Telematics I

Chapter 5
Medium Access Control

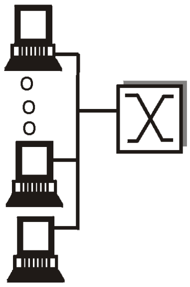
Goals of this Chapter

- Learn how to share one medium among multiple entities
- Understand performance problems of fixed multiplexing schemes
- Important performance metrics
- Options for MAC protocols: sending, receiving, listening, synchronizing; environment in which they work
- Classification & examples of MAC protocols, performance aspects
  - An important example: Ethernet

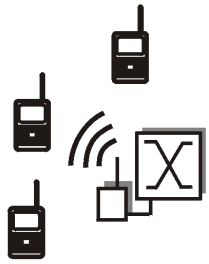
Intro: Multiple Access Links and Protocols

Two types of "links":

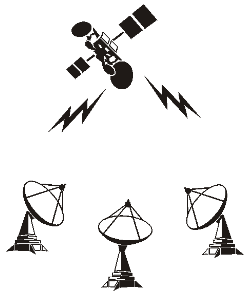
- Point-to-point
  - PPP for dial-up access
  - Point-to-point link between Ethernet switch and host
- **Broadcast** (shared wire or medium)
  - Traditional Ethernet
  - Upstream HFC
  - 802.11 wireless LAN



shared wire (e.g. Ethernet)



shared wireless (e.g. Wavelan)



satellite

cocktail party

Intro: Multiple Access Protocols

- Single shared broadcast channel
- Two or more simultaneous transmissions by nodes: interference
  - **Collision** if node receives two or more signals at the same time

Multiple access protocol

- Distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- Often: Communication about channel sharing must use channel itself!
  - No out-of-band channel for coordination

Intro: Ideal Multiple Access Protocol

Broadcast channel of rate R bps

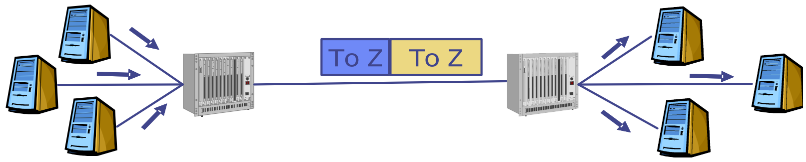
1. When one node wants to transmit, it can send at rate R.
2. When M nodes want to transmit, each can send at average rate R/M
3. Fully decentralized:
   - ❑ No special node to coordinate transmissions
   - ❑ No synchronization of clocks, slots
4. Simple

Outline

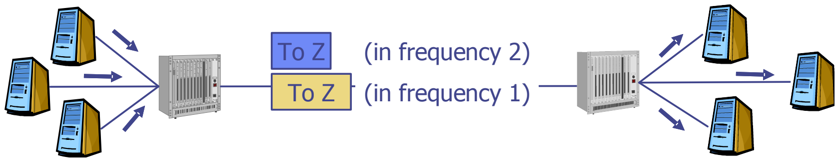
- ❑ ***Static multiplexing (revisited)***
- ❑ Dynamic channel allocation
- ❑ Collision-based protocols
- ❑ Contention-free protocols
- ❑ Limited contention protocols
- ❑ Case study: Ethernet

Static Multiplexing

- Given a single resource, it can be statically multiplexed
  - Assigning fixed time slots to multiple communication pairs



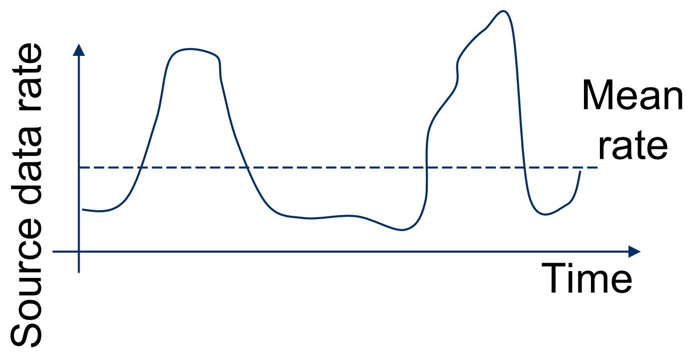
  - Assigning fixed frequency bands



- Assigning fixed resources to different sources is fine if
  - Data rate of source and multiplexed link are matched
  - Sources can always saturate the link

Bursty Traffic

- What happens if sources have ***bursty*** traffic?
  - Definition: Large difference between peak and average rate
  - In computer networks: Peak : average = 1000 : 1 quite common

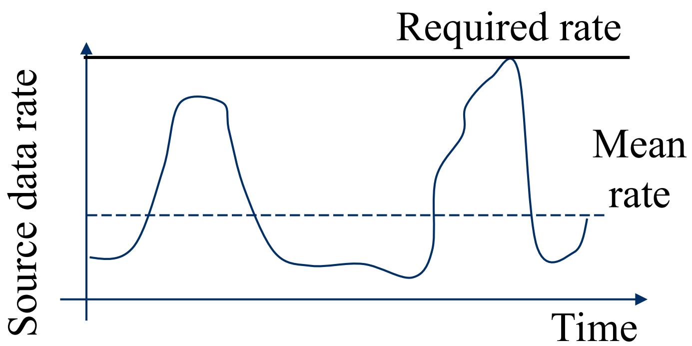


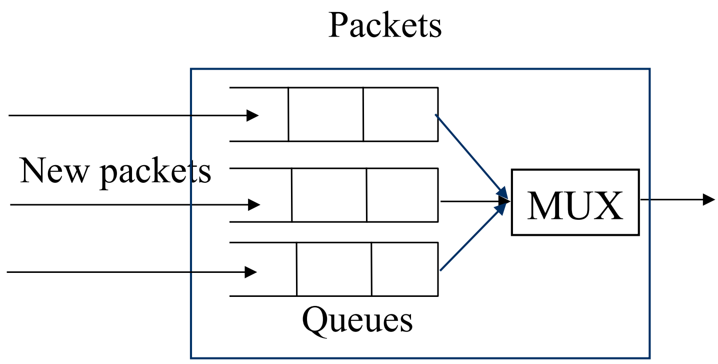
Static Multiplexing & Bursty Traffic

- Statically multiplexed resources must either:
  - Be large enough to cope with the peak data rate immediately

    ! Big waste, since *on average* the link/channel will not be utilized
  - Be dimensioned for average rate, but then need a *buffer*

    ! What is the ***delay*** until a packet is transmitted?





Statically Multiplexed Bursty Traffic – Delay

- Compare the delay resulting from static multiplexing
- Base case: No multiplexing, a single traffic source with average rate ρ (bits/s), link capacity C bits/s
  - **Delay is T**

    (In case you really want to know: $T = 1/(\mu C-\lambda)$, $\rho=\lambda/\mu$)
- Multiplexed case: Split the single source in N sources with same total rate, statically multiplex over the same link (e.g., FDM)
  - **Delay T_{FDM} = NT**
  - Irrespective of FDM, TDM, …
- Hence: multiplexing increases N-fold the delay of a packet
  - Intuition: Because some of channels are idle sometimes

Outline

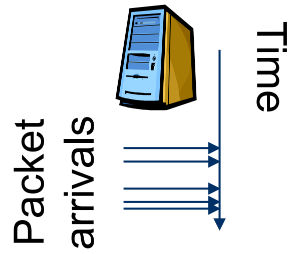
- Static multiplexing
- ***Dynamic channel allocation***
- Collision-based protocols
- Contention-free protocols
- Limited contention protocols
- Case study: Ethernet

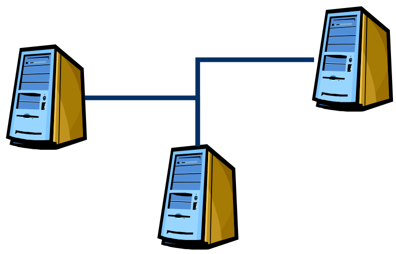
Dynamic Channel Allocation – MAC

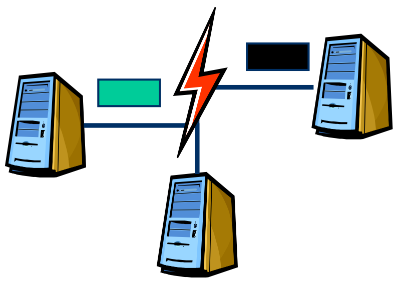
- Because of the bad delay properties – caused by idle sub-channels – static multiplexing is not appropriate for bursty traffic sources
  - Telephony is not bursty, computer networks are bursty
- Alternative: Assign channel/link/resource to that source that **currently** has data to send
  - ***Dynamic channel allocation***
  - Instead of fixed assignments of parts of a shared resource
- Terminology: Access to the transmission has to be organized – a ***medium access control protocol (MAC)*** is required

Assumptions for Dynamic Channel Allocation (1)

- **Station model** (or terminal model)
  - N independent stations want to share a given resource
  - One possible load model: probability of generating a packet in interval Δt is $\lambda \Delta t$, λ = const
- **Single channel assumption**
  - Only a single channel for all stations
  - No possibility to communicate/signal anything via other means
- **Collision assumption**
  - Only a single frame can be successfully transmitted at a time
  - Two (or more) frames overlapping in time will ***collide*** and are both destroyed
  - No station can receive either frame
  - Note: there are sometimes exceptions to this rule

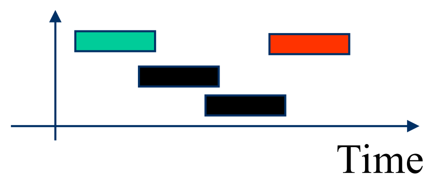


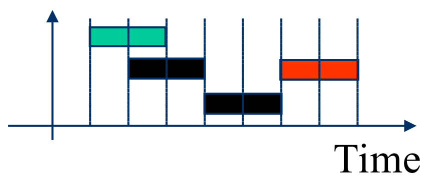


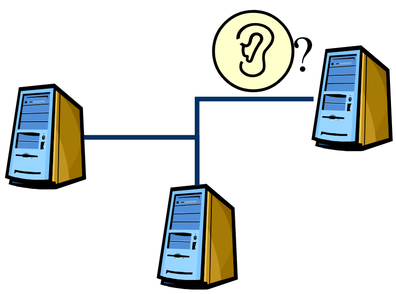


Assumptions for Dynamic Channel Allocation (2)

- **Time model**
  - Continuous time: Transmissions can begin at any time; no central clock
  - Slotted time: Time is divided in slots; transmissions can only start at a slot boundary. Slot can be idle, a successful transmission, or a collision
- **Carrier Sensing**
  - Stations can/cannot detect whether the channel is currently used by some other station
  - There might be imperfections involved in this detection (e.g., incorrectly missing an ongoing transmission)
  - Usually, a station will not transmit when the channel is sensed as busy







Figures of Merit

- How to judge the efficiency of a dynamic channel allocation system?
  - Intuition: transmit as many packets as quickly as possible

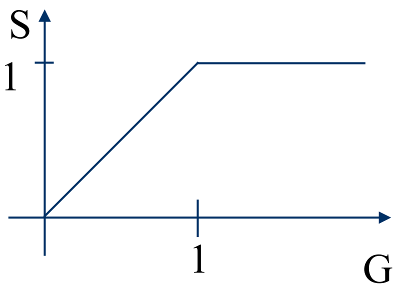
- At high load (many transmission attempts per unit time): ***Throughput*** is crucial – ensure that many packets get through

- At low load (few attempts per time): ***Delay*** is crucial – ensure that a packet does not have to wait for a long time

- ***Fairness:*** Is every station treated equally?

Throughput and Offered Load

- ***Offered load G***: The number of packets per unit packet time that the protocol is asked to handle
  - More than one packet per packet time equals overload
- Ideal protocol:
  - Throughput S equals offered load G as long as G<1
  - Throughput S = 1 as soon as G>1

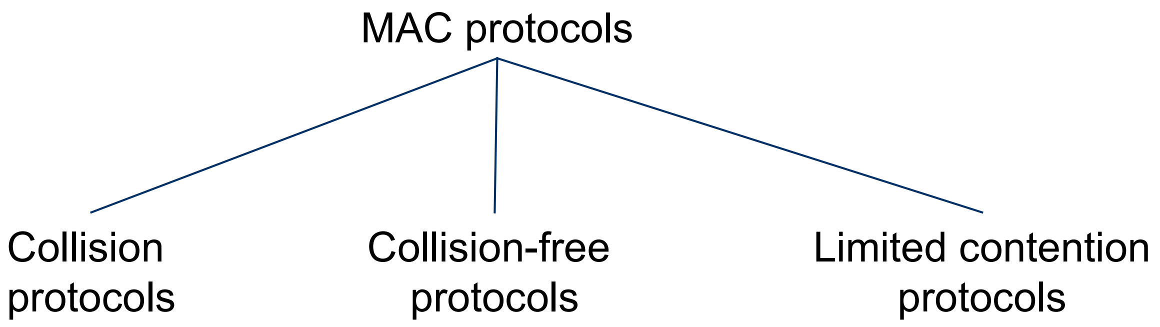


- And:
  - have constant small delay,
  - for an arbitrary number of terminals

- Not very realistic hope!

Principal Options for MAC Protocols

- Main distinction: Does the protocol allow collisions to occur?
  - As a deliberately taken risk, not as an effect of an error
  - Always, for every packet, or in some restricted form?



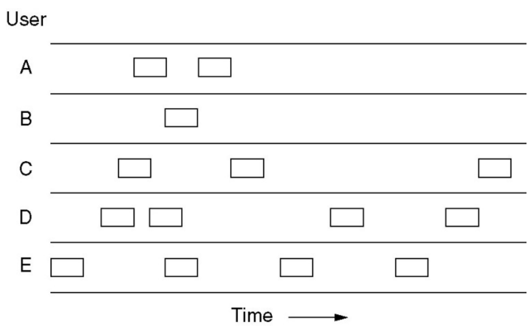
Terminology: Systems where collisions can occur are often called ***contention*** systems

Outline

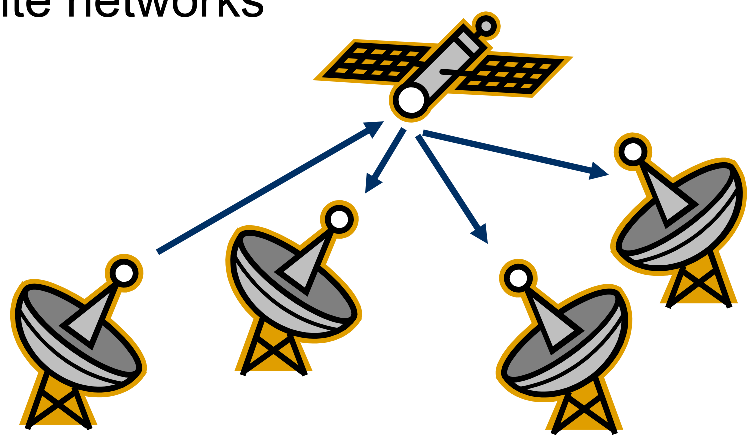
- Static multiplexing
- Dynamic channel allocation
- ***Collision-based protocols***
- Contention-free protocols
- Limited contention protocols
- Case study: Ethernet

ALOHA

- The simplest possible medium access protocol:
  Just talk when you feel like it
- Formally: Whenever a packet should be transmitted, it is transmitted immediately
- Introduced in 1970 by Abrahmson et al., University of Hawaii
- Goal: Support of satellite networks



Packets are transmitted at arbitrary times



ALOHA – Analysis

- ALOHA advantages
  - Trivially simple
  - No coordination between participants necessary
- ALOHA disadvantages
  - Collisions can and will occur – sender does not check channel state
  - Sender has no (immediate) means of learning about the success of its transmission – link layer mechanisms (ACKs) are needed
    - ACKs can collide as well ☹

ALOHA – Performance

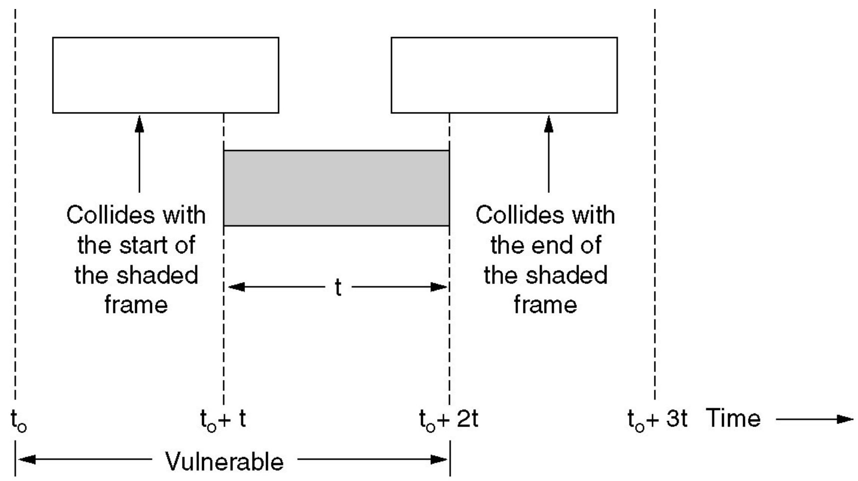
- Assume a Poisson arrival process to describe packet transmissions
  - Infinite number of stations, all behave identically, independently
  - Let G be the mean number of transmission attempts per packet length
  - All packets are of unit time length
  - Then: $P(k \text{ attempts in time } t) = \frac{(Gt)^k}{k!} e^{-Gt}$

    (Ok, this may be a bit hard to understand here, but for the moment let us just accept it.)

- For a packet transmission to be successful, it must not collide with any other packet
- How likely is such a collision?
  - Question: How long is a packet "vulnerable" by other transmissions?

ALOHA – Performance

- A packet X is destroyed by a packet either
  - Starting up to one packet time ***before*** X
  - Starting up to immediately before the end of X



- Hence: Packet is successful if there is no *additional* transmission in two packet times
  - Probability: P_0 = P (1 transmission in two packet times) = $2Ge^{-2G}$
  - Maximal throughput S (G) = 1 Packet / 2 time units * Probability = Ge^{-2G}
  - Optimal for G = 0.5 $\Rightarrow$ S = 1/(2e) $\approx$ 0.184

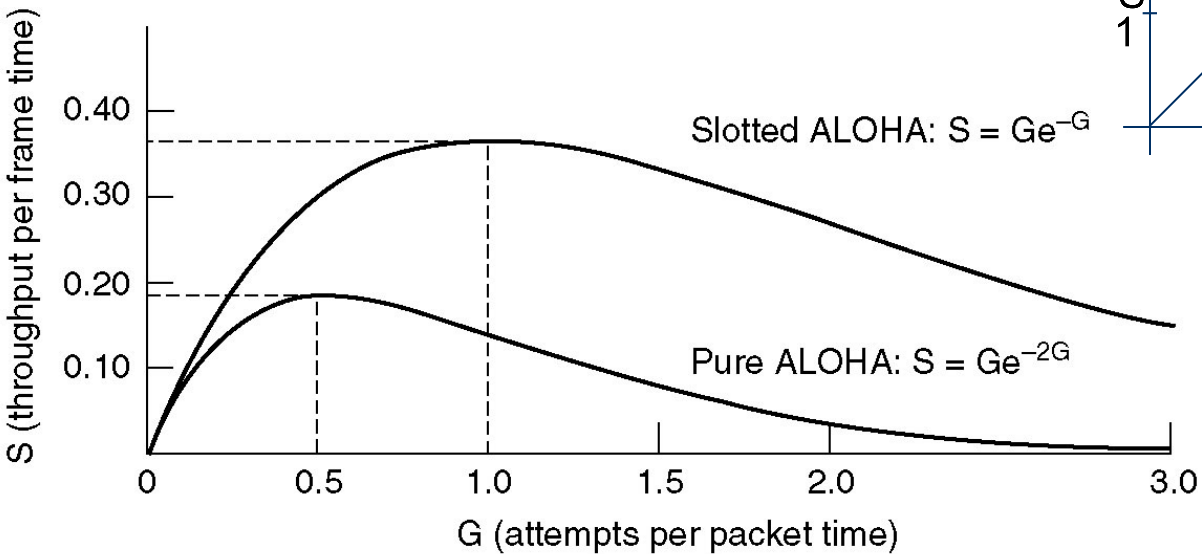
    $\Rightarrow$ Mean achievable throughput is less than 20%!

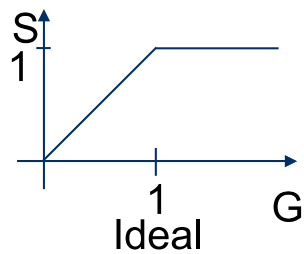
A Slight Improvement: Slotted ALOHA

- ALOHA's problem: Long vulnerability period of a packet
- Reduce it by introducing time slots – transmissions may only start at the start of a slot
  - Slot synchronization is assumed to be "somehow" available
- Result: Vulnerability period is halved, throughput is doubled
  - $S(G) = Ge^{-G}$
  - Optimal at G=1, S=1/e

Performance Dependence on Offered Load

- For (slotted) ALOHA, closed form analysis of throughput S as function of G is simple





! Anything but a high-performance protocol

- In particular: throughput collapses as load increases!

Carrier Sensing

- (Slotted) ALOHA are simple, but not satisfactory
- Be a bit more polite: ***Listen before talk***
  - Sense the carrier to check whether it is idle before transmitting
  - ***Carrier Sense Multiple Access (CSMA)***
  - Abstain from transmitting if carrier not idle (some other sender is currently transmitting)
- Crucial question: How to behave in detail when carrier is busy?
  - In particular: WHEN to retry a transmission?

1-Persistent CSMA

- When carrier is busy, wait until it is idle
- Then, immediately transmit
  - "Persistent" waiting
- Obvious problem: if more than one station wants to transmit, they are *guaranteed* to collide!
  - Just too impatient…
- But certainly better than pure ALOHA or slotted ALOHA

Non-Persistent CSMA

- When channel is idle, transmit
- When channel is busy, wait a random time before checking again whether the channel is idle
  - Do not continuously monitor to greedily grab it once it is idle
  - Conscious attempt to be less greedy

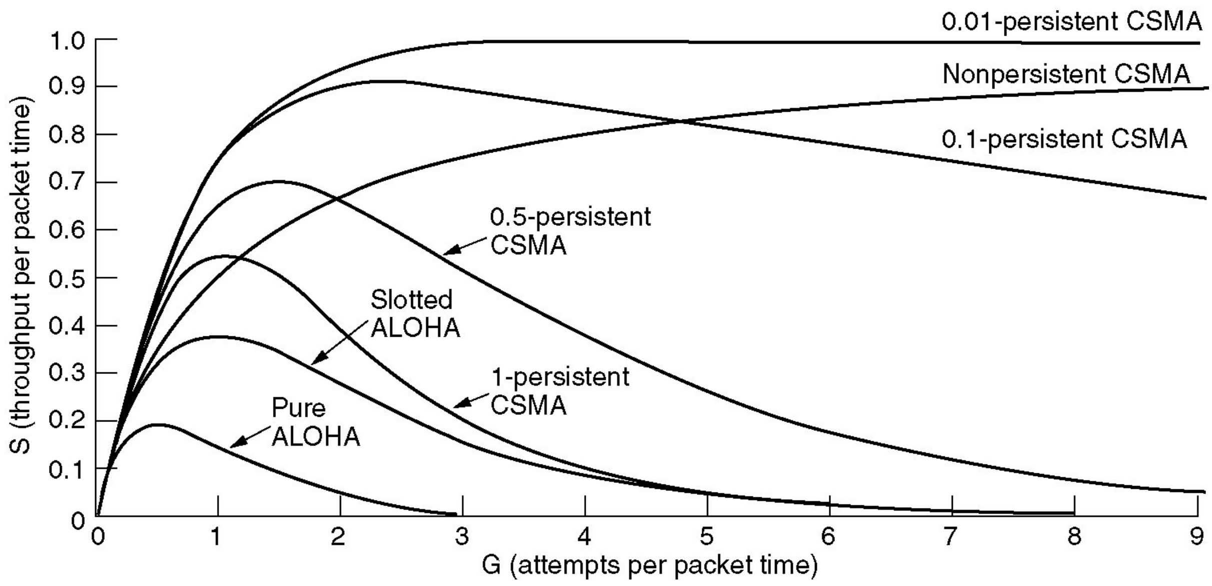
- Performance depends a bit on the random distribution used for the waiting time
  - But in general better throughput than persistent CSMA for higher loads
  - At low loads, random waiting is not necessary and wasteful

P-Persistent CSMA

- Combines ideas from persistent and non-persistent CSMA
  - Uses a slotted time model

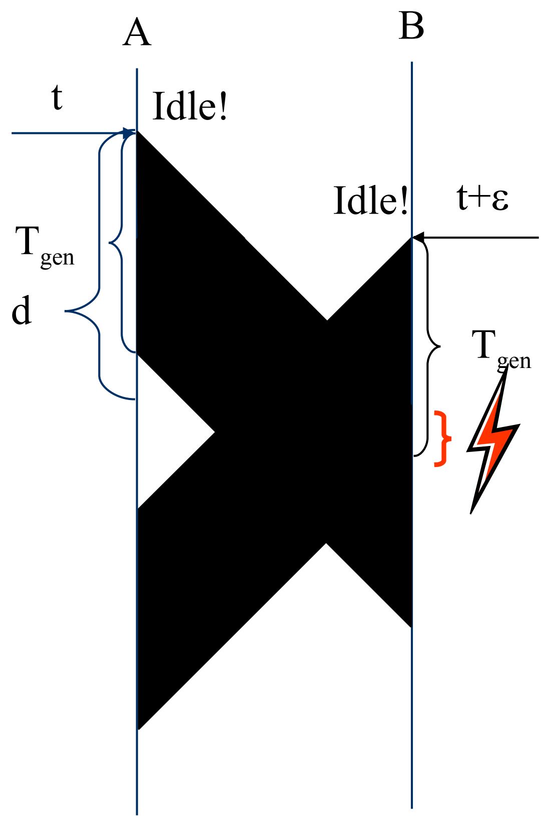
- When channel is idle, send
- When channel is busy, continuously monitor it until it becomes idle
  - But then, do not always transmit immediately
  - But flip a coin – transmit with probability p
  - With probability 1-p, do not send and wait for the next slot
    - If channel is busy in the next slot, monitor for idleness
    - Else, flip a coin again

Performance of CSMA



CSMA and Propagation Delay

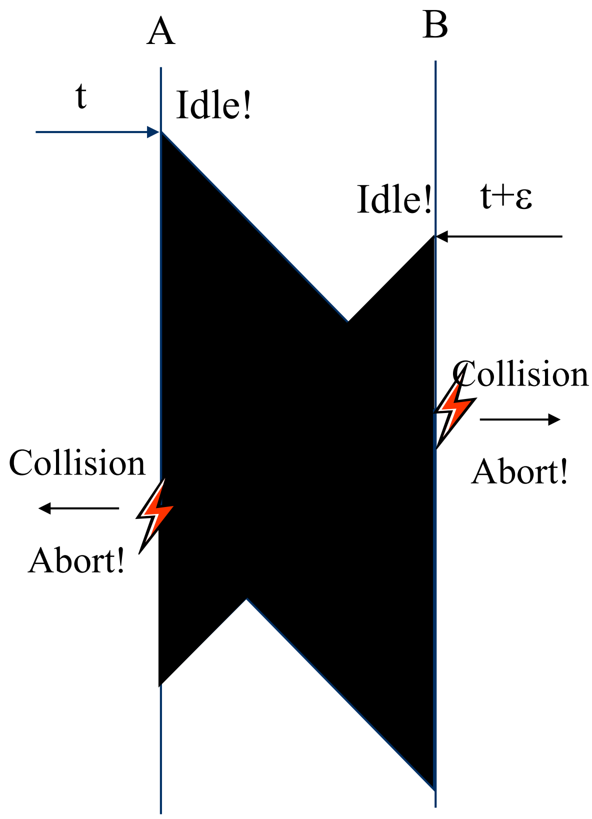
- Any CSMA scheme has a principal obstacle: The propagation delay d
- Suppose two stations become ready to send at time t and t+ε
  - At t, the channel is completely idle
  - The stations are separated by a propagation delay d > ε
- Second station cannot detect the already started transmission of first station
  - Will sense an idle channel, send, and collide (at each other, or at a third station)



Collision Detection – CSMA/CD

- When two packets collide, lots of time is wasted by completing their transmission
- If it were possible to ***detect*** a collision when it happens, transmission could be aborted and a new attempt made
  - Wasted time reduced, no need to wait for (destroyed) packets to complete
- Depending on physical layer, collisions ***can*** be detected!
  - Necessary: Sender must be able to listen to the medium when sending, compare what it sends with what it receives
  - If different: declare a collision

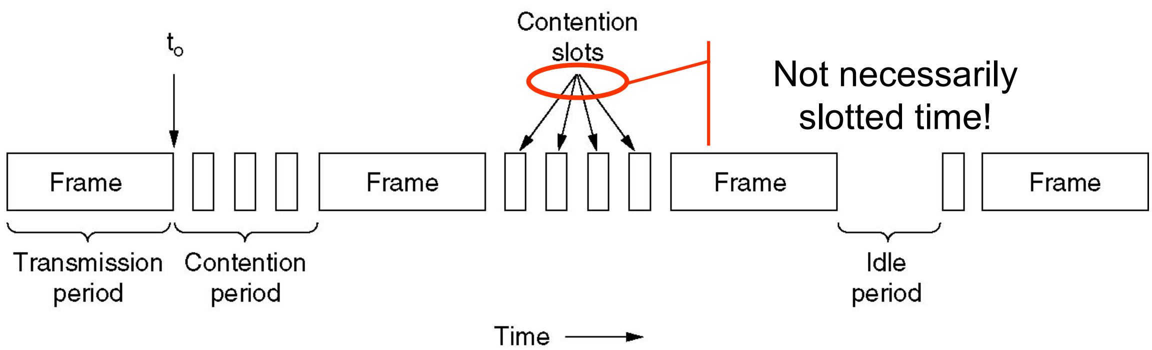
! ***CSMA/CD – Carrier Sense Multiple Access/Collision Detection***



What to Do After a Collision Happens?

- Stations do want to transmit their packets, despite detecting a collision
- Have to try again
  - Immediately? Would again ensure another collision ☹
  - Coordinate somehow? Difficult, no communication medium available
  - Wait a random time!
    - Randomization “de-synchronizes” medium access, avoids collisions
    - However: will result in some idle time, occasionally

! Alternation between contention and transmission phases



How to Choose Random Waiting Time?

- Simplest approach to choose a random waiting time: Pick any one of *k* slots
  - Assumes a slotted time model for simplicity
  - Uniformly distributed from [0,…, k-1] – the ***contention window***
- Question: How to choose upper bound *k*?
  - Small *k*: Short delay, but high risk of repeated collisions
  - Large *k*: Low risk of collisions (as stations' access attempts are spread over a large time interval), but needlessly high delay if few stations want to access the channel
    - With large contention window, collisions become less likely

! Let *k* ***adapt*** to the current number of stations/traffic load

How to Adapt *k* to Traffic Load?

- One option: somehow *explicitly* find out number of stations, compute an optimal *k*, signal that to all stations
  - Difficult, high overhead, …
  - An *implicit* approach possible?
- What is the consequence of a small *k* when load is high?
  - Collisions!
  - Hence: Use a collision as an indication that the contention window is too small – increase it!
    - Will reduce probability of collisions, automatically adapt to higher load
- Question: How to increase *k* after collision, how to decrease it again?

How to Adapt *k* – Binary Exponential Backoff

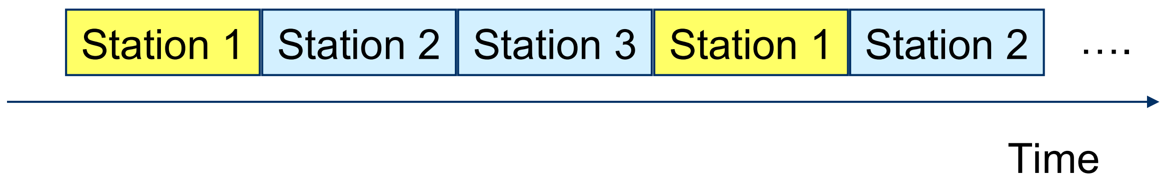
- Increase after collisions: Many possibilities
  - Commonly used: **Double** the contention window size *k*
  - But only up to a certain limit, say, 1024 slots – start out with *k* = 2
  - This is called ***binary exponential backoff***
- Decreasing *k*: Also many options possible
  - E.g., if sufficiently many frames have not collided reduce *k* (subtract a constant, cut in half, …)
    - Complicated, might waste resources by not being agile enough, …
  - Or play it simple: Just start *every time* at k = 1!
    - Common option

Outline

- Static multiplexing
- Dynamic channel allocation
- Collision-based protocols
- ***Contention-free protocols***
- Limited contention protocols
- Case study: Ethernet

Contention-Free Protocols

- Since collisions cause problems, how about using protocols without contention for the medium?
- Simplest example: Static TDMA
  - Each station/terminal is assigned a fixed time slot in a periodic schedule



- But disadvantages of static multiplexing are clear
- Are there *dynamic, contention-free* protocols?

Bit-Map Protocol

- Problem of static TDMA: When a station has nothing to send, its time slot is idling and wastes resources
- Possible to only have time slots assigned to stations that have data to transmit?
  - Needs some information exchange *which* station is ready to send
  - They should *reserve* resources/time slots
- Some approaches:
  - Central master assigns right to talk (like in classroom; polling)
  - Token Passing
  - Stations announce will to send (bit-map protocol)

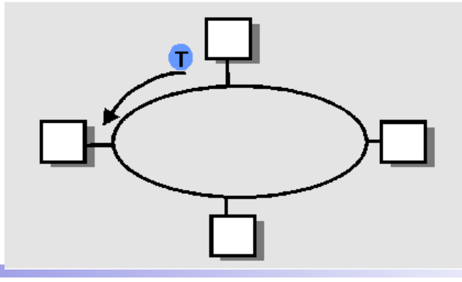
Polling & Token Passing

Polling:

- Master node "invites" slave nodes to transmit in turn
- Concerns:
  - Polling overhead
  - Latency
  - Single point of failure (master)

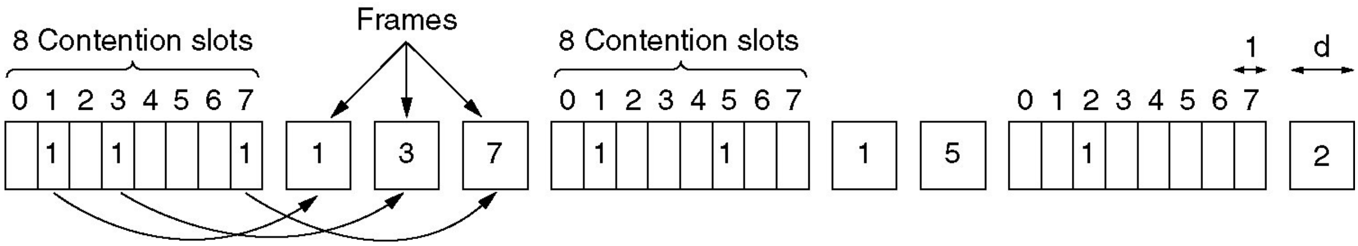
Token passing:

- Control **token** passed from one node to next sequentially
- Can be realized on either topological or logical ring structure
- Token message
- Concerns:
  - Token overhead
  - Latency
  - Token might get lost



Bit-Map Protocol

- Stations announce their will to transmit
  ! Bit-map protocol
  - Short reservation slots, only used to announce desire to transmit
  - Must be received by every station
  - All stations need to know when slots start
    ⇒ requires tight time synchronization!



Bit-Map Protocol – Properties

- Behavior at low load
  - If there is (hardly) any packet, the medium will repeat the (empty) contention slots
  - A station that wants to transmit has to wait its turn before it can do so
  - ! Delay
- Behavior at high load
  - At high load, medium is dominated by data packets (which are long compared to contention slots)
  - Overhead is negligible
  - ! Good and stable throughput
- Note: Bit-map *is* a carrier-sense protocol!

Outline

- Static multiplexing
- Dynamic channel allocation
- Collision-based protocols
- Contention-free protocols
- ***Limited contention protocols***
- Case study: Ethernet

Best of Both Worlds?

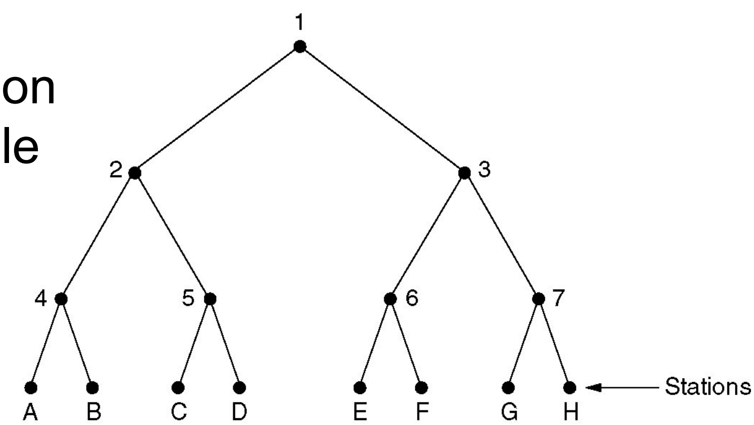
- Desirable: Protocol with
  - Low delay at low load – like a contention protocol
  - High throughput at high load – like a contention-free protocol
- Hybrid or ***adaptive*** solution?

! ***Limited-contention protocols*** do exist

- One possible idea: adapt number of stations per contention slot
  - Contention slots are nice for throughput, but at low load, we cannot afford to wait a long time for every station's slot
  - Several stations have to share a slot, dynamically

Adaptive Tree Walk

- Idea: Use several "levels of resolution" for the contention slots
  - Inspired by levels in a tree
  - At highest level, all nodes share a single slot
  - If only node from this group claims the contention slot, it may transmit
  - If more than one ⇒ collision in contention slot ⇒ double slots, half the stations assigned to each slot
  - And recurse



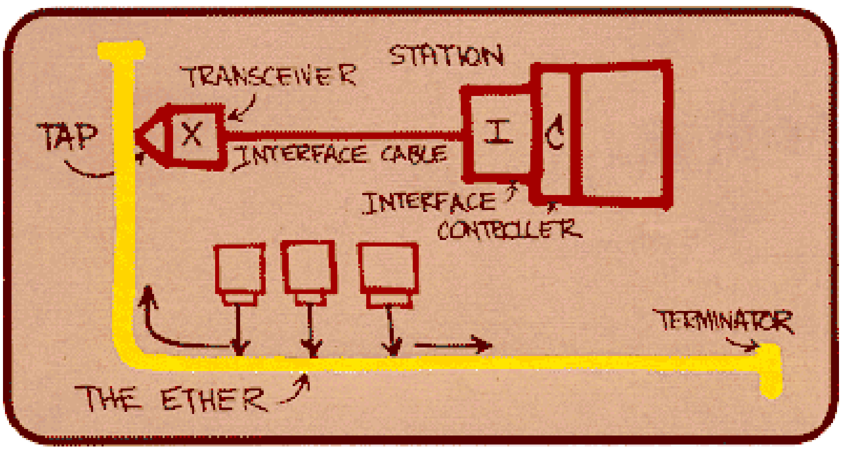
Outline

- ❑ Static multiplexing
- ❑ Dynamic channel allocation
- ❑ Collision-based protocols
- ❑ Contention-free protocols
- ❑ Limited contention protocols
- ❑ ***Case study: Ethernet***

A Case Study: Ethernet

"Dominant" wired LAN technology:

- ❑ Cheap $20 for 100Mbs!
- ❑ First widely used LAN technology
- ❑ Simpler, cheaper than token LANs and ATM
- ❑ Kept up with speed race: 10 Mbps – 10 Gbps



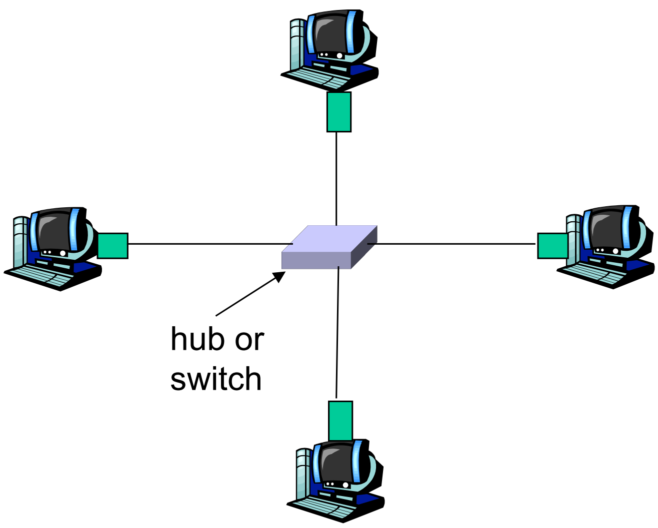
Bob Metcalfe's Ethernet sketch

Ethernet

- A practical example, dealing (mostly) with MAC: Ethernet
  - Standardized by IEEE as standard 802.3
  - Part of the 802 family of standards dealing with MAC protocols
  - Also contains PHY and DLC specifications
- Issues
  - Cabling
  - Physical layer
  - MAC sublayer
  - Switched Ethernet
  - Fast & Gigabit Ethernet

Star Topology

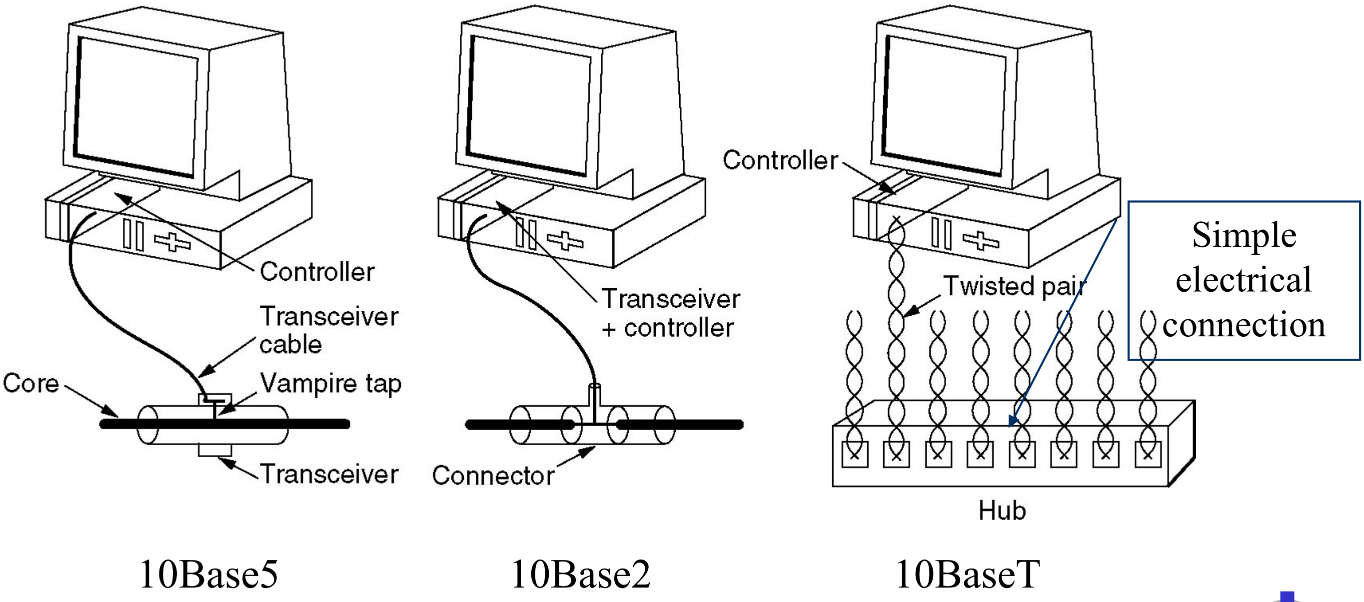
- Bus topology popular through mid 90s
- Now star topology prevails
  - Main advantage: easier (automatic) maintenance in case of a misbehaving adapter
- Connection choices: hub or switch (more later)



Ethernet Cabling

"Yellow cable"

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings



Unreliable, Connectionless Service

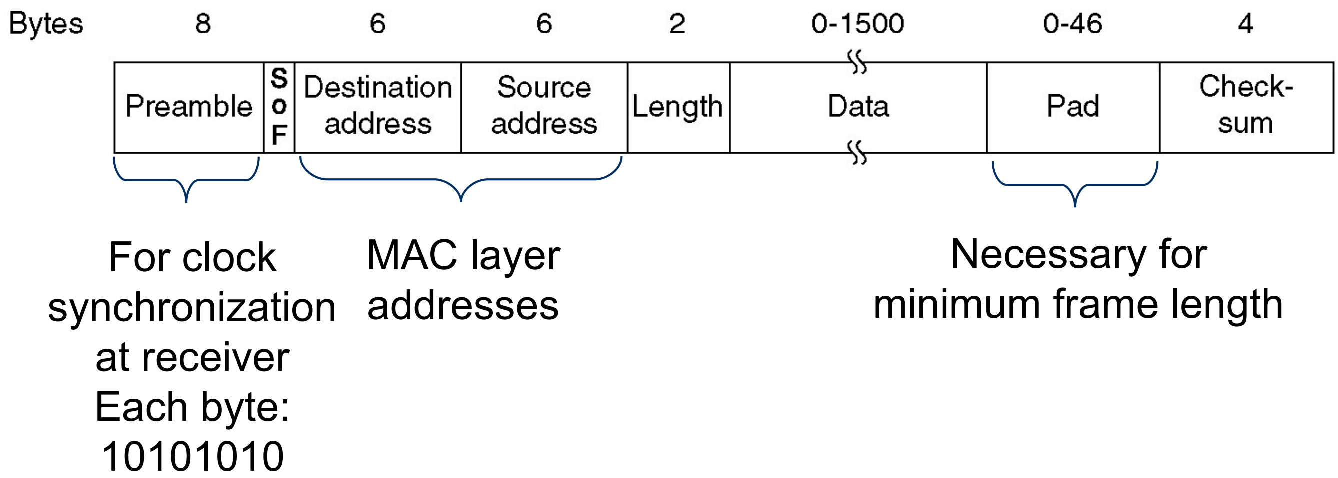
- Connectionless:
  - No handshaking between sending and receiving adapter
- Unreliable:
  - Receiving adapter doesn't send acks or nacks to sending adapter
  - Stream of datagrams passed to network layer can have gaps
  - Gaps will have to be filled by higher layers (if required)
  - Otherwise, the application will see the gaps

Ethernet Physical Layer

- Details depend on medium
- Common: Manchester encoding
  - At +/- 0.85 V (typically) to ensure DC freeness
- With option for signal violations
  - Used to demarcate frames

Ethernet MAC Sublayer

- Essentially: CSMA/CD with binary exponential backoff
- Frame format:



Ethernet Uses CSMA/CD

- No slots
- Adapter doesn't transmit if it senses that some other adapter is transmitting, that is, **carrier sense**
- Transmitting adapter aborts when it senses that another adapter is transmitting, that is, **collision detection**
- Before attempting a retransmission, adapter waits a random time, that is, **random access**

Ethernet CSMA/CD Algorithm (1)

1. Adapter receives datagram from net layer & creates frame
2. If adapter senses channel idle, it starts to transmit frame. If it senses channel busy, waits until channel idle and then transmits
3. If adapter transmits entire frame without detecting another transmission, the adapter is done with frame !
4. If adapter detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, adapter enters **exponential backoff**: after the n^{th} collision, adapter chooses a K at random from {0, 1, 2, …, 2^m-1} with m = min{10, n}. Adapter waits K×512 bit times and returns to Step 2

Ethernet CSMA/CD Algorithm (1)

Jam Signal:

- make sure all other transmitters are aware of collision; 48 bits

Bit time:

- .1 microsec for 10 Mbps Ethernet ;
  for K = 1023, wait time is about 50 msec

Exponential Backoff:

- *Goal*: adapt retransmission attempts to estimated current load
  - Heavy load: random wait will be longer
- First collision: choose K from {0, 1}; delay is K× 512 bit transmission times
- After second collision: choose K from {0,1,2,3}…
- After ten collisions, choose K from {0,1,2,3,4,…,1023}

CSMA/CD Efficiency

- t_{prop} = max prop between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame
- Exact computation of CSMA/CD efficiency is beyond the scope of this course; the following gives a good approximation:

$$efficiency = \frac{1}{1 + 5 t_{prop} / t_{trans}}$$

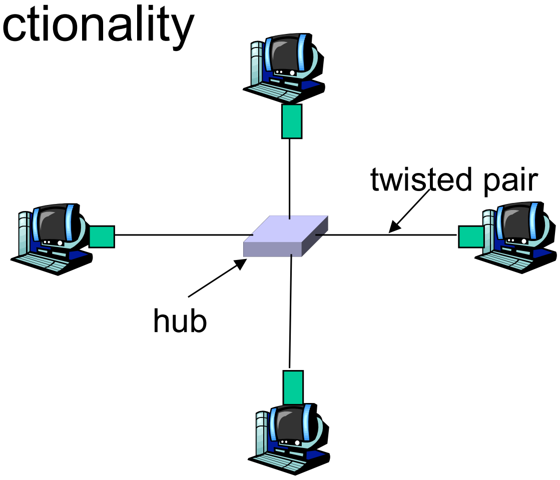
- Efficiency goes to 1 as t_{prop} goes to 0
- Goes to 1 as t_{trans} goes to infinity
- Much better than ALOHA, but still decentralized, simple, and cheap

Hubs

Hubs are essentially physical-layer repeaters:

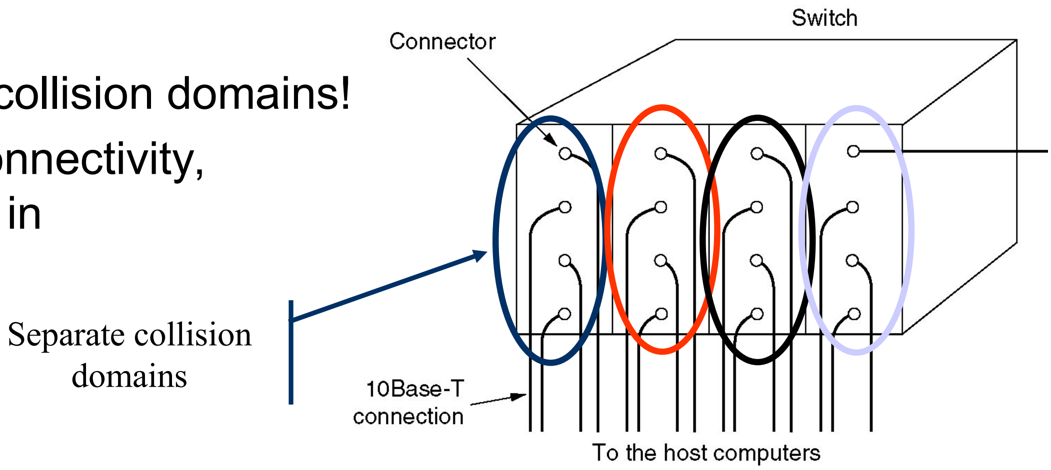
- Bits coming from one link go out all other links
- At the same rate
- No frame buffering
- No CSMA/CD at hub: adapters have to detect collisions
- Provides net management functionality

Consequence: As a *hub* is electrically connected, it realizes a single collision domain



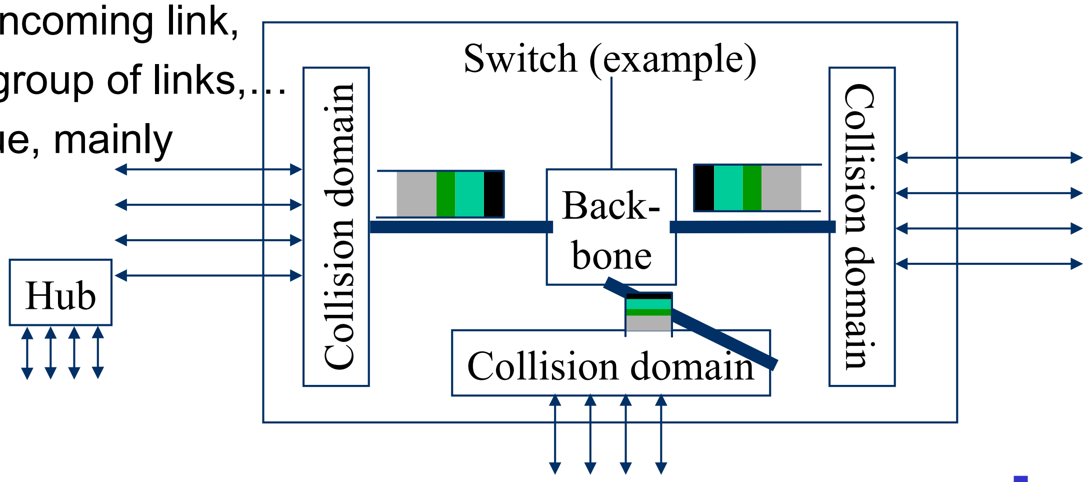
Switched Ethernet

- With conventional 10Base5/10Base2 Ethernet, all stations attached to a single cable form a ***collision domain***
  - Packets from all these stations might potentially collide
  - Big collision domains stress the CSMA/CD mechanism, reducing performance
- How to reduce collision domains but still maintain connectivity of local stations?
  - Use smaller collision domains!
  - To ensure connectivity, put a ***switch*** in



An Ethernet Switch

- Unlike a hub, not a simple electrical connection for a star-wired topology
- How to exchange packets between different collision domains?
  - Switch contains buffers to intermediately store incoming packets before forwarding them towards their destination
  - Different buffer structures possible:
    - one per incoming link,
    - one per group of links,…
    - Cost issue, mainly



Fast Ethernet

- “Normal” (even switched) Ethernet “only” achieves 10 MBit/s
- 1992: Build a faster Ethernet!
  - Goals: Backward compatible, stick with the old protocol to avoid hidden traps, get job done quickly
  - Result: 802.3u – aka “Fast Ethernet”
- Fast Ethernet
  - Keep everything the same (frame format, protocol rules)
  - Reduce bit time from 100 ns to 10 ns
  - Consequences for maximum length of a wiring segment, minimum packet sizes? (Recall unavoidable collisions in CSMA!)

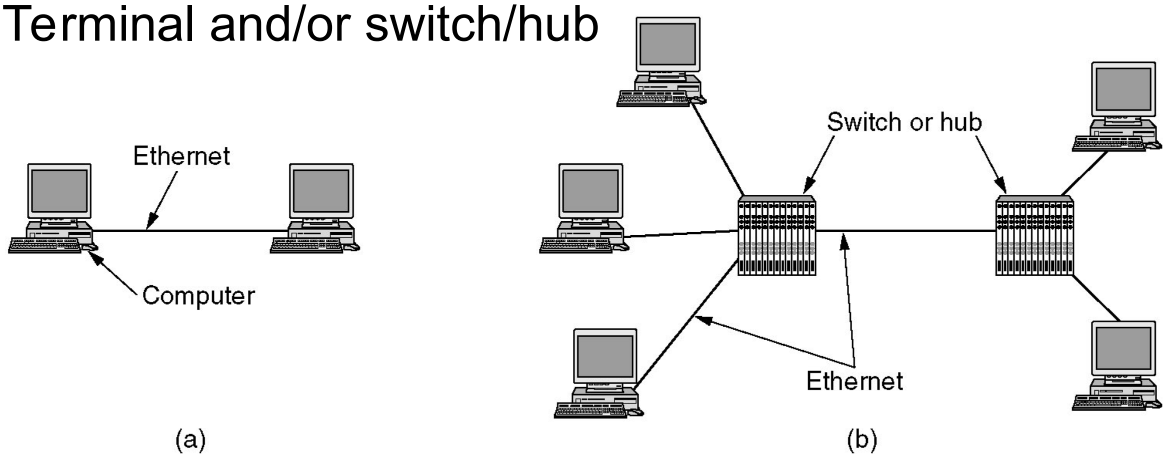
Fast Ethernet – Cabling

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

- Standard category 3 twisted pairs (telephony cables) cannot support 200 MBaud over 100 m cable length
  - Solution: use 2 pairs of wires in this case, reduce baud rate (recall Manchester has baudrate = 2 bitrate)
- Also, cat. 5 cabling does not use Manchester, but 4B/5B (thus, 4 bits are send in 5 signal steps)

Gigabit Ethernet

- Ok: can we go another factor of 10 faster?
  - 1995 – gigabit Ethernet
  - Goal: again, keep basic scheme as it is
- In Gigabit Ethernet (and Fast Ethernet), each wire has ***exactly*** two machines attached to it:
  - Terminal and/or switch/hub



Gigabit Ethernet

- With a switch
  - No shared collision domains ! no collision ! no need for CSMA/CD
  - Allows full-duplex operation of each link
- With a hub
  - Collisions, half duplex, CSMA/CD
  - Maximum cable distance is reduced to 25 m
  - Actually: not very sensible combination from a cost/performance perspective (you already paid the cabling cost...)
- Cabling:

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP

Further Ethernet Evolution

- Of course, the speed evolution did not stop at 1 Gbit/s
  - 2.5 GBASE-T: 2.5 Gbit/s
  - 5 GBASE-T: 5 Gbit/s
  - 10 GBASE-T: 10 Gbit/s
  - 25 GBASE-T: 25 Gbit/s
  - 40 GBASE-T: 40 Gbit/s
- Some observations:
  - The higher the bitrate, the shorter the maximum cable length
  - At rates higher than 10 Gbit/s different connectors than RJ45 are used
  - At rates ≥ 10 Gbit/s energy consumption of transmission over copper raises significantly
  - Fiber optical transmission can support much larger distances, has lower error rates, uses less energy and is thus often preferred for very high data rates

Conclusions

- MAC protocols are a crucial ingredient, pivotal for good performance
  - Static multiplexing just won’t do for bursty traffic
- Main categories: Collision, collision-free, limited contention
- Main figures of merit: Throughput, delay, fairness
  - There hardly is a “best” solution
- Important case study: Ethernet
  - Main lesson to be learned: Keep it simple!