frame for transmission?* Start from 0

Taking-turns protocols

- Getting a fair share
  - Random access protocols do not guarantee a fair share of bandwidth
  - Taking-turns protocols assign slots but the reservations are dynamic, not pre-scheduled
- Bluetooth, a taking-turns protocol called a Polling Protocol
  - A master node polls others, round-robin, sending a msg telling each to send up to H packets
  - If master sees a node stopped sending packets early, it polls next one
  - Polling msgs add some coordination overhead and the master can die



Token-passing protocols

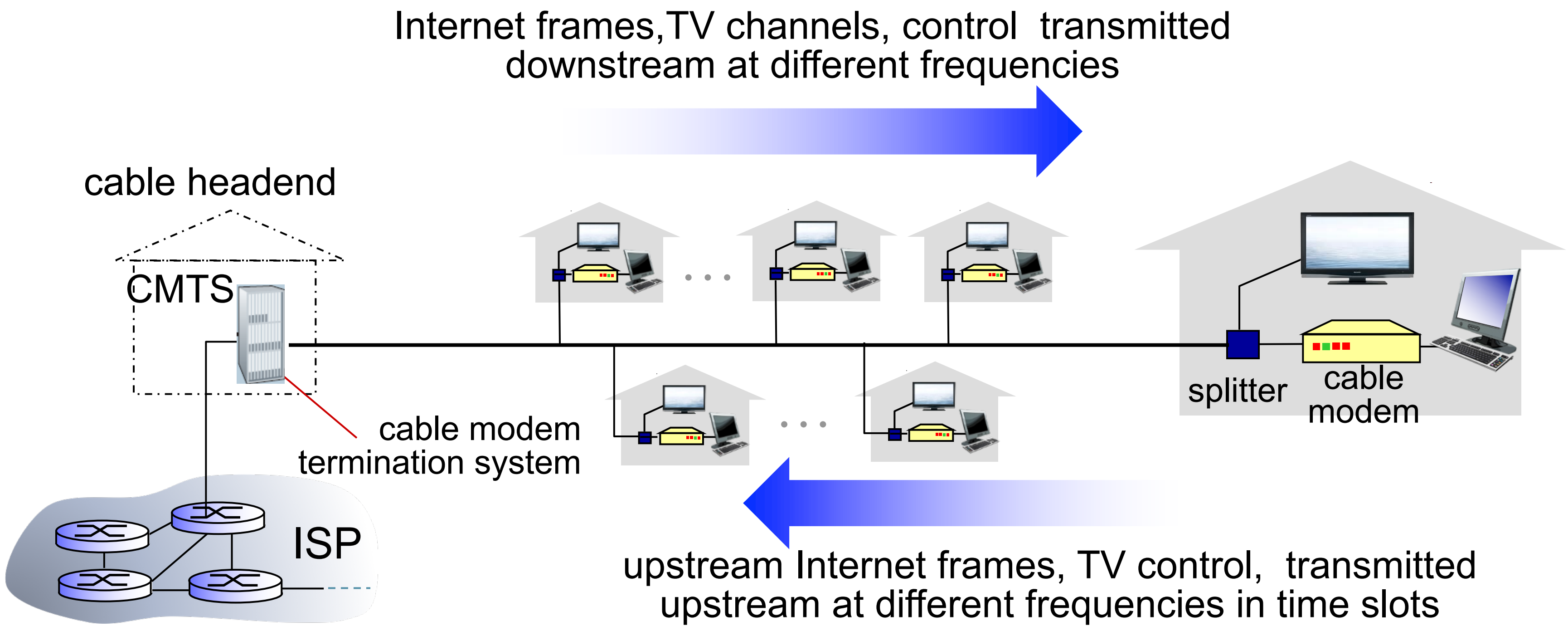
- Similar to polling protocol, but without the special master node
- A token (a special frame) is exchanged among the nodes in some fixed order
  - Node holding the token can transmit up to some maximum number of frames, then must pass the token on
- No collision – Nodes only sends while "holding" the token
- But lost token problem – If token-holding node crashes, the entire network crashes
- FDDI (fiber distributed data interface) and IEEEE 802.5 are token-passing protocols

Multiple access protocol summary

	Channel Partitioning		Random Access		Taking-Turns	
	FDM	**TDM**	**ALOHA**	**CSMA/CD**	**Polling**	**Token-Passing**
Single Sender throughput	R/N	R/N	R	R	R – C*N	R – C*N
Busy throughput	R	R	~37% R *(slotted)* or ~18% R	$\frac{R}{1+5d_{prop}/d_{trans}}$	R – C*N	R – C*N
Collisions			yes	unlikely		
Centralized	yes	yes			yes	
Crash-sensitive					yes	yes
Requires time synchronization		yes	optional			
Requires carrier sensing				yes	yes	

DOCSIS – Link-layer protocol for cable Internet

- Combines ideas from all classes of multiple access protocols
- All modems in a neighborhood share the same coaxial medium
  - They are all connected to "one big wire"
  - They can all "hear" each other's traffic, in both directions



DOCSIS – Link-layer protocol for cable Internet

- FDM (channel partitioning) to divide upstream and downstream segments into multiple frequency channels
  - Each is a broadcast channel
  - No multiple access problem downstream (only the CMTS)
- Upstream channels also use TDM
  - Some time slots are for modems to send time requests
  - Time requests use a random-access protocol and may collide
    - How do you know? No allocated slot next time, exponential backoff to re-request
  - Remaining time slots are assigned to specific modems (taking turns)
- CMTS periodically broadcasts the time slot assignments, taking into account the time requests that were received

Recap

- Link-layer handles sharing a link/medium with multiple nodes
  - And handles error detection and correction: Parity, Checksum, and CRC.
- Medium Access Control / Multiple Access Protocol
  - Decide how to share the link
  - Two nodes sending simultaneously is a collision., packets are lost
- Three classes of sharing protocols
  - Channel Partitioning, random access and turn-taking
- And one example in DOCSIS