CS 655 – System and Network Architectures and Implementation

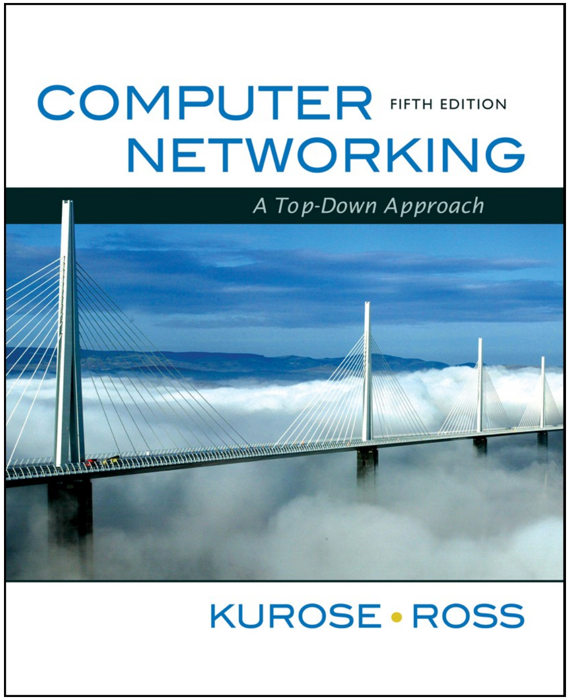
Module 1 - Channels

Martin Karsten

mkarsten@uwaterloo.ca

Notice

Some slides and elements of slides are taken from third-party slide sets. In this module, parts are taken from the Kurose/Ross slide set. See detailed statement on next slide...



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- ❑ If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- ❑ If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Overview

- Signal Transmission
- Framing
- Error Detection / Correction
- Medium Access Control
- Principles
  - Encapsulation
  - Performance

Signal Transmission

- encode/decode binary information as
  - electrical signals
  - acoustic signals
  - optical signals
  - radio signals
  - etc.

Signals

- challenge: signal fading
  - signal level
  - transition edge
- binary vs. n-ary signals?
  - robustness vs. efficiency

- communication -> reconstruct original signal

Physical Media

- properties
    - capacity
    - shielding
    - propagation

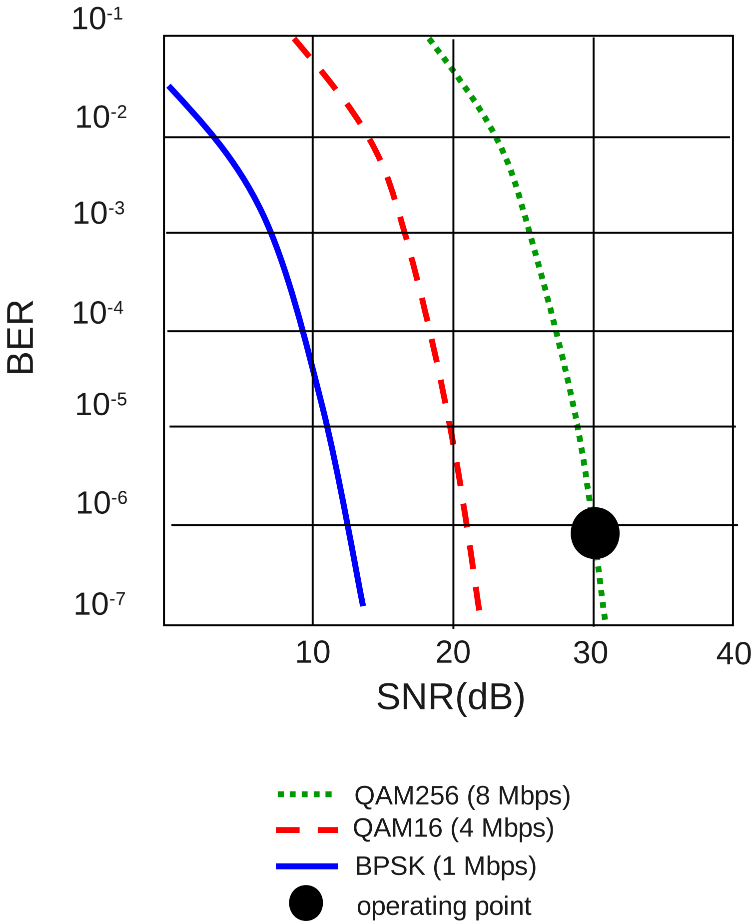
- cable
    - cost & quality: twisted pair < coax < fiber

Radio Transmission

- medium: electromagnetic spectrum
  - public good: regulated vs. unregulated bands
- no wire – no shielding
- propagation effects
  - obstruction (path loss)
  - reflection (multipath)
  - interference
- transmission pattern: base station – mobile
  - no ad-hoc or peer-to-peer communication

Radio Transmission – Adaption

- SNR: signal-to-noise ratio
- BER: bit error rate
- SNR vs. BER trade-offs
  - increase power -> increase SNR -> decrease BER
  - for fixed SNR: choose appropriate coding, such that BER is reasonable



Radio – Power Management

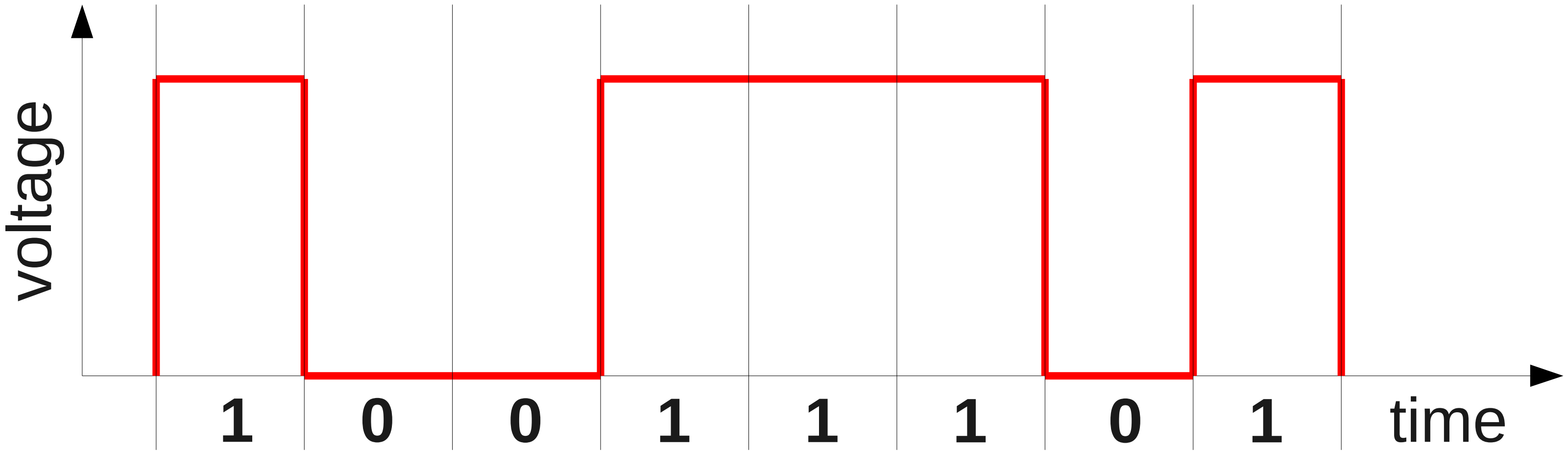
- radio ~ battery-powered devices
  - sending / receiving consumes significant power
- power savings mode: disable receiving
  - maybe hibernate device
- coordinate wake/sleep cycles with neighbor
  - fixed schedule or announce sleep periods
  - simple for base station scenarios, else difficult
- also: control sending power / interference

Optical Transmission

- computers are built using electrical components
- vs. much more efficient optical signals
  - “more efficient” -> more transitions per time
- electrical circuits (especially memory) have difficulty with keeping up...
- transmission pattern: point-to-point
- optical switches?
  - optical memory? optical delay lines

Binary Coding

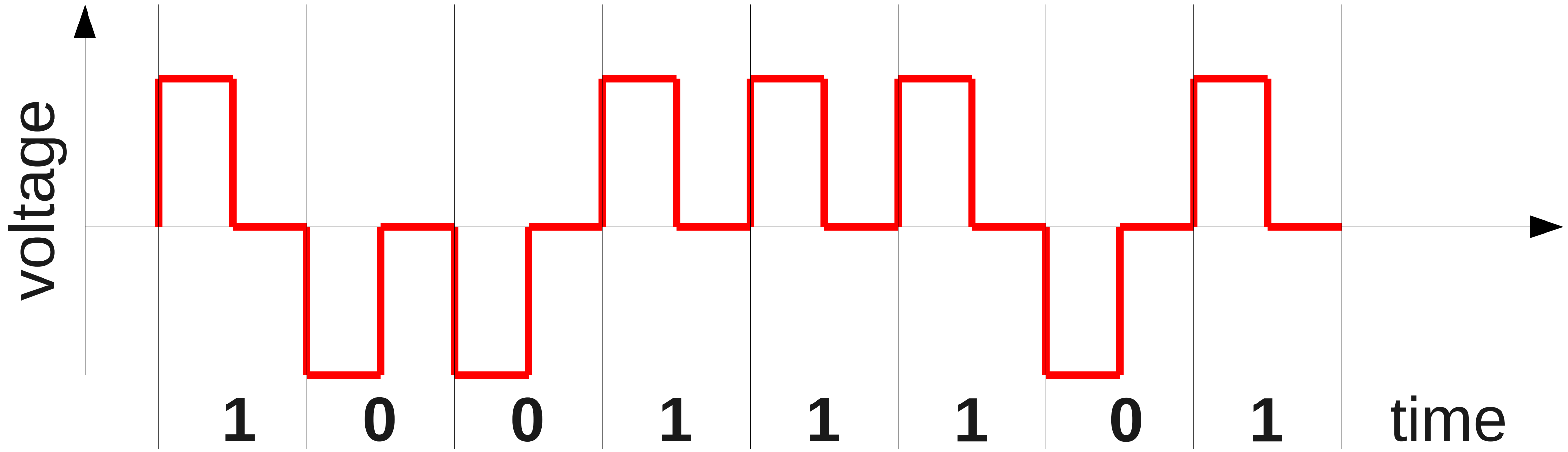
- simplest coding: unipolar, non-return-to-zero



- clock synchronization?
- direct current (DC) bias?

Self-Clocking

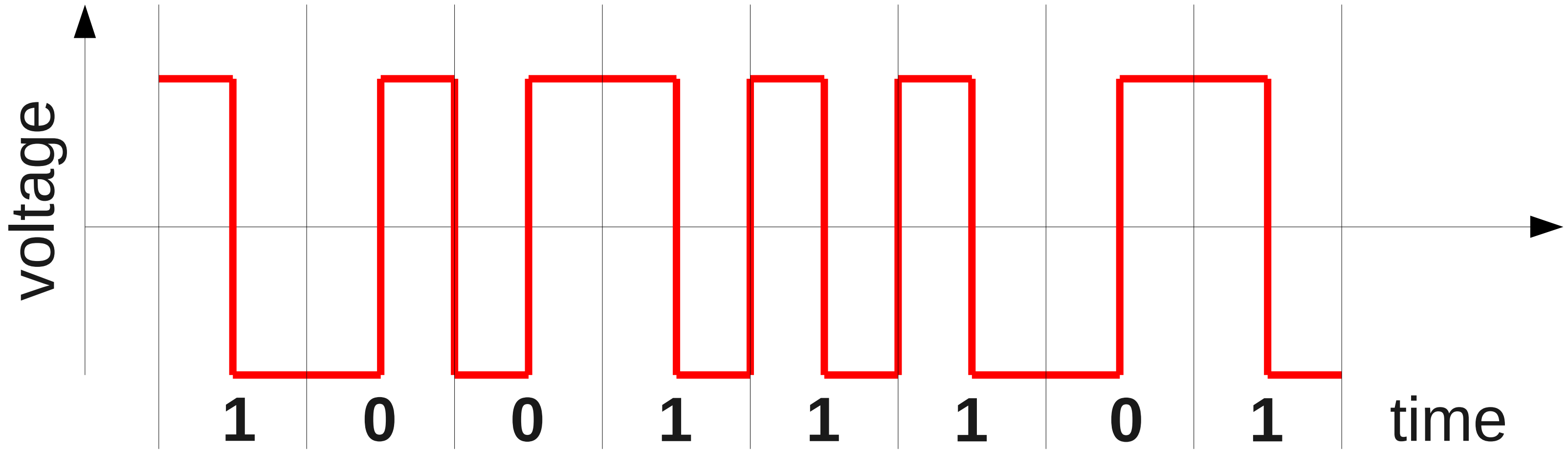
- bipolar, return-to-zero



- transition in middle of interval -> clock signal
  - but: only 50% efficient

Manchester Encoding

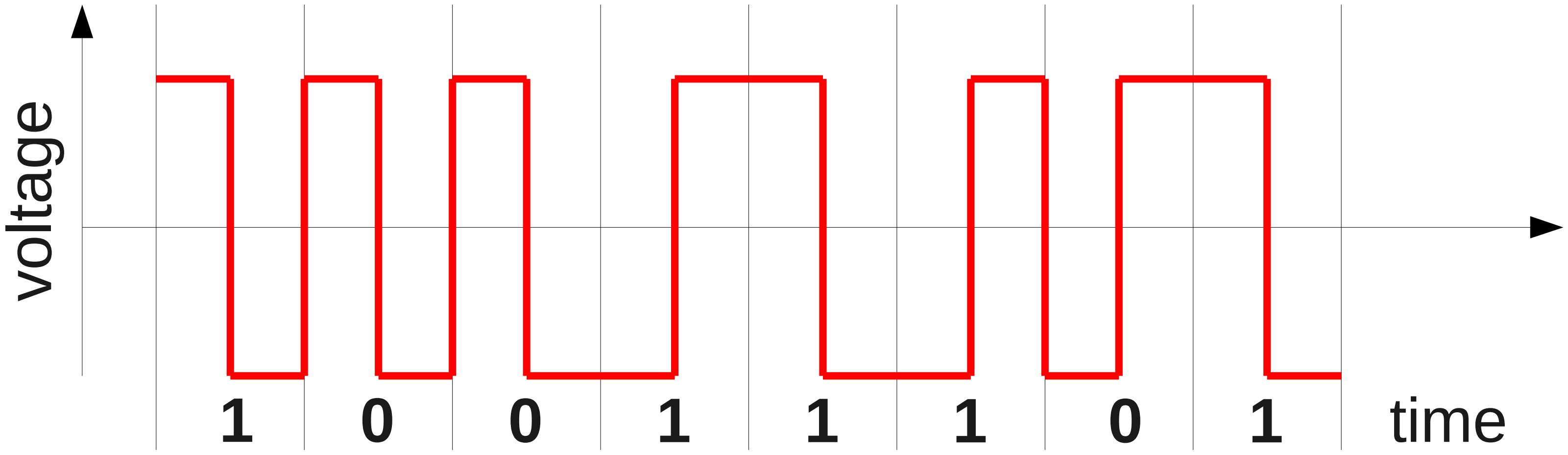
- direction of transition in middle of signal period



- no DC bias at all
- inversion (cable swapped)?
    - both versions exist in literature and standards

Differential Manchester Encoding

- presence of transition at start of period



- transition easier to detect than level
- robust against inversion

4B/5B Encoding

- encode 4 bits in 5 NRZ transitions
- fixed mapping dictionary
  - each code word encoded with at least 2 transitions
  - coding room for 'idle' and other control signals
- more recent: 64B/66B
  - use scrambler instead of dictionary
  - first two bits: preamble, guarantee transition
- dictionary and scrambling also limit DC bias

Framing

- split bit stream in discrete units
- signal start and end of message
  - basis for any kind of reliable communication
  - define unit for basic error detection and correction

- signal/code for 'idle'?
  - yes: can use frame length field
  - no: must use frame delimiter

Frame Delimiter

- example: PPP
- Point-to-Point Protocol (PPP)
  - transparency: transmit binary data over anything
  - assume 0- and 1-signal, but no 'idle' signal
  - operates on bytes

- designate special bit pattern as delimiter
  - e.g., 01111110 (for both start and end)
- what if 01111110 appears in data stream?

Byte Stuffing

- PPP sender algorithm

  if 01111110 appears in data stream:

  insert another 01111110 before

- PPP receiver algorithm

  if 01111110 appears in communication stream:

  followed by 01111110 -> remove one of them

  else -> 01111110 is a delimiter

Notes

- can have different start/end delimiters for robustness
- generalized sender algorithm

  insert *escape pattern* before any occurrence of control patterns in data (including escape itself)
- generalized receiver algorithm

  if escape pattern in stream:

  remove and treat next symbol as data

Escape Character

- e.g., in many programming languages:

  ```
  print “Tom said “Hello” to me.”;
  ```

  ... is a syntax error, because double quotation mark is the string delimiter token
- use *escape character* (backslash):

  ```
  print “Tom said \”Hello\” to me.”;
  ```

- same for regular expressions and related tools

Framing with Idle-Signal

- with 'idle'-signal: delimiters not strictly needed
- use 'length' field in message for robustness

- but: why does Ethernet define 'preamble'?
  - for channel cleanup (see medium access control),
  - ...and clock synchronization
    - self-clocking code

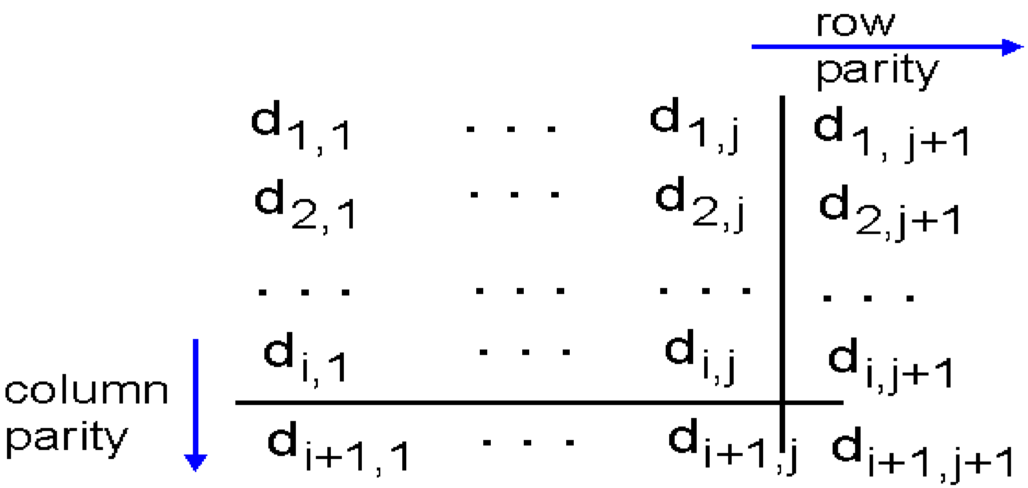
Error Control

- error detection: append parity bit to bit string
  - bit string can be codeword, message, etc.
  - add bit, such that number of 1s is odd
    - 'even' variant exists, as well

- 0111000110101011 -> append 0
- receiver can detect single-bit errors
- correction: *Automatic Repeat reQuest* (ARQ)
  - hold back ACK or send NAK

Forward Error Correction

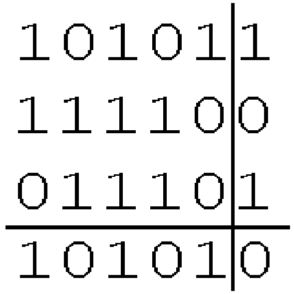
- matrix of parity bits



row parity

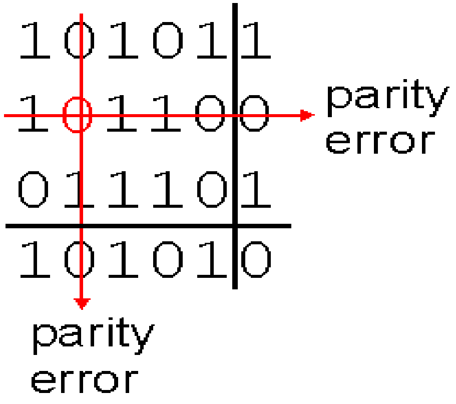
column parity

$$\begin{array}{ccc|c} d_{1,1} & \cdots & d_{1,j} & d_{1,j+1} \\ d_{2,1} & \cdots & d_{2,j} & d_{2,j+1} \\ \cdots & \cdots & \cdots & \cdots \\ d_{i,1} & \cdots & d_{i,j} & d_{i,j+1} \\ \hline d_{i+1,1} & \cdots & d_{i+1,j} & d_{i+1,j+1} \end{array}$$



$$\begin{array}{ccccc|c} 1&0&1&0&1&1 \\ 1&1&1&1&0&0 \\ 0&1&1&1&0&1 \\ \hline 1&0&1&0&1&0 \end{array}$$

no errors



$$\begin{array}{ccccc|c} 1&0&1&0&1&1 \\ 1&0&1&1&0&0 \\ 0&1&1&1&0&1 \\ \hline 1&0&1&0&1&0 \end{array}$$

correctable single bit error

Error Control: Redundancy

- what are we actually doing?
  - add redundancy to message (cf. natural language)
  - opposite of compression!

- data <-> code mapping
  - code properties determine error control capabilities

    -> *Hamming Distance*

Hamming Distance

- number of positions in which two strings of equal length are different

  h(a,b) for strings a, b

- assume code / set of fixed-length bit strings
- Hamming Distance of code C

  $H(C) = \min(h(a,b)\ \forall\ a,b \in C)$

Hamming Distance

- H(C) > n   => n-bit errors can be **detected**
- H(C) > 2n => n-bit errors can be **corrected**

- previous examples
  - simple Parity scheme: H(C) = 2 > 1
    - 1-bit errors can be detected
  - matrix Parity scheme: H(C) = 3 > 2 * 1
    - 1-bit errors can be corrected

- challenge: dictionary not feasible for messages

Internet Checksum

one's complement of sum of one's complements

  - process data as stream of 16-bit words
  - one's complement sum: add carry bit
  - not very strong: errors might go undetected

- why this format?
  - implementation in software
  - incremental update
  - independent of byte ordering
  - see RFC 1071 for discussion

Advanced Error Control

- Cyclic Redundancy Check (CRC)
  - error detection with small, fixed message overhead
  - simple in hardware – used, e.g., in Ethernet
- Erasure Codes
  - consider loss, instead of changes
  - parity -> e.g., in RAID systems
- Fountain Codes
  - any suitable subset of encoded symbols an recreate original message

Medium Access Control (MAC)

Goals

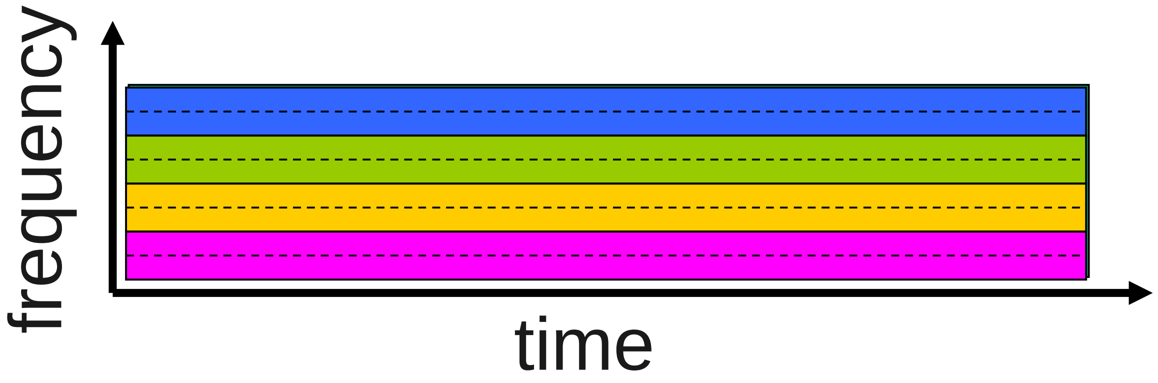
1. high channel utilization for single/few nodes
2. fair sharing for many nodes
3. decentralized, robust
4. simple

Medium Access Control Schemes

- channel partitioning
  - pessimistic: split channel into sub-channels
- random access
  - optimistic: uncoordinated access
  - distributed: handle collisions, resolve contention
- taking turns
  - distributed coordinated: avoid collisions
  - high complexity

Partitioning – Frequency

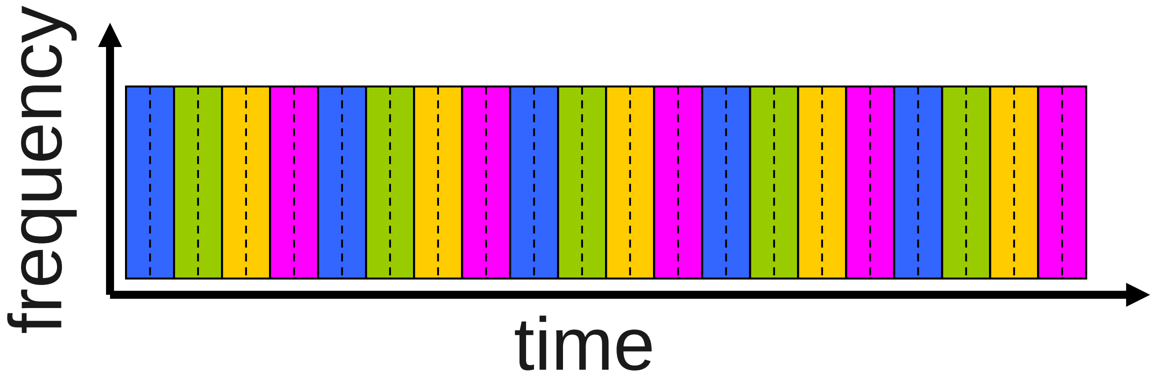
- Frequency Division Multiple Access (FDMA)
- channel spectrum divided into frequency bands
- each station assigned fixed frequency band



- unused capacity goes idle – utilization?

Partitioning – Time

- Time Division Multiple Access (TDMA)
- channel divided into “rounds”
- each station gets fixed length time slot



- unused slots go idle

Partitioning – Code

- Code Division Multiple Access (CDMA)
- ultimately uses multi-level signals
- each station is assigned chipping sequence
  - set of binary code vectors that are *orthogonal*
- downlink (from base station)
  - synchronous: signals add up, but can be decoded
- uplink (from mobile stations)
  - asynchronous: add pseudo-random “noise” signal

Partitioning

- need control functionality to manage channels
  - assign frequency or time slot or codes
  - mobile registration with base station
    - rare event (compared to message transmission)
  - ultimately: random access (bootstrapping problem)
    - truly global codes without collision not feasible

=> control overhead

- centralized
- utilization with few stations?

Random Access

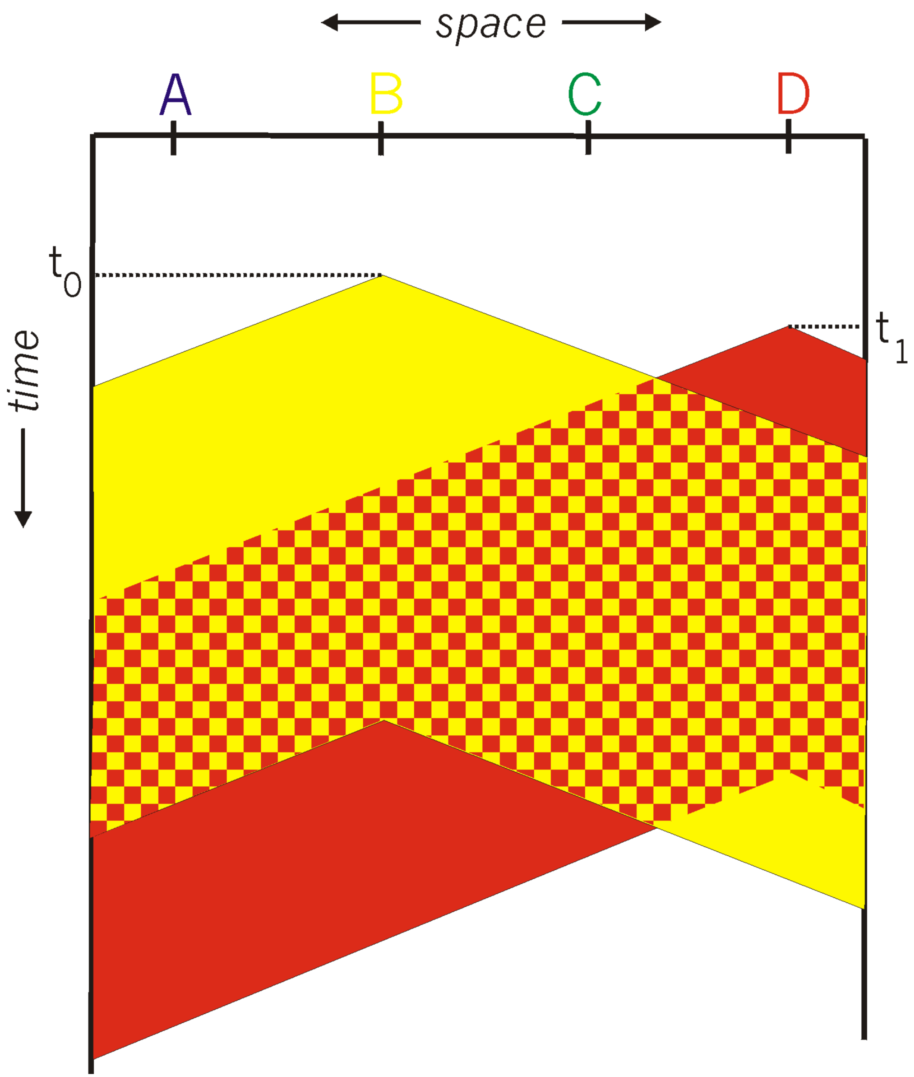
- assumption: signal overlap detectable as error
- potential collision not avoidable
- first approach: send any time (ALOHA)
  - fixed message length
  - detect collisions, random wait time, retransmit
  - best-case utilization: 18%
- next: send at slot intervals (slotted ALOHA)
  - best-case utilization: 36%

Carrier Sensing

- Carrier-Sense Multiple Access (CSMA)
  - listen to channel before transmission
  - if channel busy, wait
    - persistency – when to send after channel is sensed free?
- collisions? still possible...
  - non-zero propagation delay

CSMA – Collisions

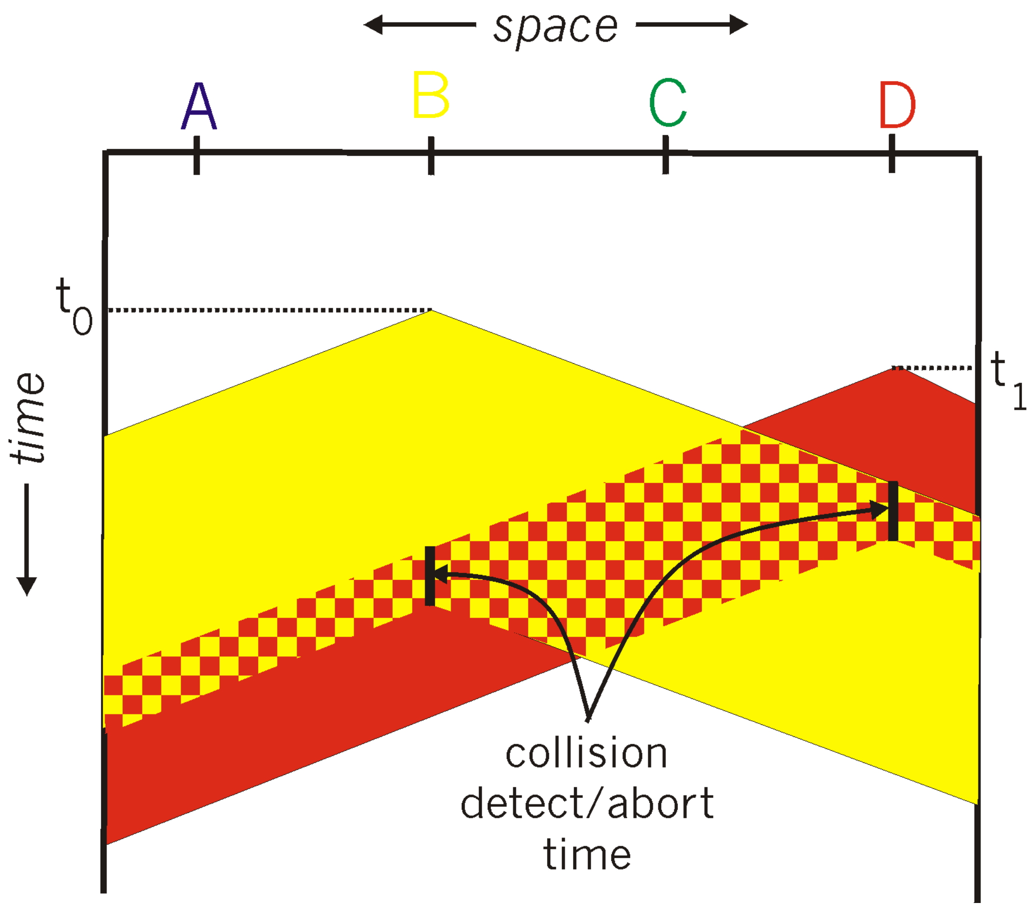
- propagation delay gap: two nodes do not hear each other
    - propagation speed
    - distance
- collision
    - entire message corrupt



Collision Detection

- compare signal on channel while own message
- detect collision locally -> abort transmission
  - reduce channel wastage
- easy in wired medium
  - compare signal strengths of transmitted/received signall
- difficult in radio networks
  - received signal strength overwhelmed by local transmission

Collision Detection



Minimum Message Size

- carrier sensing and minimum message size
  -> effective channel reservation scheme
- assume maximum distance d, prop speed v
  => maximum travel time: d/v
- remote station could transmit just before d/v
- if no collision after 2 * d/v -> channel reserved
- assume transmission rate b
  => minimum message size: b * 2 * d/v

Ethernet

- 1-persistent CSMA/CD, no ACKs
- minimum message size -> channel reservation
- but: message simply lost, if no receiver present
- collision -> randomized exponential backoff
  - increase wait time during continuous collisions
  - adaptive to level of load/contention
- modern versions: not much need for MAC
  star topology & network forwarding

Collision Avoidance

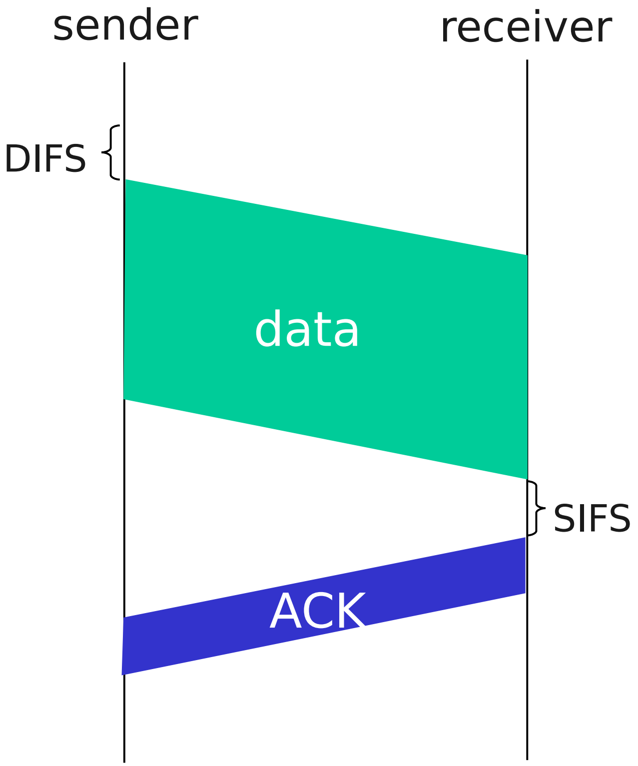
- radio: local collision detection not feasible
  - single radio cannot transmit/receive at same time
  - two radios: local signal would drown out remote
- hidden terminal problem

-> more conservative collision avoidance

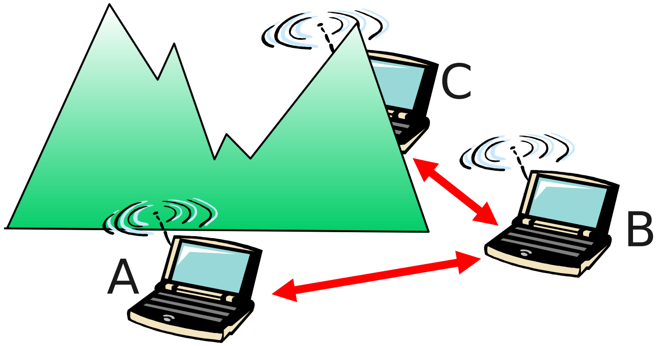
  - exponential backoff during CSMA
  - RTS/CTS mechanism to mitigate collision effect
  - ACKs to confirm successful transmission

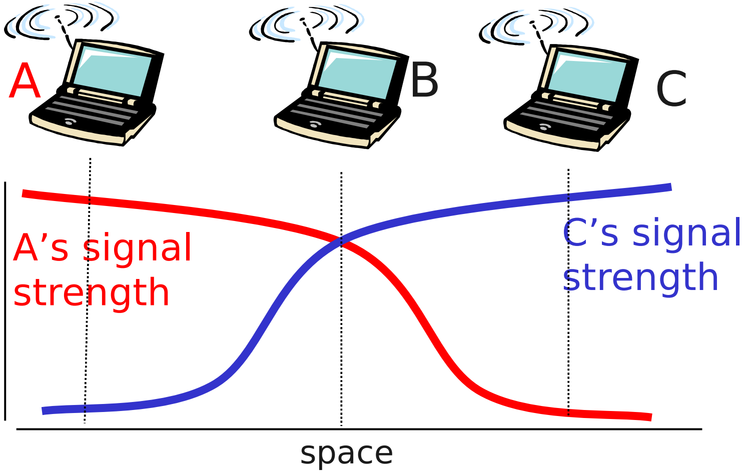
Priorities

- wait time counts down when medium is idle
- wait time priorities built into CSMA mechanism
    - SIFS – last receiver / ACK
    - PIFS – base station / control
    - DIFS – any station / data
- wait time includes backoff if applicable



Hidden Terminal

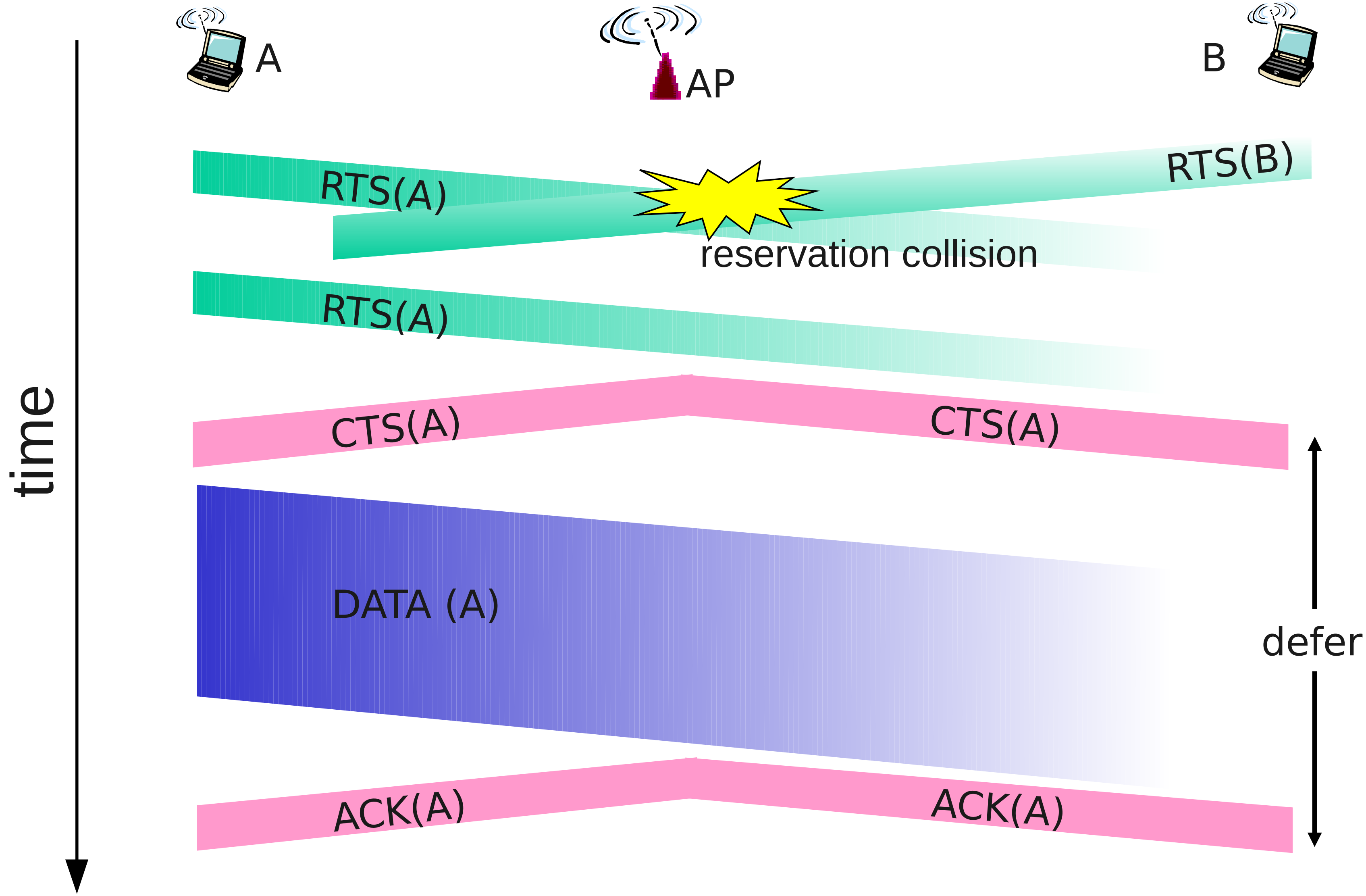




- obstruction
  - A,B hear each other
  - B,C hear each other
  - A,C can not

- signal attenuation
  - A,B hear each other
  - B,C hear each other
  - A,C can not

RTS/CTS


A
AP
B
RTS(A)
RTS(B)
reservation collision
RTS(A)
CTS(A)
CTS(A)
DATA (A)
defer
ACK(A)
ACK(A)
time

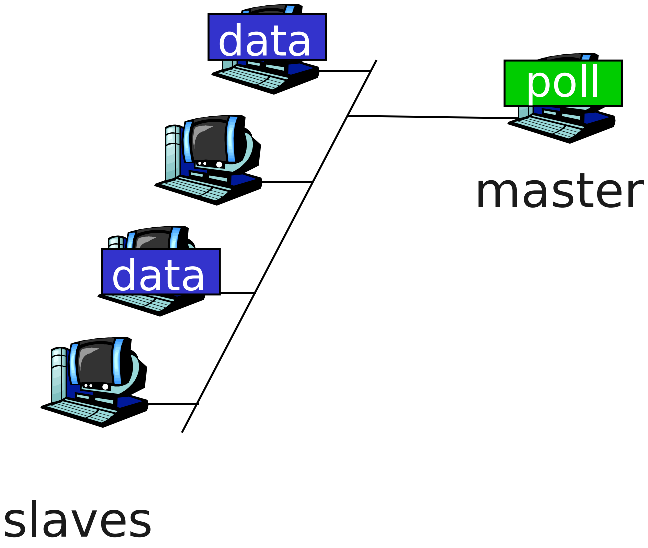
Taking Turns

- channel partitioning protocols:
  - static, coordinated
  - very efficient at high load
  - low utilization at low load
- random access
  - little a-priori coordination
  - very efficient at low load
  - collision overhead at high load

-> try dynamic, coordinated

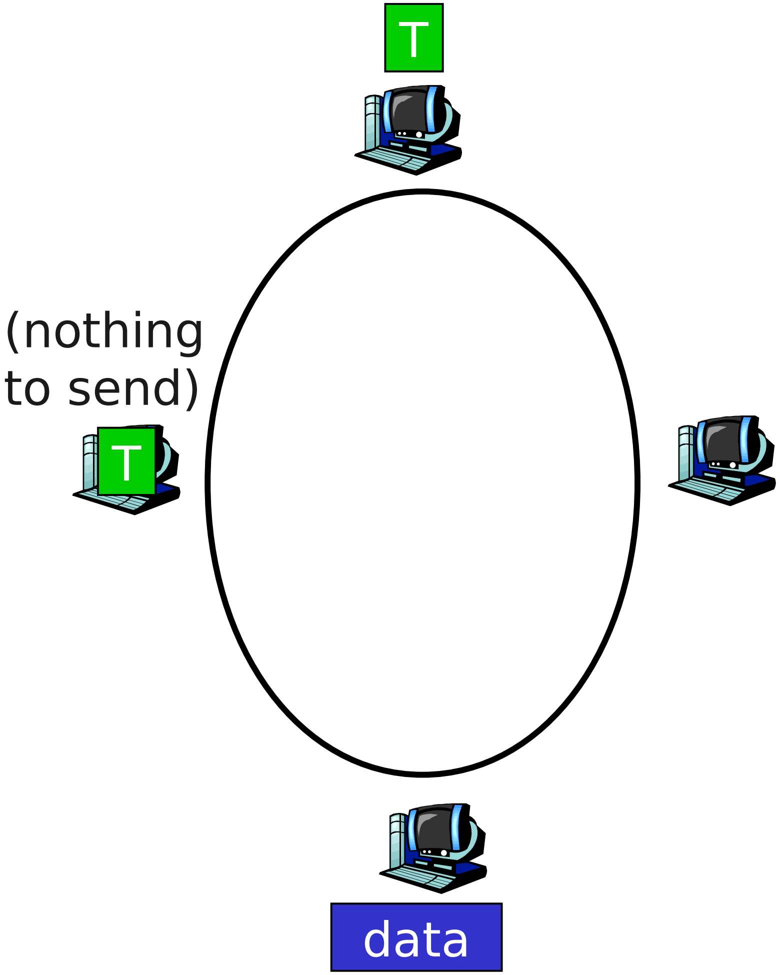
Centralized Polling

- master node "invites" slave nodes to transmit
- traditional scenario: host/terminal networks
- concerns
  - polling overhead
  - latency to send
  - single point of failure (master)



Token Passing

- control *token* passed from one station to next
- examples:
  Token Bus, Token Ring
- concerns
  - token overhead
  - latency
  - single point of failure (token)



Taking Turns

- efficient at low and high load
- manageability / control
    - minimum transmission rate
    - maximum latency
- but:
    - complexity
        - token passing: complexity in each station
    - obsolescence by point-to-point/switched Ethernet

Encapsulation

- wrap payload data into message format
- message header/trailer for control
- payload might be message itself

  -> encapsulation / layering
- typical header/trailer:
  - type, length, addresses, contro, checksuml

Performance

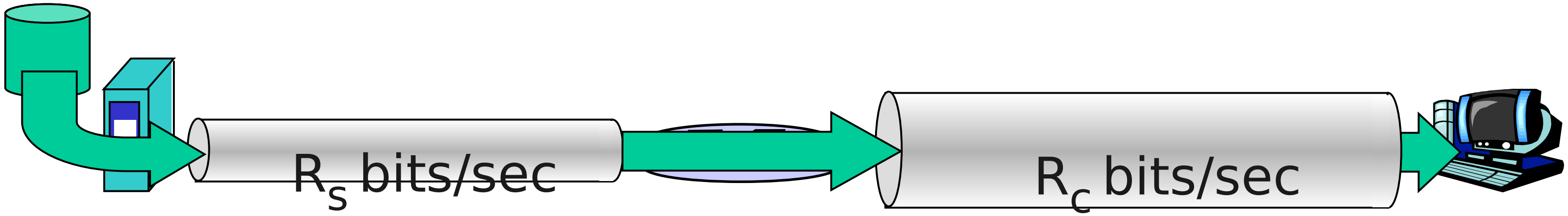
- throughput vs. delay vs. loss (later)
- example: Tim Hortons
  - number of served customers per hour
    - throughput
  - vs. time to wait for coffee
    - delay
- high throughput does not guarantee low delay
  - synchronous vs. asynchronous execution of job
  - high level of multiplexing: good for throughput

Throughput

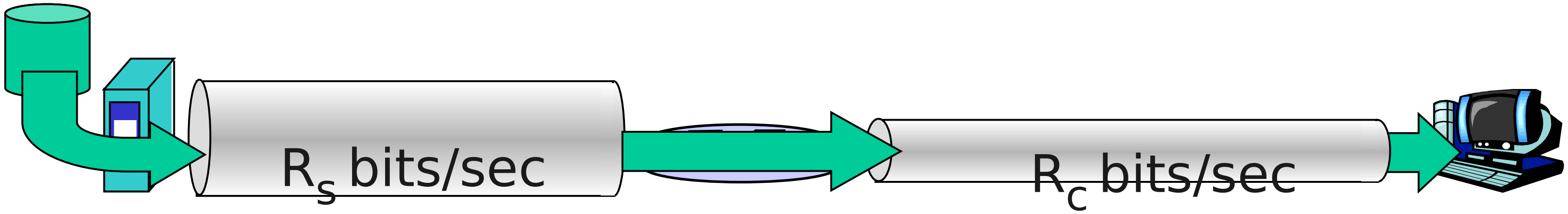
- transfer rate in bits/time
- always an average
  - different time scales relevant in different scenarios
    - streaming: minimum throughput needed “continuously”
      - i.e., over short time periods
    - download: average throughput
      - i.e., overall average throughput relevant
- capacity sharing: fairness vs. utility
  - consider two identical file downloads
  - sequential vs. concurrent download – utilities?

Throughput

- "width" of pipe
- multiple transmission segments
  - minimum throughput segment is *bottleneck*



- vs.



Delay

- processing delay
  - local processing at communication node
- queueing delay
  - later
- transmission delay (packetization)
  - message size / link throughput
- propagation delay
  - propagation speed / distance

Delay

- speed / length of pipe
- multiple transmission segments
    - total delay is sum of delays