Link Layer

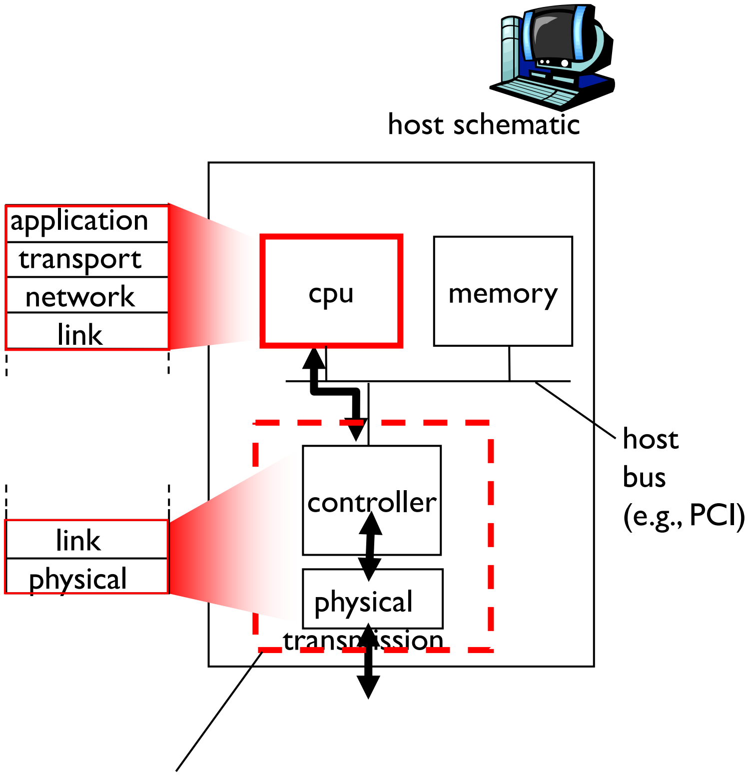
Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 16

puc@marshall.edu

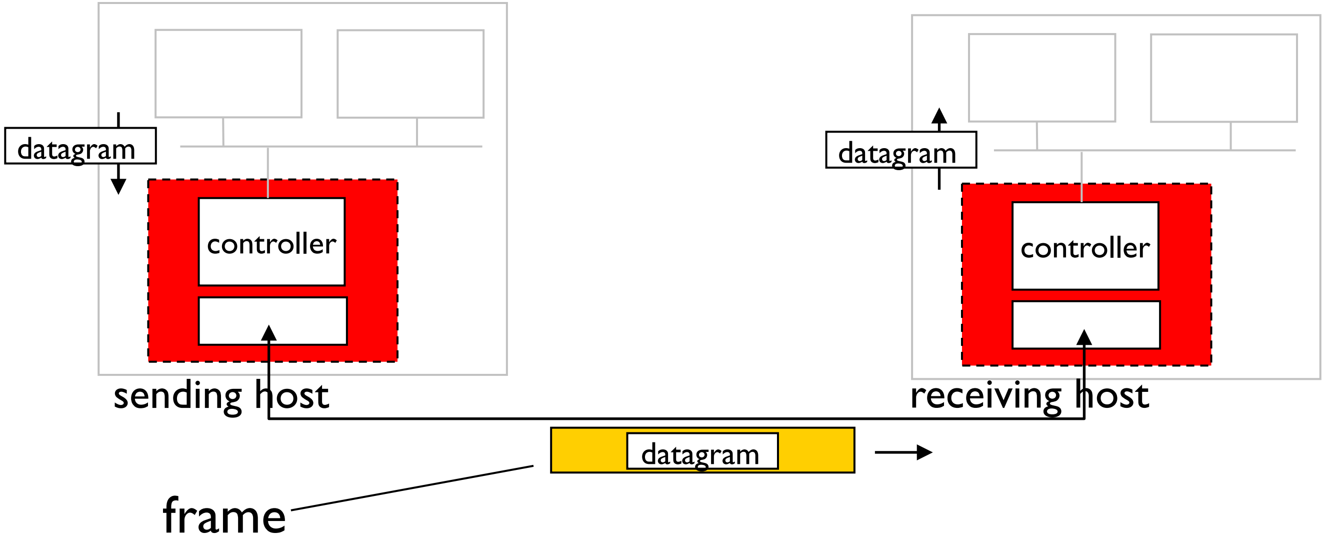
Where is the link layer implemented?

- figure shows a typical host architecture
- the link layer is implemented in a ***network adapter*** or ***network interface card*** (NIC)
- the heart of NIC: ***link-layer controller***
    - single and special-purpose chip
    - implements many services
        - such as framing, error detection, etc
    - much of link-layer's functionality is implemented hardware



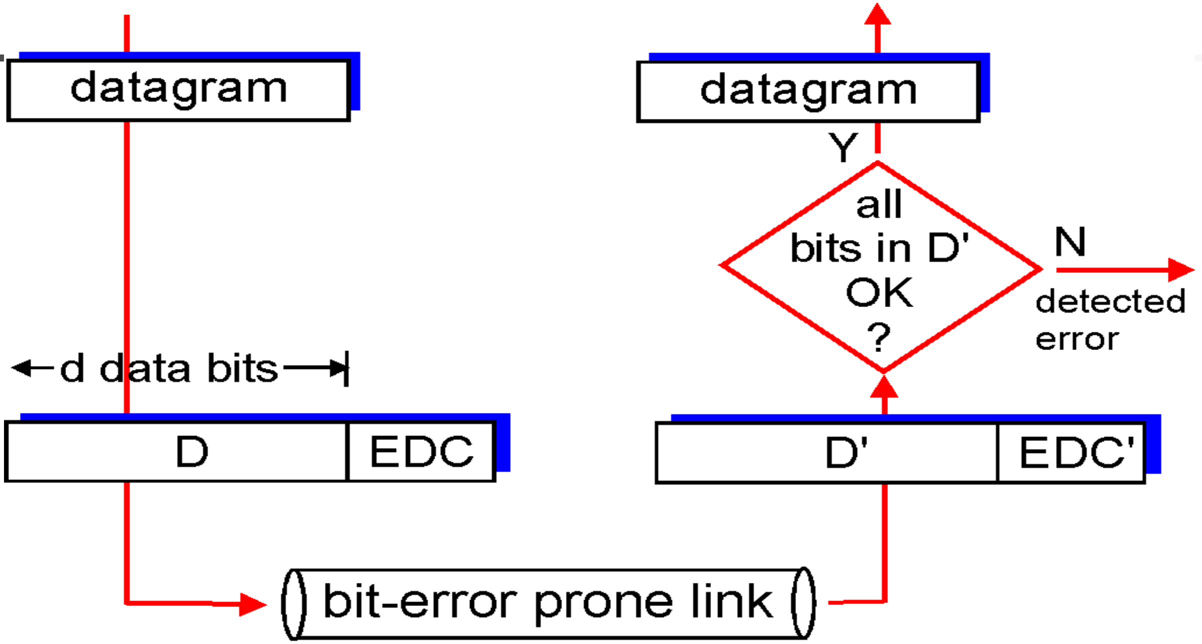
network adapter
or
network interface card (NIC)

Adaptors Communicating



- sending side:
  - takes a datagram
  - encapsulates the datagram in ***frame***
  - adds error checking bits
  - transmits the frame into commu. link
- receiving side
  - receive frame
  - extracts datagram, passes to upper layer at receiving side
  - look for errors

Error Detection



- EDC = Error Detection and Correction bits (redundancy)
- D = Data protected by error checking (may include header fields)
- error detection not 100% reliable! (undetected bit errors are possible)
  - unaware of bit errors
  - deliver a corrupted datagram to net. layer

Parity Checking

Single Bit Parity:

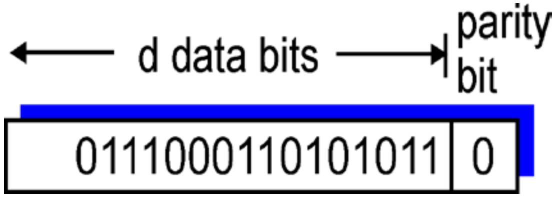
- detect single bit errors

- suppose the information D has ***d*** bits

- ***even parity scheme***: sender includes one additional bit and chooses its value such that the total number of 1s in the *d* +1 bits is ***even***

- ***odd parity scheme***: sender includes one additional bit and chooses its value such that the total number of 1s in the *d* + 1 bits is ***odd***

odd parity scheme:



receiver checks d+1 bits

receiver operations:

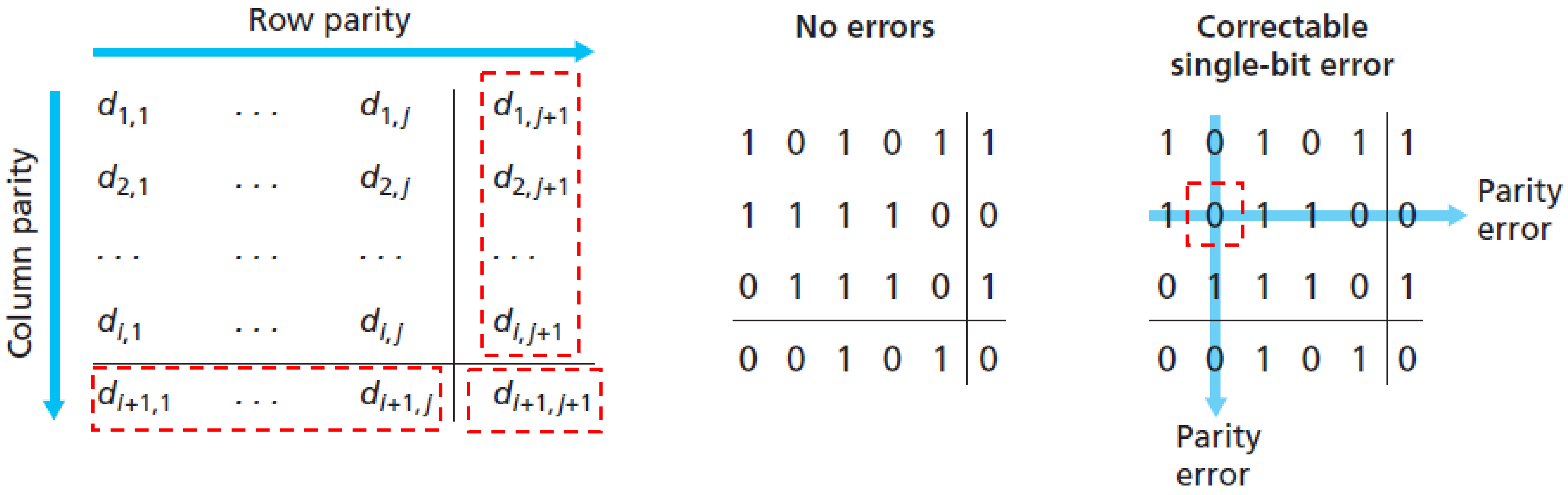
count the number of 1s in the received d + 1 bits.

- odd number of 1s for odd parity
- even number of 1s for even parity

Parity Checking

Two Dimensional Bit Parity:

- ***detect*** and ***correct*** single bit errors
- *d* bits in information D are divided into *i* rows and *j* columns
- a parity value is computed for each row and for each column
- the resulting *i* + *j* + 1 parity bits comprise the link-layer frame's error-detection bits



$d_{1,1}$	. . .	$d_{1,j}$	$d_{1,j+1}$
$d_{2,1}$	. . .	$d_{2,j}$	$d_{2,j+1}$
. . .	. . .	. . .	. . .
$d_{i,1}$	. . .	$d_{i,j}$	$d_{i,j+1}$
$d_{i+1,1}$	. . .	$d_{i+1,j}$	$d_{i+1,j+1}$

No errors

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

Correctable single-bit error

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

Internet Checksum

Goal: detect "errors" in transmitted packet (note: used at **transport layer** only)

Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition of segment contents (1's complement sum)
- sender puts checksum value into checksum field

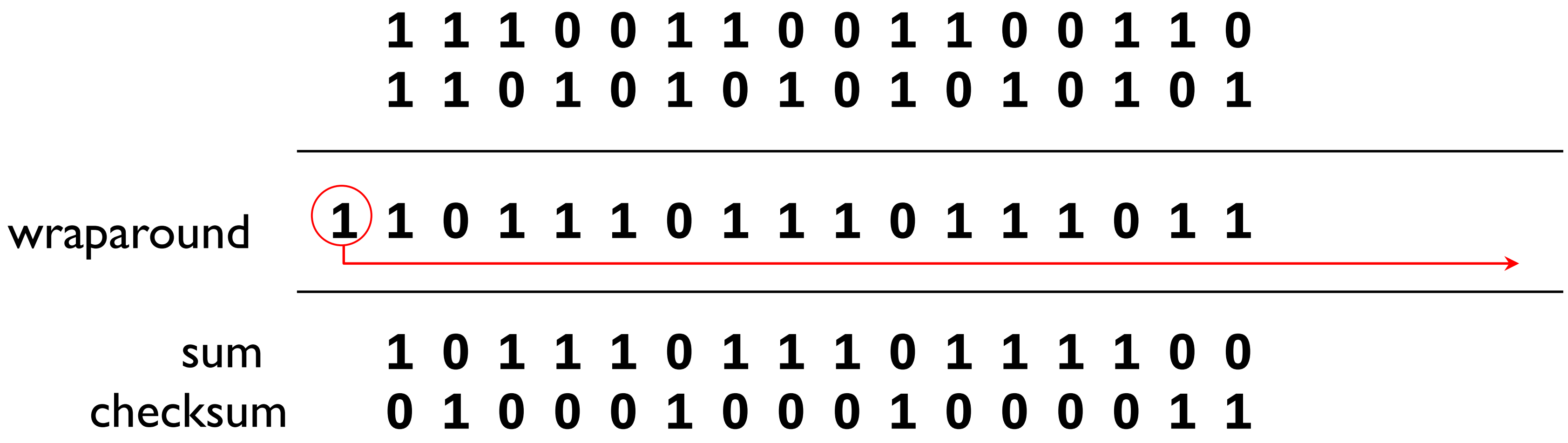
Receiver:

- compute checksum of received segment (taking 1's complement of the sum of the received data)
- check if whether the result is all 1 bits:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

- checksumming at the **transport layer** Vs. cyclic redundancy check at the **link layer**
  - transport layer error detection is implemented in software
    - require simple and fast error-detection scheme
  - link layer error detection is implemented in hardware
    - can rapidly perform the more complex CRC operations

Internet Checksum Example

- note
  - when adding numbers, a carryout from the most significant bit needs to be added to the result
- example: add two 16-bit integers



1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0

1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

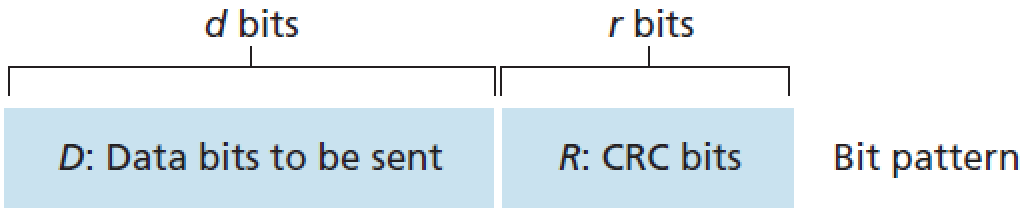
wraparound 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1

sum 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0

checksum 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

Cyclic Redundancy Check (CRC)

- view data bits, D, as a binary number
- choose r + 1 bit pattern (generator), G (agreement between sender and receiver)
  - the most significant (leftmost) bit of G must be 1
- key idea: for a given data, D, the sender will choose r additional bits, R, and append them to D such that the resulting d + r bit pattern is exactly divisible by G using modulo-2 arithmetic.
  - error checking: the receiver divides the d + r received bit by G. if the remainder is nonzero, the receiver knows that an error has occurred.

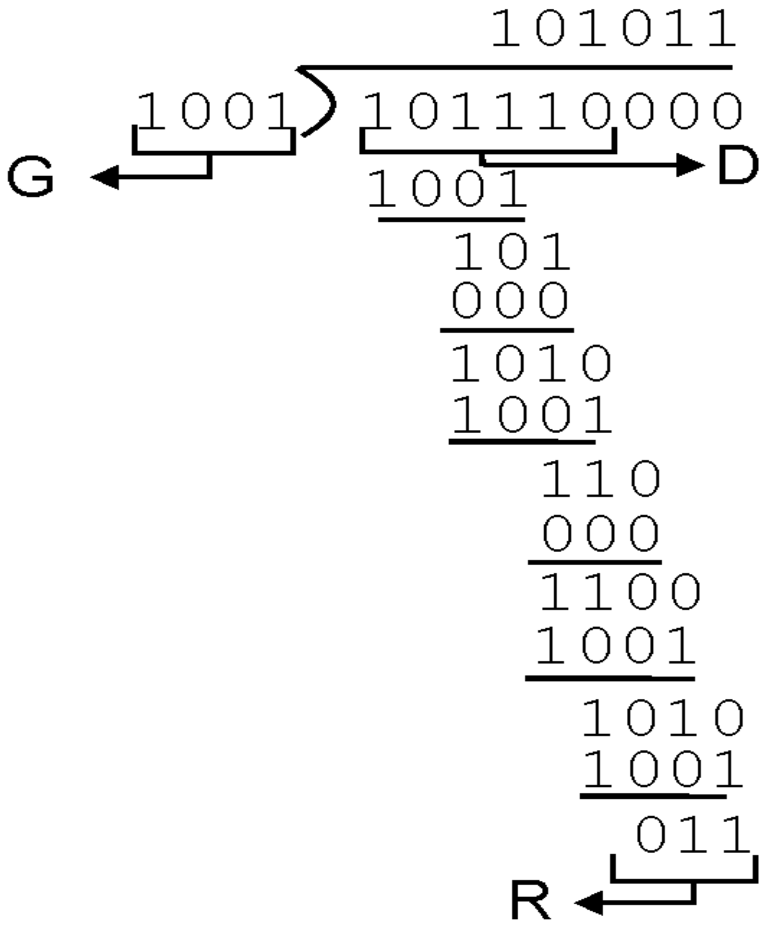


- goal: choose r CRC bits, R, such that

$$\boxed{R = \text{remainder}\left[\ \frac{D \cdot 2^r}{G}\ \right]}$$

CRC Example (cont.)

- D = 1 0 1 1 1 0
- d = 6
- G = 1 0 0 1
- r = 3



Multiple Access Links and Protocols

- two types of "links":
  - **point-to-point link**
    - single sender & single receiver
    - point-to-point protocol (PPP) and high-level data link control (HDLC)
  - **broadcast link**
    - multiple sending and receiving nodes connected to shared wire or medium
    - one node transmits a frame, all other nodes receives a copy

Multiple Access Protocols

- ***multiple access protocols***: regulate nodes' transmission into the shared broadcast channel
- all nodes are capable of transmitting frames, more than two nodes can transmit frames at the ***same time***
  - when this happens
    - all of the nodes receive multiple frames at the same time
    - the transmitted frames ***collide*** at all of the receivers
    - **Collision**
      - none of the receiving nodes can decode the frames that were transmitted
      - all the frames involved in the collision are lost
      - broadcast channel is wasted
- it is necessary to ***coordinate*** the transmissions of the active nodes!

Multiple Access Protocols

- **multiple access protocol**
  - **responsibility**: coordinate active nodes to access broadcast channel

- three categories:
  - **channel partitioning protocols**
    - divide channel into smaller "pieces" (time slots, frequency, etc)
    - allocate piece to node for exclusive use
  - **random access protocols**
    - channel not divided, allow collisions
    - "recover" from collisions
  - **taking-turns protocols**
    - nodes take turns, but nodes with more to send can take longer turns

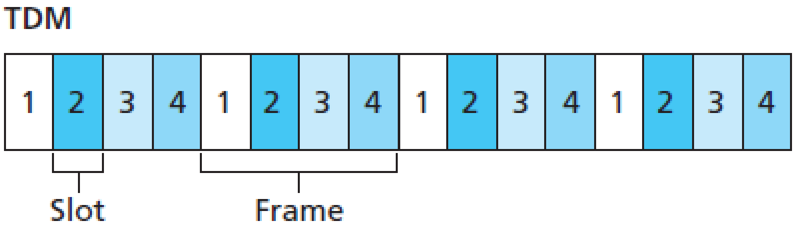
Ideal Multiple Access Protocol

- broadcast channel of rate *R* bps

1. when one node wants to transmit, it can send at rate *R* bps
2. when *M* nodes want to transmit, each can send at average rate *R*/*M* bps
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks
4. simple

Channel Partitioning MAC protocols: TDMA

- TDMA: **T**ime **D**ivision **M**ultiple **A**ccess
  - suppose the channel supports *N* nodes, the transmission rate of the channel is *R* bps.
  - TDMA divides time into ***time frames*** and further divides each time frame into *N* ***time slots***
  - each time slot is assigned to one of the *N* nodes
  - when a node has a packet to send, it transmits the packet during its assigned time slot in the frame
  - Example: a simple four-node TDM

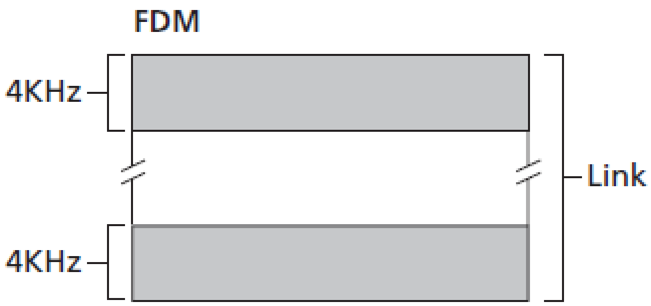


Channel Partitioning MAC protocols: TDMA

- TDMA: **T**ime **D**ivision **M**ultiple **A**ccess
  - ***perfectly fair*** and ***eliminate collisions***
    - each node gets a dedicated transmission rate of *R/N* during each frame time
  - ***drawbacks***
    - a node is limited to an average rate of *R/N* bps even when it is the *only node* with packets to send
    - a node must always wait for its turn in the transmission sequence even when it is the *only node* with packets to send

Channel Partitioning MAC protocols: FDMA

- FDMA: **F**requency **D**ivision **M**ultiple **A**ccess
  - divides the R bps channel into different frequencies
  - assigns each frequency to one of the *N* nodes
  - FDMA creates *N* smaller channels of *R*/*N* bps



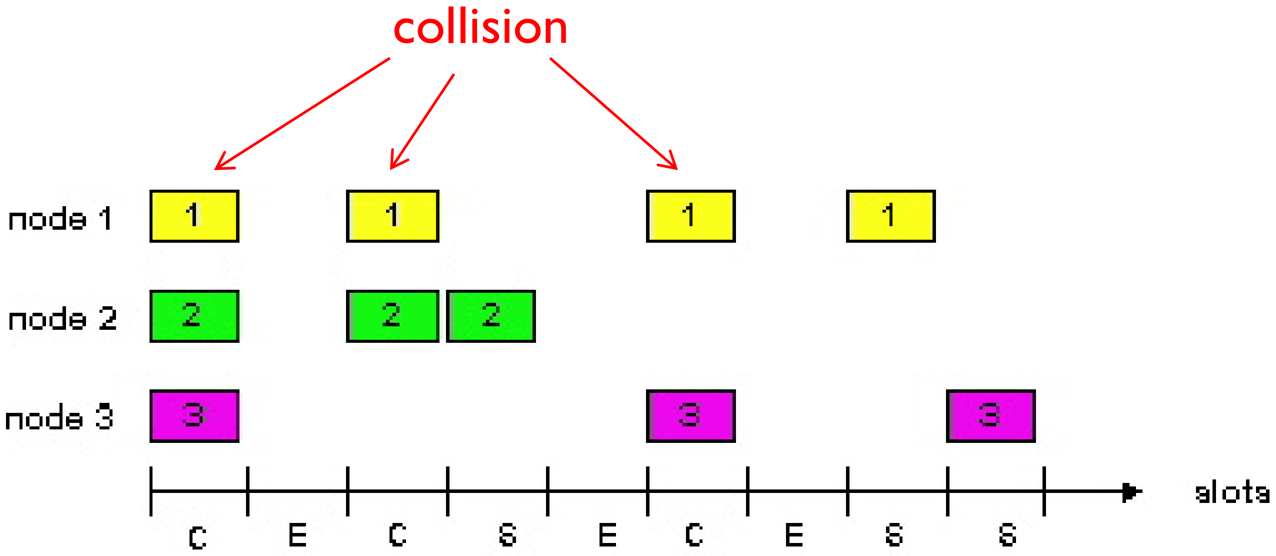
Random Access Protocols

- when a node has a packet to send
  - a transmitting node always transmits at the full rate of the channel, *R* bps
  - if there is a ***collision***,
    - each node involved in the collision ***repeatedly retransmits*** its frame until its frame gets through without a collision
  - usually, when experiences a ***collision***
    - the node ***does not*** retransmit the frame right away
    - instead, it waits a ***random delay*** before retransmitting the frame
    - each node involved in a collision chooses ***independent random delays***
      - random delays are independently chosen, less chance of collision

Slotted ALOHA

- assumptions:
  - all frames consists of exactly *L* bits
  - time is divided into slots of size *L*/*R* seconds
    - a slot equals the time to transmit one frame
  - nodes starts to transmit frames only at the beginning of slots
  - the nodes are synchronized so that each node knows when the slots begins
  - if two or more frames collide in a slot, then all the nodes detect the collision event before the lost ends
- operations:
  - when a node has a fresh frame to send, it waits until the beginning of the next slot and transmits the entire frame in the slot
    - if ***no collision***, the node has successfully transmitted its frame, no retransmission needed
    - if ***collision***, the node detects the collision before the end of the slot
      - the node retransmits its frame in each subsequent slot with ***probability p*** until the frame is transmitted without a collision

Slotted ALOHA



Slotted ALOHA (cont.)

pros

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

cons

- collisions, wasting slots
- idle slots
- clock synchronization