COMP 4759/6777 - Fall 2022

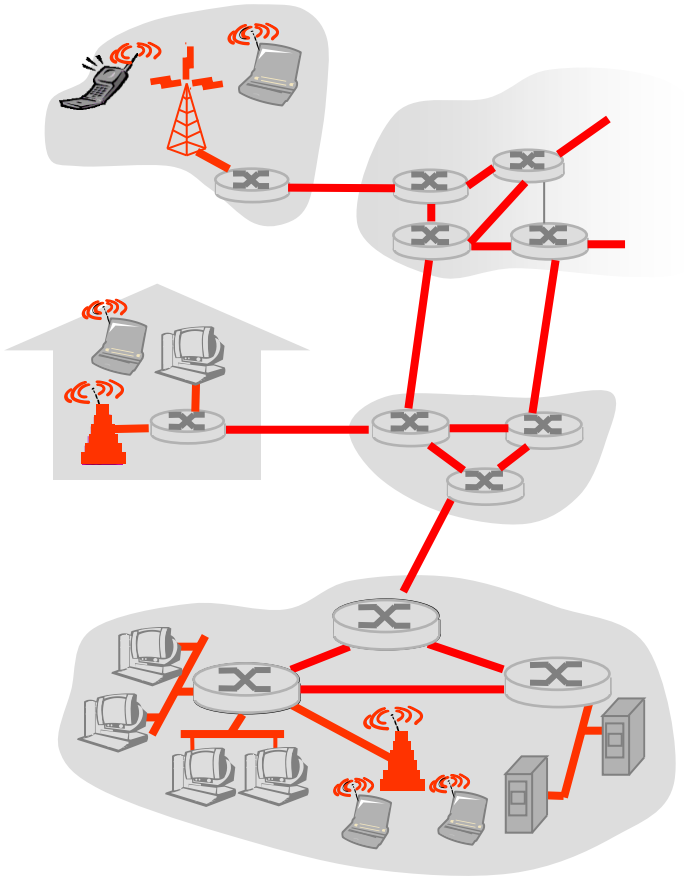
Chapter 3 Data Link Layer

Instructor: Dr. Qiang Ye
Department of Computer Science
Memorial University of Newfoundland
EN-2033
qiang.ye@mun.ca
https://www.cs.mun.ca/~qiangy/

Chapter 3: The Data Link Layer

Our goals:

- understand principles behind data link layer services:
  - error detection and correction (by the receiver)
  - link access by sharing a broadcast channel: multiple access
    - Instruct the hardware (PHY layer) **when** to transmit (… MAC protocols)
  - link layer addressing
  - reliable data transfer, flow control
- implementation of various link layer technologies



Link Layer

3.1 Introduction and services

3.2 Framing

3.3 Error detection and correction

3.4 Retransmission

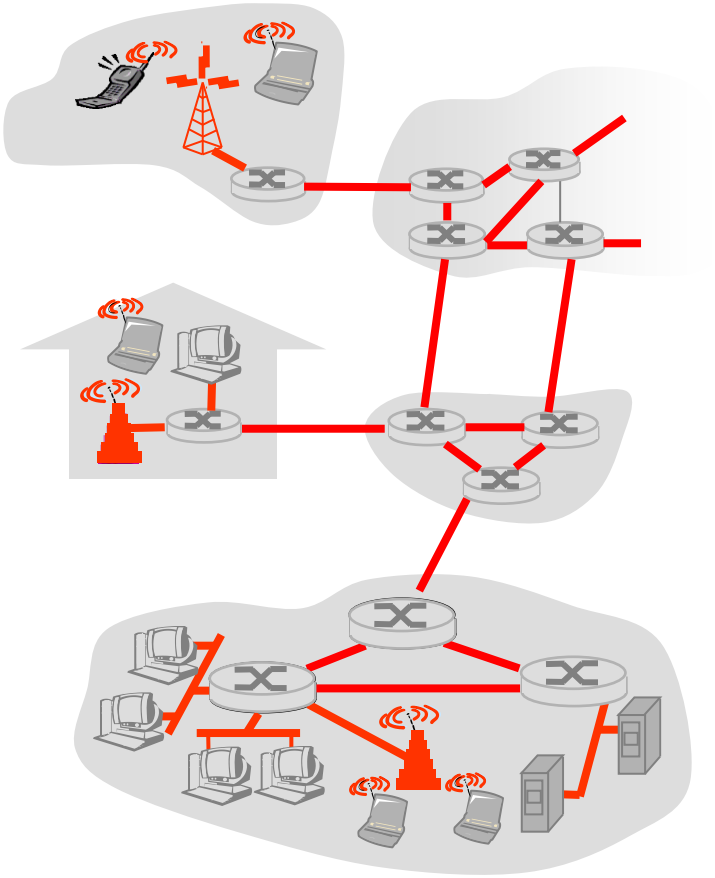
3.5 Multiple access protocols

3.6 Link-layer Addressing

Link Layer: Introduction

Terminology:

- hosts and routers are nodes
- communication channels that connect adjacent nodes are links
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a frame, encapsulates datagram (from network layer)



Link layer: context

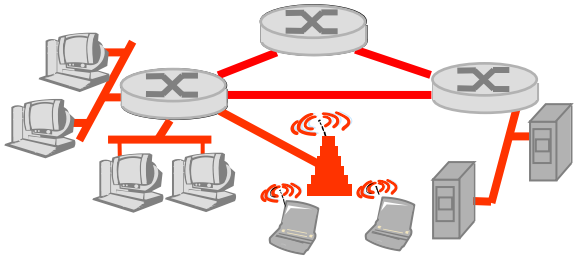
- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, 802.11 on last link

- each link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

Link Layer Services

- framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium (a.k.a. broadcast medium)
  - "MAC" addresses used in frame headers to identify source, destination
    - different from IP address!
- reliable delivery between adjacent nodes
  - seldomly used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q.: Why both link-level and end-to-end reliability?

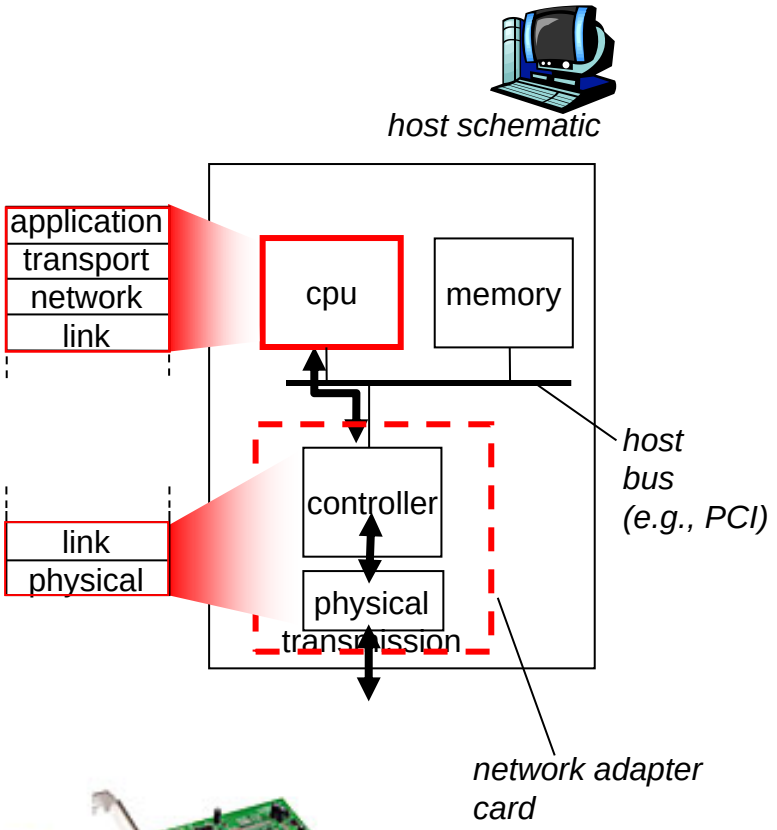
Link Layer Services (more)



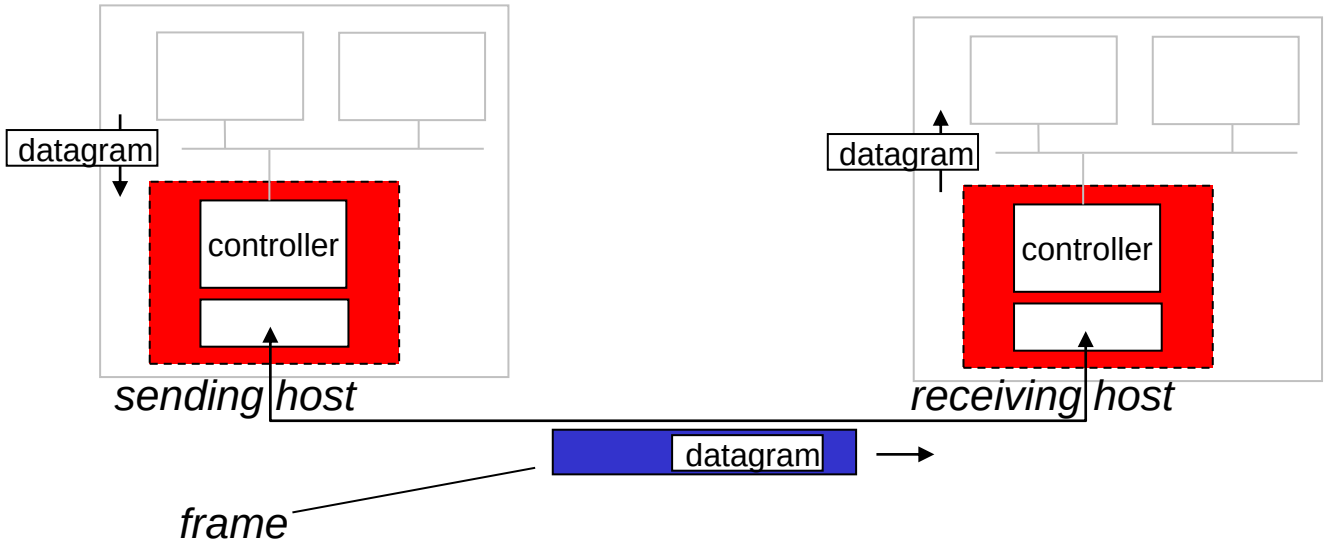
- flow control:
  - pacing between (adjacent) sending and receiving nodes
- error detection:
  - errors caused by signal attenuation and noise
  - receiver detects presence of errors
  - receiver signals sender for retransmission
- error correction:
  - receiver identifies and corrects bit error(s) without resorting to retransmission

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka network interface card NIC)
  - Ethernet card, 802.11 card
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Adaptors Communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, flow control, etc.
- receiving side
  - looks for errors, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Link Layer

3.1 Introduction and services

3.2 Framing

3.3 Error detection and correction

3.4 Retransmission

3.5 Multiple access protocols

3.6 Link-layer Addressing

Framing

- Framing at sender: Large block of data may be broken up into small frames at the sender because:
  - A larger block of data has higher probability of error
  - Only a smaller amount of data needs to be retransmitted
  - On a shared medium, such as Ethernet and wireless LAN, small frame size can prevent one node from occupying medium for long periods

- At receiver:
  - It has to map a bit steam into frames.

Framing

- The link layer needs to:

  - Pack bits into frames, so that each frame is distinguishable from another.

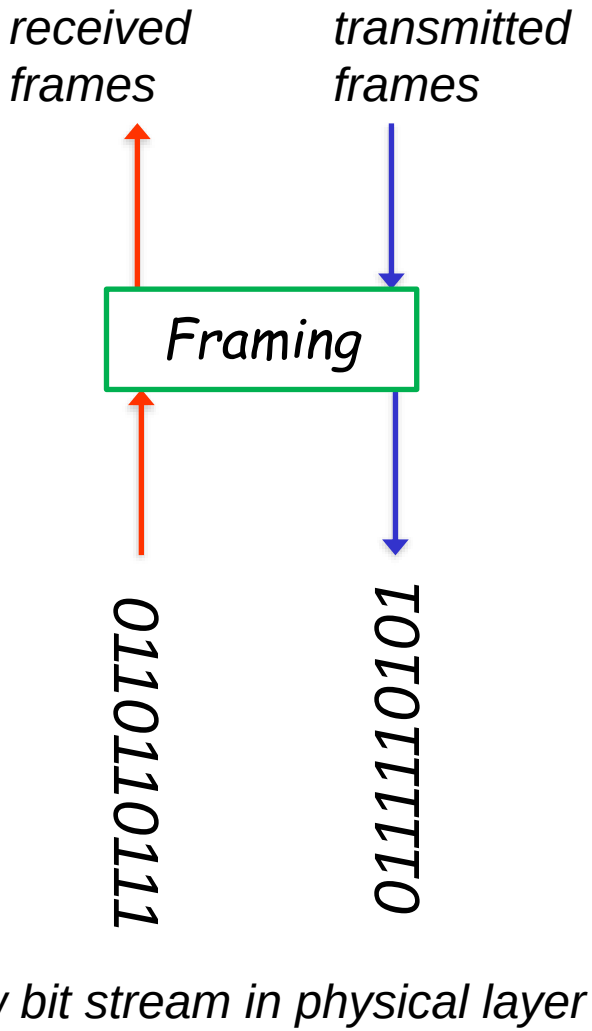
- Two techniques:

  1 Fixed-size framing: no boundaries for frames, size used as the delimiter

  - very difficult to resynchronize after bit loss

2 Variable-size framing: define beginning and end of frames.

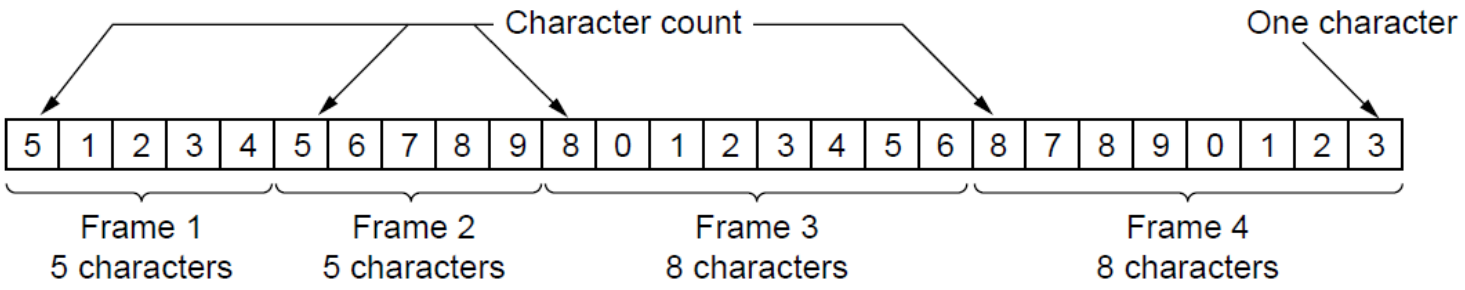
- Frame boundaries can be determined using: character counts or flags



raw bit stream in physical layer

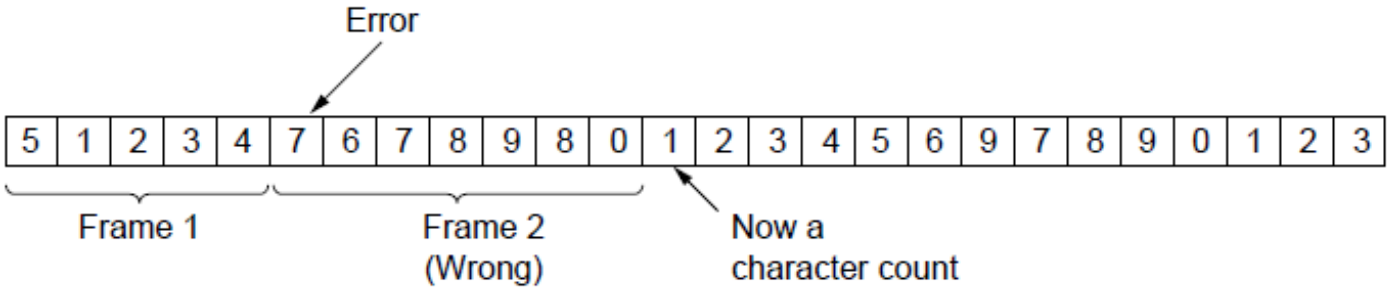
Byte Count Framing

- Field in the header specifies the number of bytes in the frame. Once the header information is received it is used to determine end of the frame (synchronization).

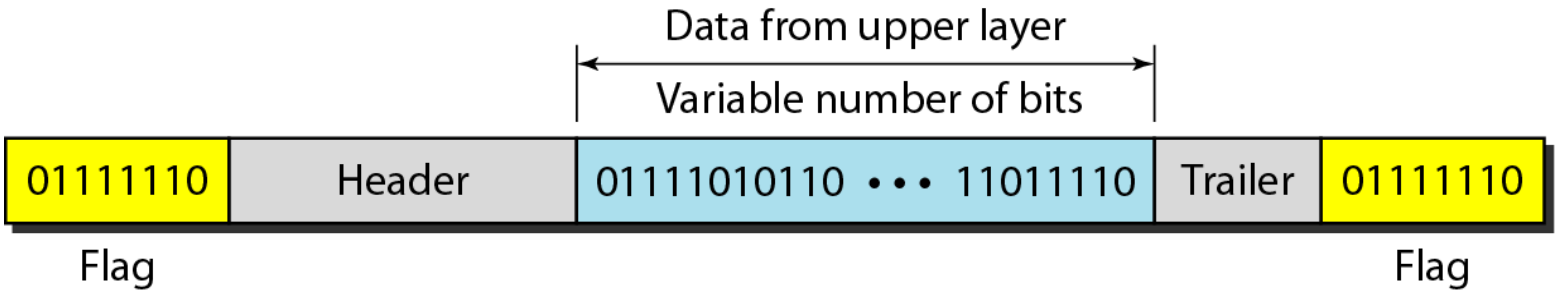


- Problem occurs when the count is lost or in error

Very difficult to resynchronize after count loss/error - the receiver cannot figure out where the next frame starts



Flag bits with bit stuffing



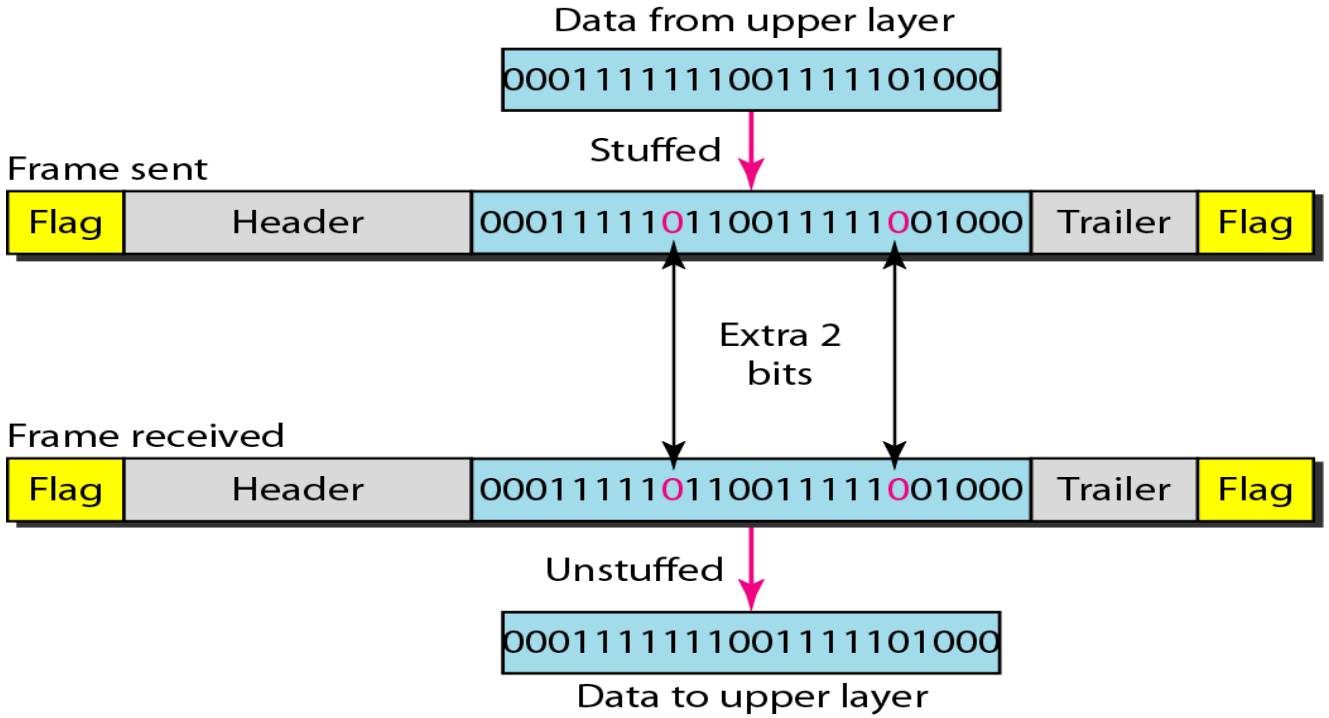
- Frame delineated by flag character; Each frame begins and ends with a special bit pattern: flag byte 01111110
- Uses bit stuffing to prevent occurrence of flag 01111110 inside the frame
- Transmitter inserts extra 0 after each consecutive five 1s inside the frame
- Receiver checks for five consecutive 1s

  If next bit = 0, it is passed

  If next two bits are 10, then flag is detected

  If next two bits are 11, then frame has errors

Bit stuffing and unstuffing



- Bit stuffing is the process of adding extra bits to ensure that flag sequence does not appear in the data, so that the receiver does not create wrong frames. Bit unstuffing is the process of removing these extra bits at the receiver.
- If the receiver ever loses synchronization, it can just search for the flag byte.

Link Layer

3.1 Introduction and services

3.2 Framing

3.3 Error detection and correction

3.4 Retransmission

3.5 Multiple access protocols

3.6 Link-layer Addressing

Error Control (detection and correction)

- Digital transmission systems introduce errors
- Errors occur due to noise or interference on a communication channel, e.g., Bit Error Rate (BER) = 10^{-6}
- Applications require certain reliability level

  Data applications require **error-free transfer**

  Voice & video applications **tolerate some errors**

- Error control ensures that a data stream is transmitted to a certain level of accuracy despite errors
- Two basic strategies to deal with errors:

  Error detection: detect errors in transmission

  Error correction: fix errors

Error Control

- Error-correcting: can correct the errors, but they generate too large transmission overhead.
- Error-detecting: the goal is to only detect the errors with the minimal transmission overhead. **When error is detected, the data is retransmitted.**
- Error-detecting may be enough when the channel BER is small such as copper wire or fiber.
- Error-correcting codes are widely used on wireless links that are noisy.
- Larger redundant bits yield better detection and correction, but more transmission overhead: trade-off

Error types: single or burst

- **Single-bit error**: Only 1 bit of a given data unit is changed from 1 to 0 or from 0 to 1.
- A **burst error** means that two or more bits in the data unit have changed - not necessarily consecutive bits
- Burst error is most likely to happen since the duration of noise is normally longer than the duration of a bit. The number of bits affected depends on the data rate and duration of noise.
- Example:

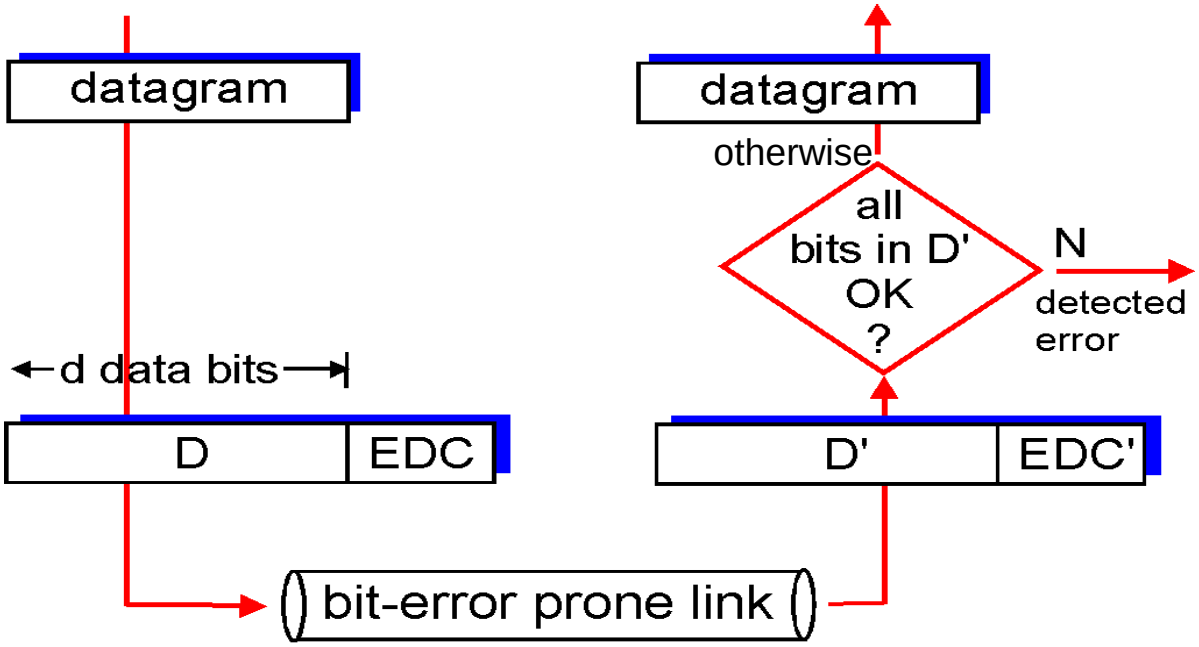
  If data is sent at rate = 1Kbps then a noise of 1/100 sec can affect 10 bits (1/100*1000)

  If same data is sent at rate = 1Mbps then a noise of 1/100 sec can affect 10,000 bits ($1/100*10^6$)

Error Control (detection and correction)

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Error detection schemes

1- Parity checking: very simple and easy error detection

a) Single bit parity

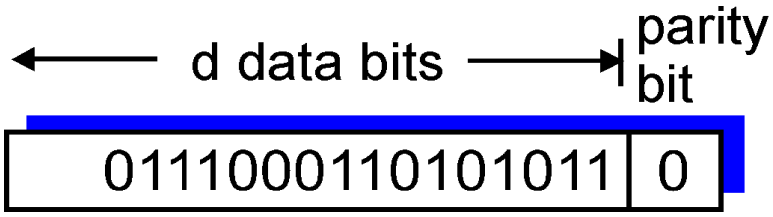
b) Two-dimensional bit parity

2- Cyclic Redundancy Checks (CRC)

more complex, but very effective and efficient

- General idea: Suppose that each frame has m data bits, we add r redundant bits, so that the total length is n bits (n = m + r)
- At the receiver: the r redundant bits are used to detect errors.
- The goal: to maximize the probability of detecting errors using only a small number of redundant bits.

a) Single Parity Check



- Add a single extra bit (known as the parity bit) to a group of data bits.

•**Odd parity:** Set the parity bit to 1 or 0 to make the total # of 1's an odd number.

•**Even parity:** Set the parity bit to 1 or 0 to make the total # of 1's an even number

E.g.,: data bits 0011001; add parity bit 1 --->00110011

- Receiver checks to see if number of "1" is even
- All error patterns that change an odd number of bits are detectable

Single Parity Check Code

- Redundancy: adds 1 redundant bit per k information bits: overhead = 1/(k + 1)

- Coverage:

  - All single-bit errors can be detected

  - For burst errors: all error patterns with odd number of errors can be detected - 50% of error patterns can be detected

If single error in bit 3 : (0, 1, **1**, 1, 1, 0, 0, 1)

Number of 1's =5, odd ---→ Error detected

If errors in bits 3 and 5: (0, 1, **1**, 1, **0**, 0, 0, 1)

Number of 1's = 4, even ---→ Error not detected

Single Parity Check code

- For a group of N_b bits and bit error rate P_b (probability of a bit error)

P_1 = P(no errors) = $(1 - P_b)^{N_b}$

P_2 = P(undetected error), for even parity, even number errors goes undetected; so P_2 is P(even number of errors)

$$P_2 = \sum_{i=1}^{N_b/2} \binom{N_b}{2i} P_b^{2i} (1-P_b)^{N_b-2i}$$

P_3 : P(detected error) = 1- P_2 - P_1

Frame error probability

Given a frame, sent as a sequence of n bytes, each with a parity check. What are Pf_1, Pf_2, and Pf_3?

Pf_1 = probability of no error in the whole frame = probability of zero bytes with errors = $(P_1)^n$

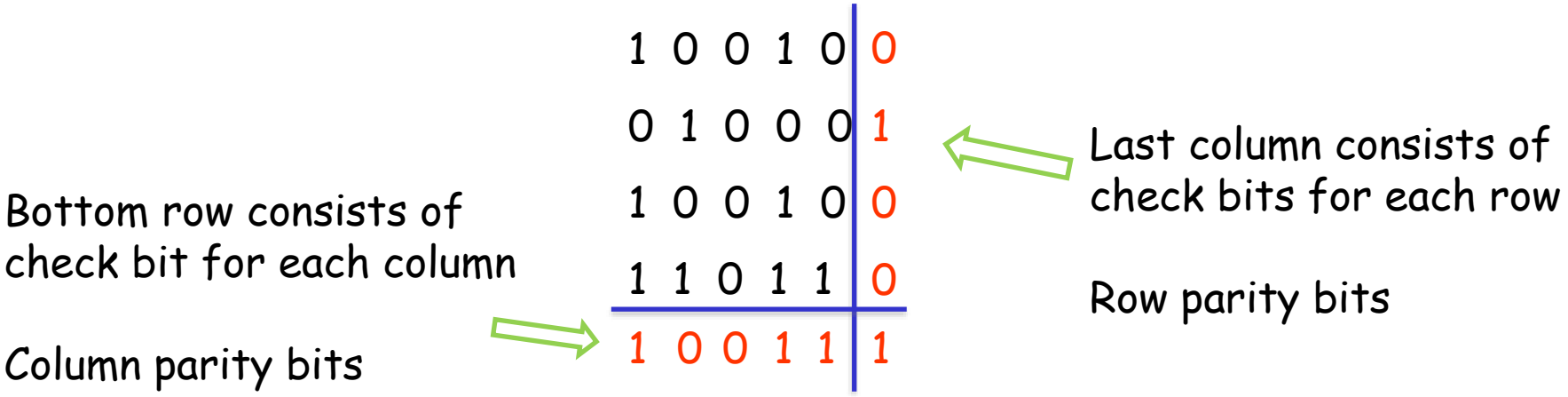
Pf_2 = probability of an undetected error = at least one of the n bytes has undetected error.

$$= \sum_{i=1}^{n} \binom{n}{i} P_2^i (1-P_2)^{n-i}$$

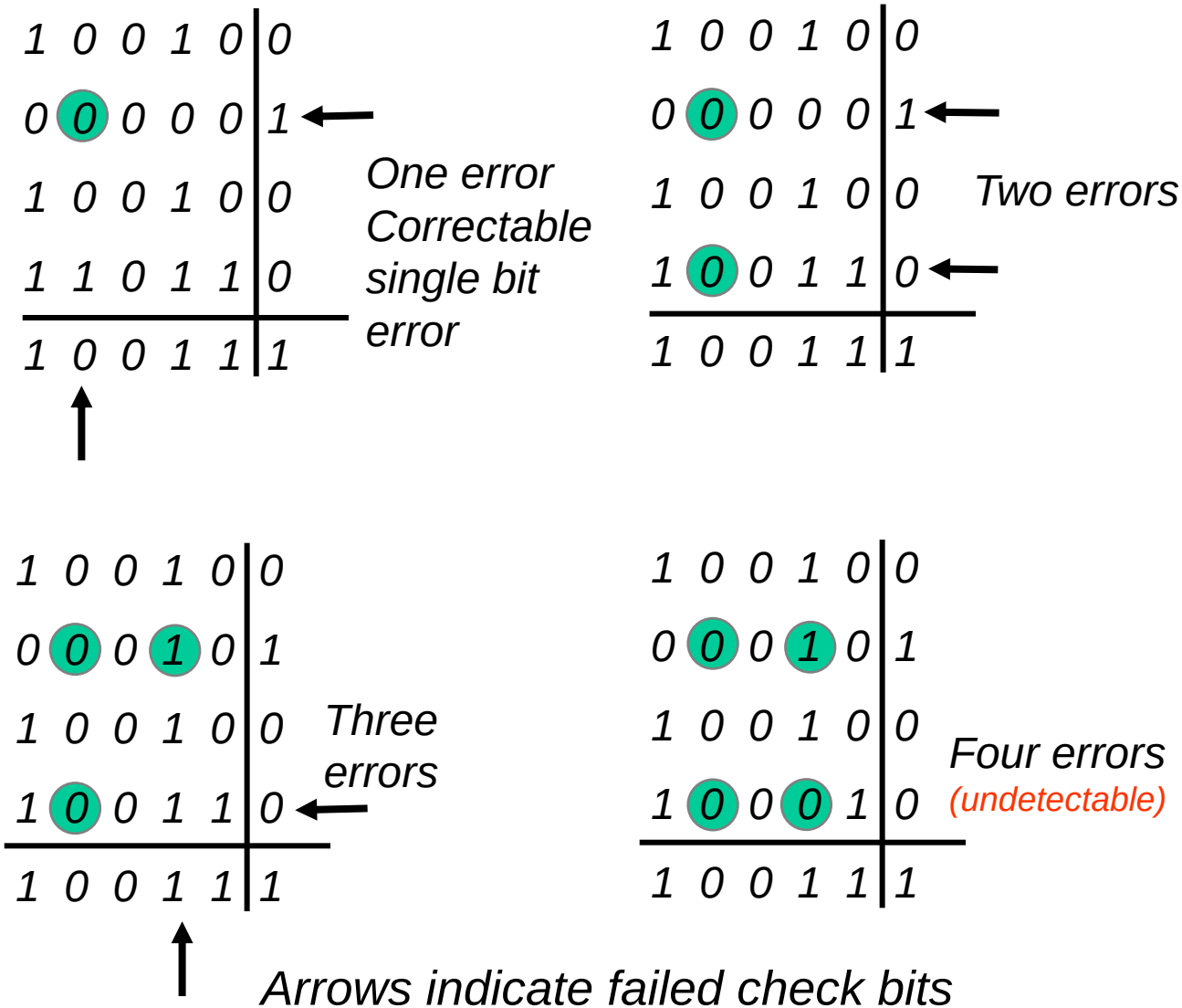
Pf_3 = probability of a detected error = $1 - Pf_1 - Pf_2$

b) Two-Dimensional Parity Check

- More parity bits to enhance the error detection capability
- Arrange information as columns, e.g., m columns and n rows: number of bits = mn
- Compute m + n + 1 parity bits
- One parity bit for each row, one for each column, and one for the whole message
- Total number of transmitted bits = mn + m + n + 1 bits



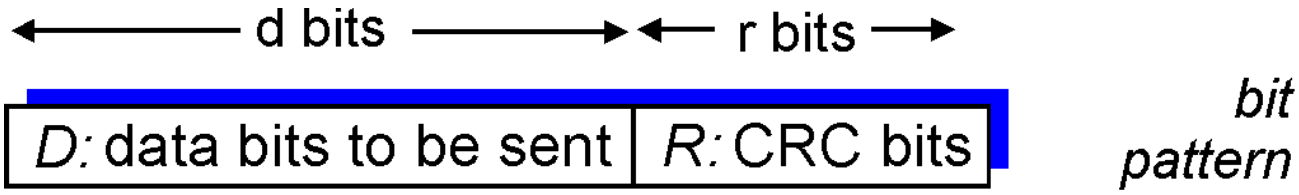
b) Two-Dimensional Parity Check



2- Cyclic Redundancy Check (CRC)

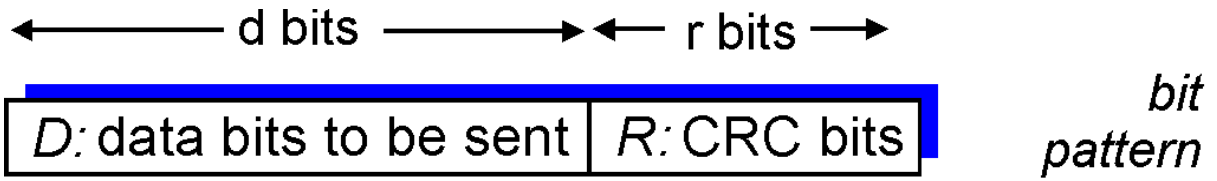
- Single parity check codes do not detect enough errors
- Two-dimensional codes require too many check bits
- CRC is widely used in practice (Ethernet, 802.11 WiFi, etc)
- Choose r+1 bit pattern generator (G) (degree is r)
- **goal:** generate r CRC bits, denoted by R, such that

  <D, R> exactly divisible by G using modulo 2 arithmetic.
- Receiver knows G, divides <D, R> by G. **If non-zero remainder: error detected.**



Cyclic Redundancy Check (CRC)

- view data bits, D, as a binary number
- choose r+1 bit generator G (leftmost and rightmost bits are both 1)
- Append r bit "0" to D
- Then, Divide it by G. The remainder is CRC code R (r bits).
- Send <D,R> codes to the receiver
- The receiver divides the received bit sequence by G
- If non-zero remainder: error detected!
- can detect all burst errors less than r+1 bits



$D * 2^r$ XOR R

mathematical formula

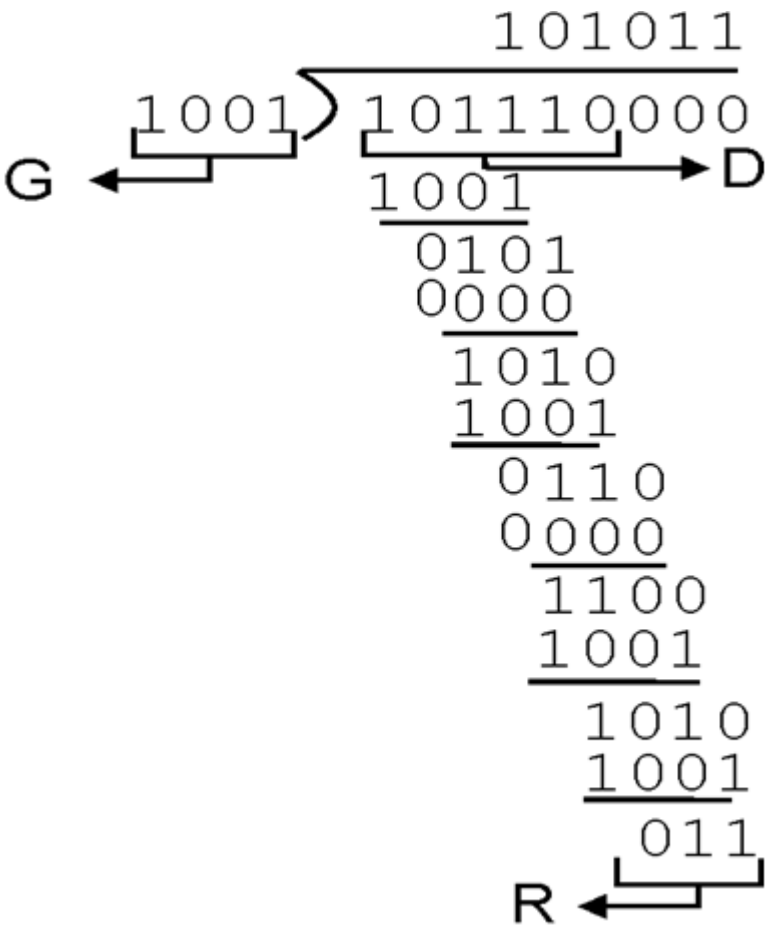
CRC Example:

D = 101110 G = 1001

- All CRC calculations are done in modulo-2 arithmetic without carries in addition or borrows in subtraction, which is equivalent to bitwise exclusive-or (XOR)

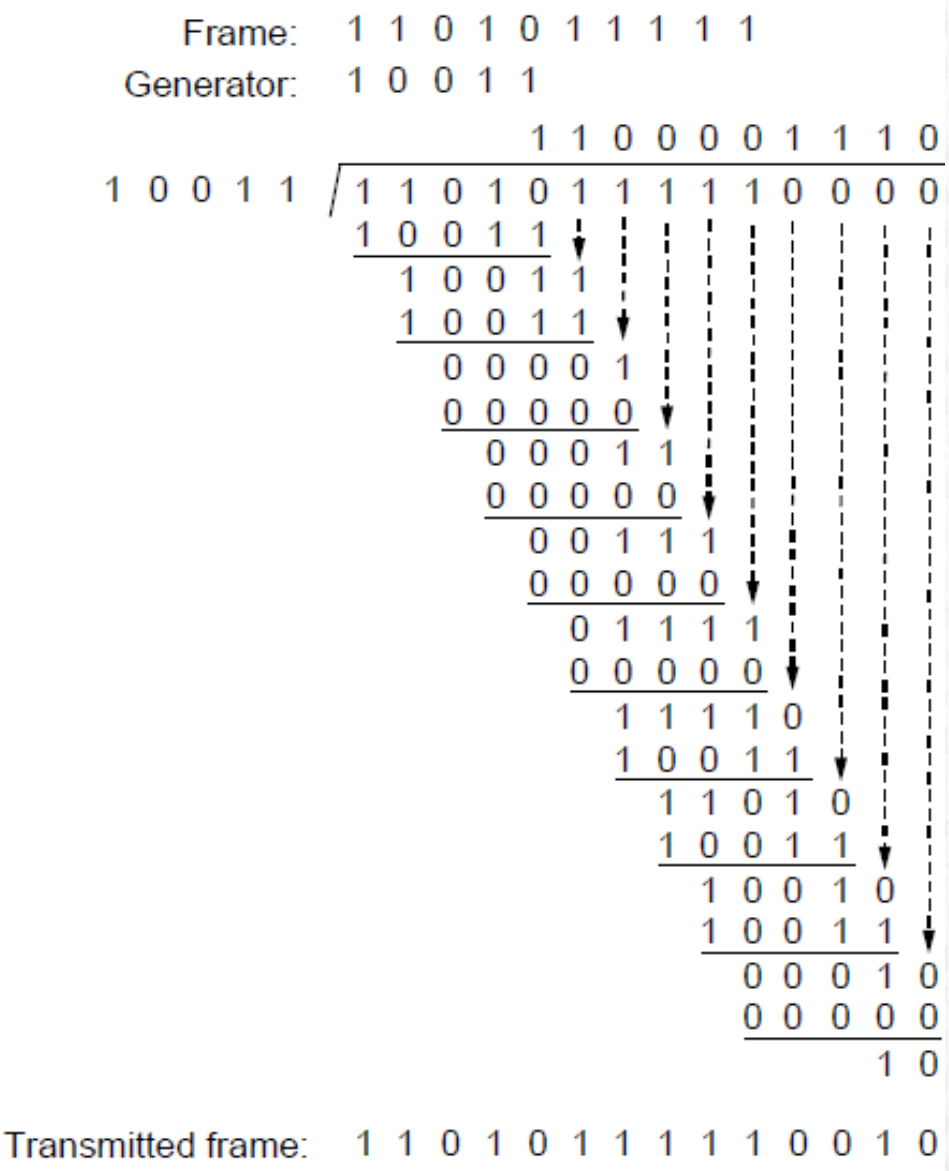
1 XOR 1 = 0	0 XOR 0 = 0
1 XOR 0 = 1	0 XOR 1 = 1

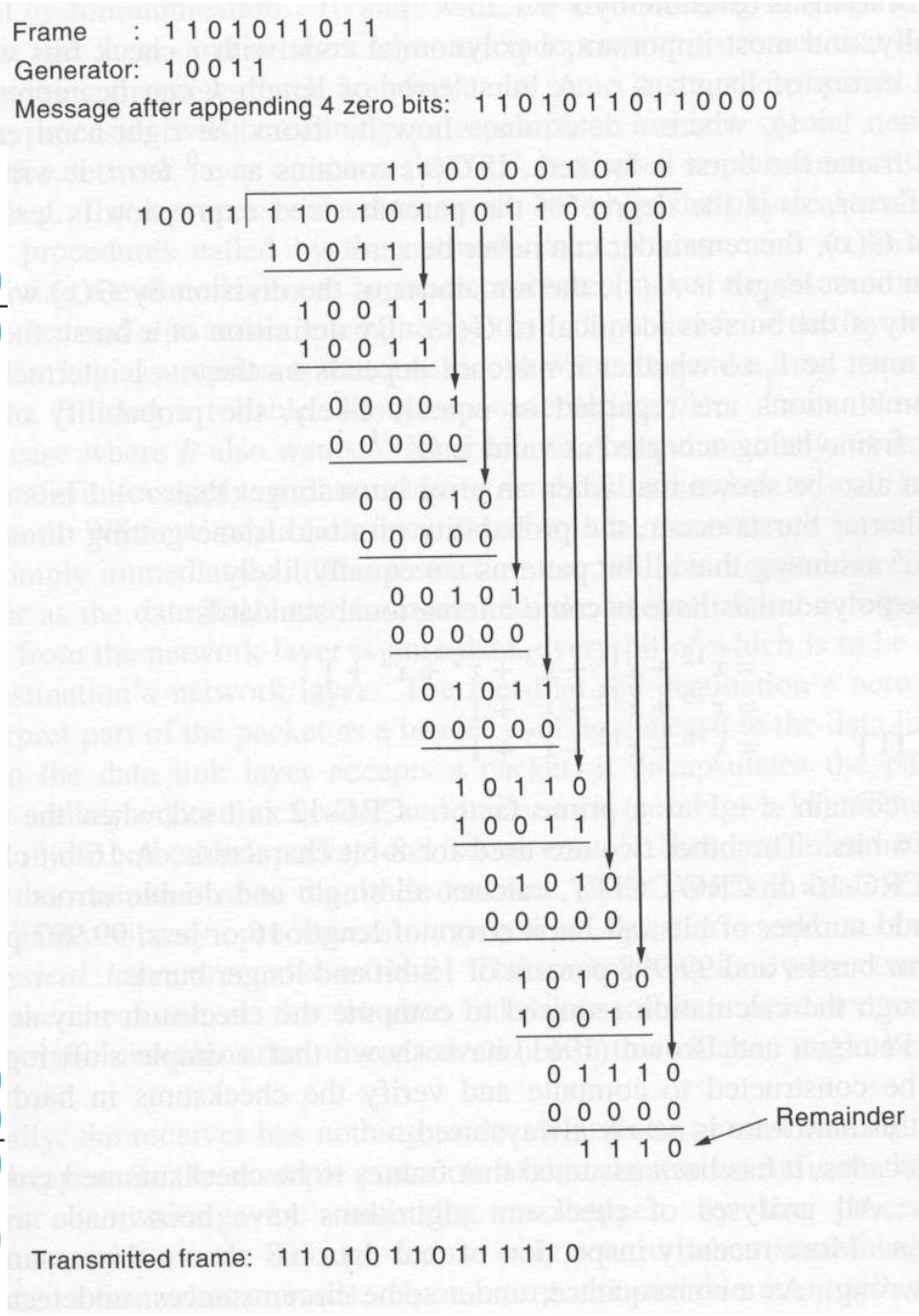
- The leading zero of the remainder (for each step) is dropped off
- When the leftmost bit of the remainder is zero, use 0000 instead of the original divisor.



Transmitted packet: 1 0 1 1 1 0 0 1 1

CRC Example:


Frame: 1 1 0 1 0 1 1 1 1 1
Generator: 1 0 0 1 1
1 1 0 0 0 0 1 1 1 0
1 0 0 1 1 | 1 1 0 1 0 1 1 1 1 1 0 0 0 0
1 0 0 1 1
1 0 0 1 1
1 0 0 1 1
0 0 0 0 1
0 0 0 0 0
0 0 0 1 1
0 0 0 0 0
0 0 1 1 1
0 0 0 0 0
0 1 1 1 1
0 0 0 0 0
1 1 1 1 0
1 0 0 1 1
1 1 0 1 0
1 0 0 1 1
1 0 0 1 0
1 0 0 1 1
0 0 0 1 0
0 0 0 0 0
1 0
Transmitted frame: 1 1 0 1 0 1 1 1 1 1 0 0 1 0


Frame : 1 1 0 1 0 1 1 0 1 1
Generator: 1 0 0 1 1
Message after appending 4 zero bits: 1 1 0 1 0 1 1 0 1 1 0 0 0 0
1 1 0 0 0 0 1 0 1 0
1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
1 0 0 1 1
1 0 0 1 1
1 0 0 1 1
0 0 0 0 1
0 0 0 0 0
0 0 0 1 0
0 0 0 0 0
0 0 1 0 1
0 0 0 0 0
0 1 0 1 1
0 0 0 0 0
1 0 1 1 0
1 0 0 1 1
0 1 0 1 0
0 0 0 0 0
1 0 1 0 0
1 0 0 1 1
0 1 1 1 0
0 0 0 0 0
Remainder
1 1 1 0
Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 1 0

Cyclic Redundancy Check (CRC)

- CRC is also called polynomial codes because we treat a k-bit frame as polynomial with coefficients of 0 and 1 only with terms from x^{k-1} to x^0

  e.g., 110001 : $1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^5 + x^4 + 1$

- Represent the data bits as a polynomial:

$$\mathrm{D}(x) = \sum_{i=0}^{n-1} d_i x^i$$

- choose r+1 bit pattern (generator), G (leftmost and rightmost bits are both 1), viewed again as polynomial

$$G(x) = \sum_{i=0}^{r} g_i x^i$$

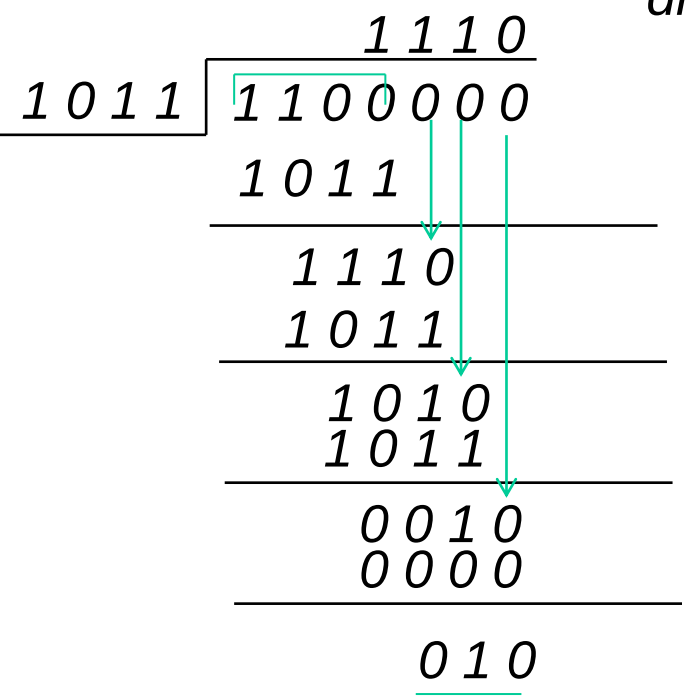
- choose r CRC bits, denoted by R, such that

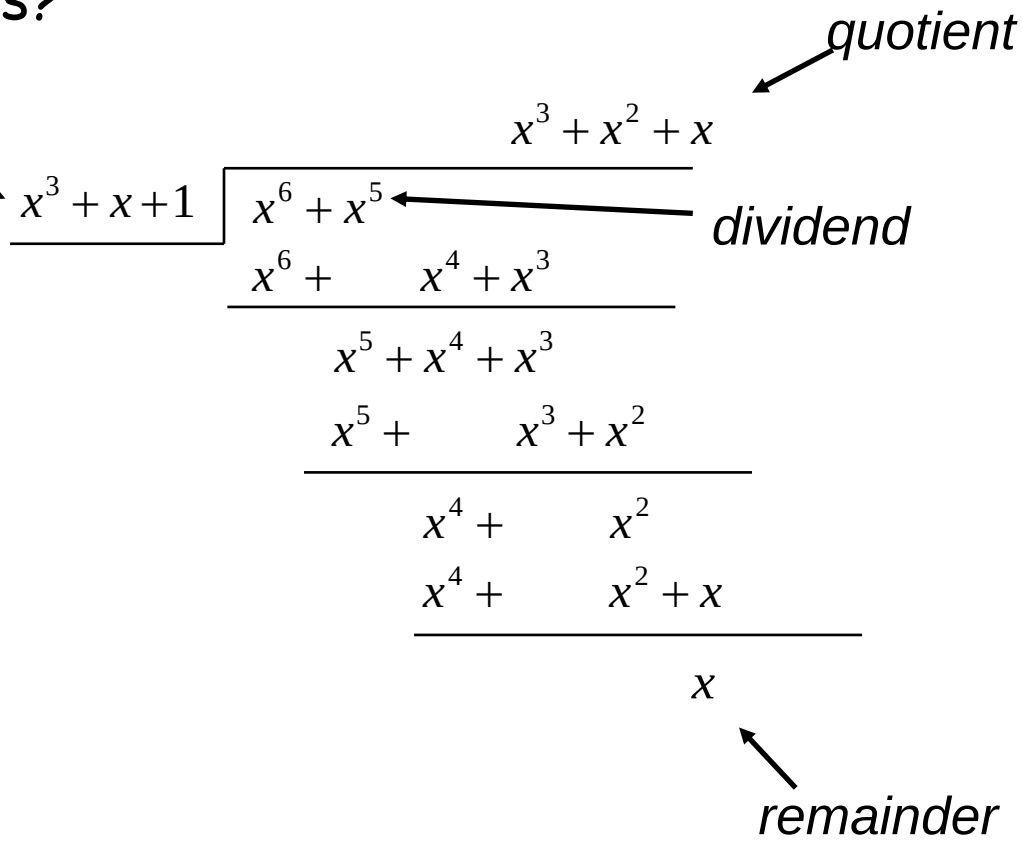
$$\mathrm{D}(x)x^r + R(x) = G(x)H(x)$$

Addition of the polynomial coefficients is modulo 2 arithmetic.

Cyclic Redundancy Check (CRC)

- The data is 1100 and the generator polynomial $G(x) = x^3 + x + 1 = 1011$. Find CRC bits?





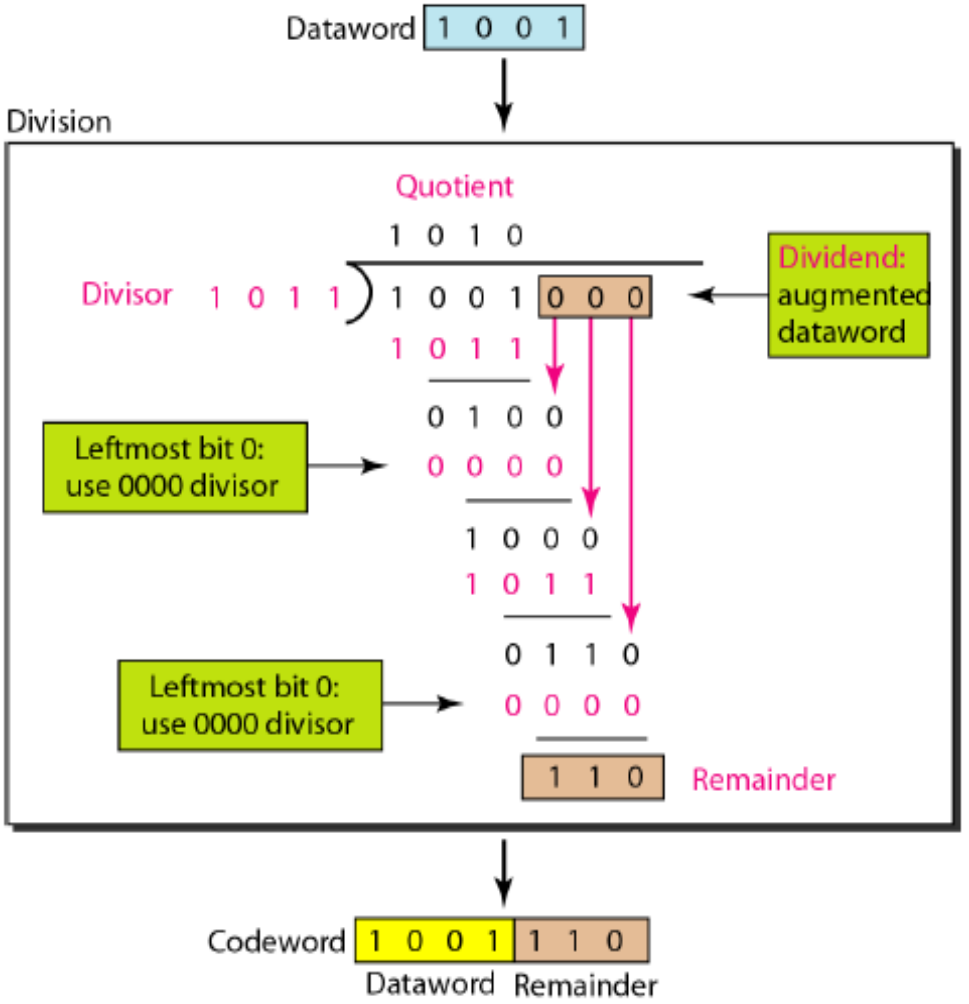
Cyclic Redundancy Check (CRC)

At the receiver:

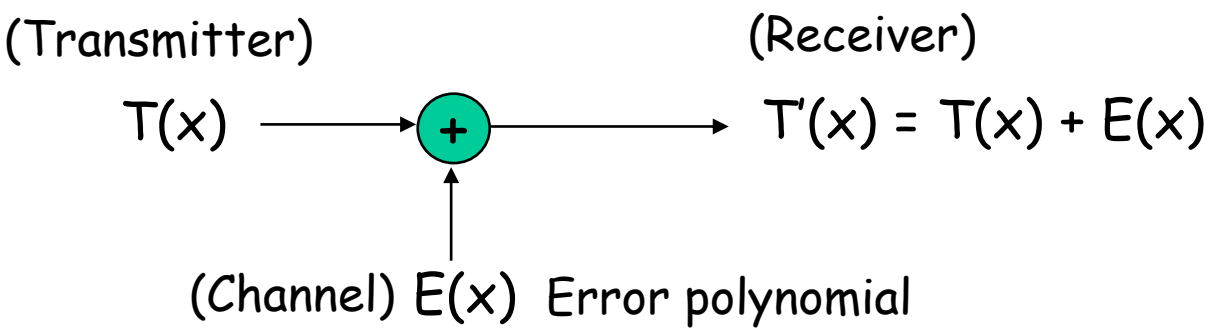
```
          1 1 1 0
      ------------
1 0 1 1 | 1 1 0 0 0 1 0
          1 0 1 1
          ---------
            1 1 1 0
            1 0 1 1
            -------
              1 0 1 1
              1 0 1 1
              -------
                0 0 0 0
                0 0 0 0
                -------
                  0 0 0
```

No error

Another example:



Undetectable Error Patterns



- E(x) has 1s in error locations & 0s elsewhere (in polynomial form)
- Receiver divides the received polynomial T'(x) by G(x)
- **Blindspot**: If E(x) is a multiple of G(x), then

$$T'(x) = T(x) + E(x) = q(x)G(x) + q'(x)G(x)$$

- The set of undetectable error polynomials is the set of errors containing G(x) as a factor - all other errors will be caught.
- Choose the generator polynomial so that selected error patterns can be detected

Cyclic Redundancy Check (CRC)

1. CRC can detect all single bit errors

$E(x) = X^i$, $E(x)/G(x)$ will not be zero, since $G(x)$ must have the first and last bit equal to 1, i.e., $G(x)$ has more than one nonzero term

2. CRC can detect burst error of length k <=r with a generator bit length r+1 bits. $E(x) = X^{k-1} + \ldots + 1$. If k <= degree of $G(x)$ (i.e., k <= r) , then $G(x)$ can never divide $E(x)$

3. CRC can detect all double-bit errors, $E(x) = X^i + X^j = X^j (X^{i-j} + 1)$

X^i is not divisible by $G(x)$. If $G(x)$ does not divide $X^{i-j} + 1$, then the resulting CRC can detect all double errors

Cyclic Redundancy Check (CRC)

- Used Generators:

  CRC-12: $G(X)=x^{12} + x^{11} + x^3 + x + 1$

  CRC-16: $G(X) = x^{16} + x^{15} + x^2 + 1$

- 16-bit CRC will detect:

  1. all 1 and 2 bit errors

  2. all error bursts of up to 16 bits in length

  3. all bursts affecting an odd number of bits

  4. 99.997% of 17 bit error bursts

  5. 99.998% of 18 and longer bursts

- CRC can be implemented efficiently in hardware - computation done bit by bit - shift register, XOR gates

Link Layer

3.1 Introduction and services

3.2 Framing

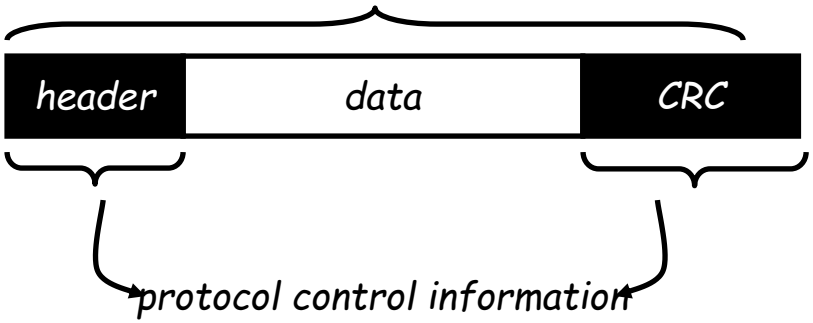
3.3 Error detection and correction

3.4 Retransmission

3.5 Multiple access protocols

3.6 Link-layer Addressing

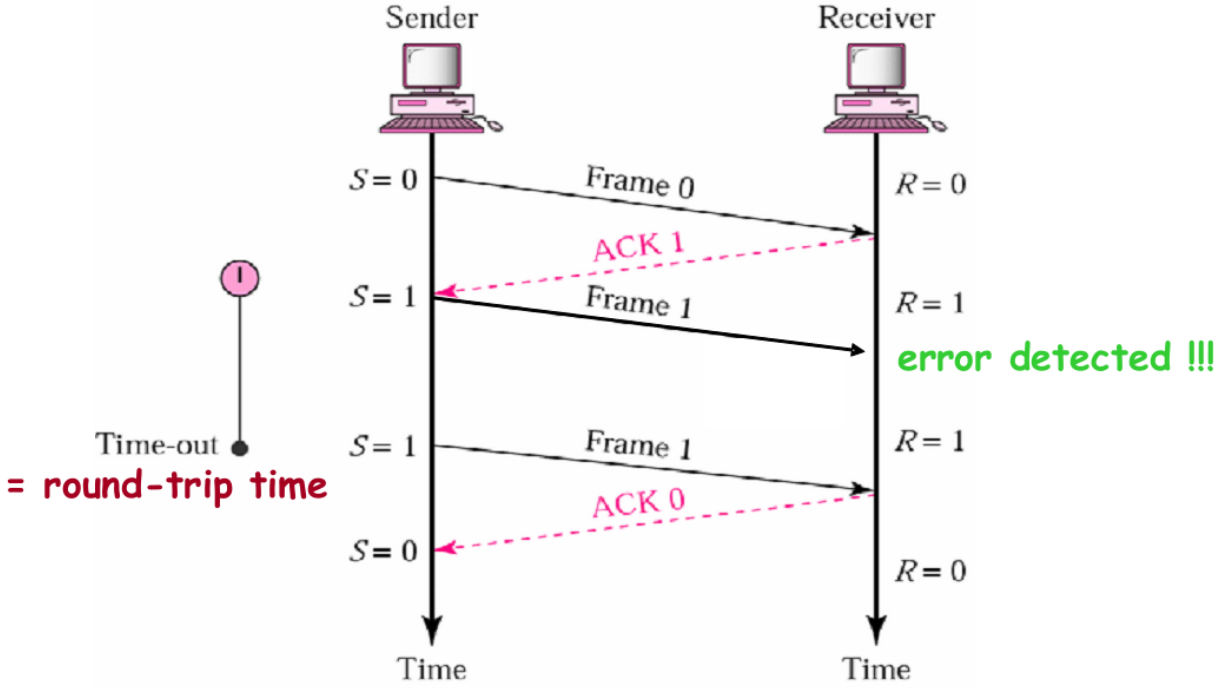
Retransmission and flow control protocols



- **Retransmission scheme: Automatic Repeat Request (ARQ)**:
  - stop-and-wait (also called **alternate bit protocol (ABP)**, stop-and-go)
  - go-back-N
  - selective-retransmit (selective acknowledgement)
- Limited resources at end-systems: **flow control** integrated with ARQ
- **An ARQ protocol should deliver data to the receiver without error, in the right order, without any duplicate**

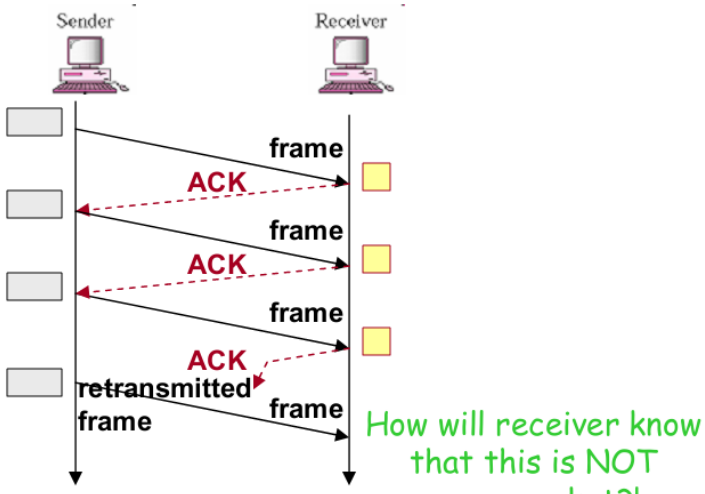
Stop-and-wait [1]

- sender sends an information frame to receiver
- sender then stops and waits for an ACK
- if no ACK arrives within time-out, sender will resend the frame, and again stop and wait
- **time-out period > roundtrip time**

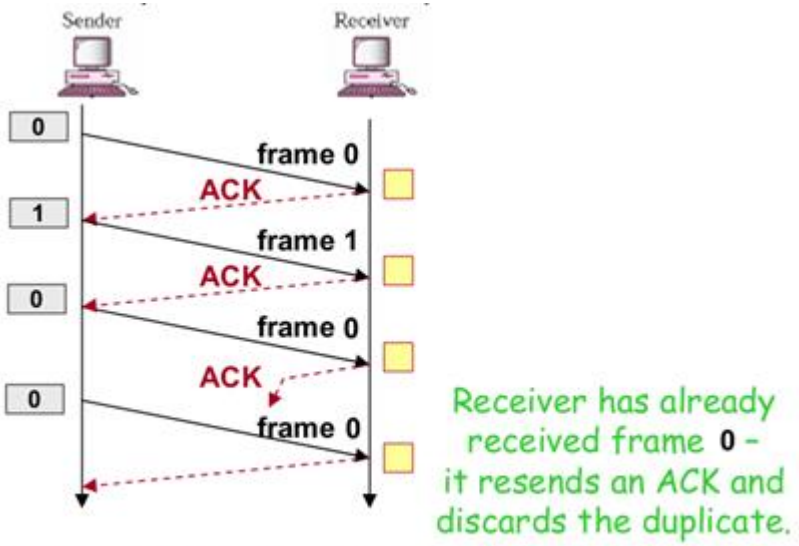


Stop-and-wait [2]

- frame received correctly, but ACK undergoes loss
  - after time-out period, sender resends frame
  - receiver receives the same frame twice
- frames must be numbered so that receiver can recognize and discard duplicate frames
  - sequence # (SN) are included in packet header (SN modulo by 2)
    - only 1 bit required ("0" and "1")



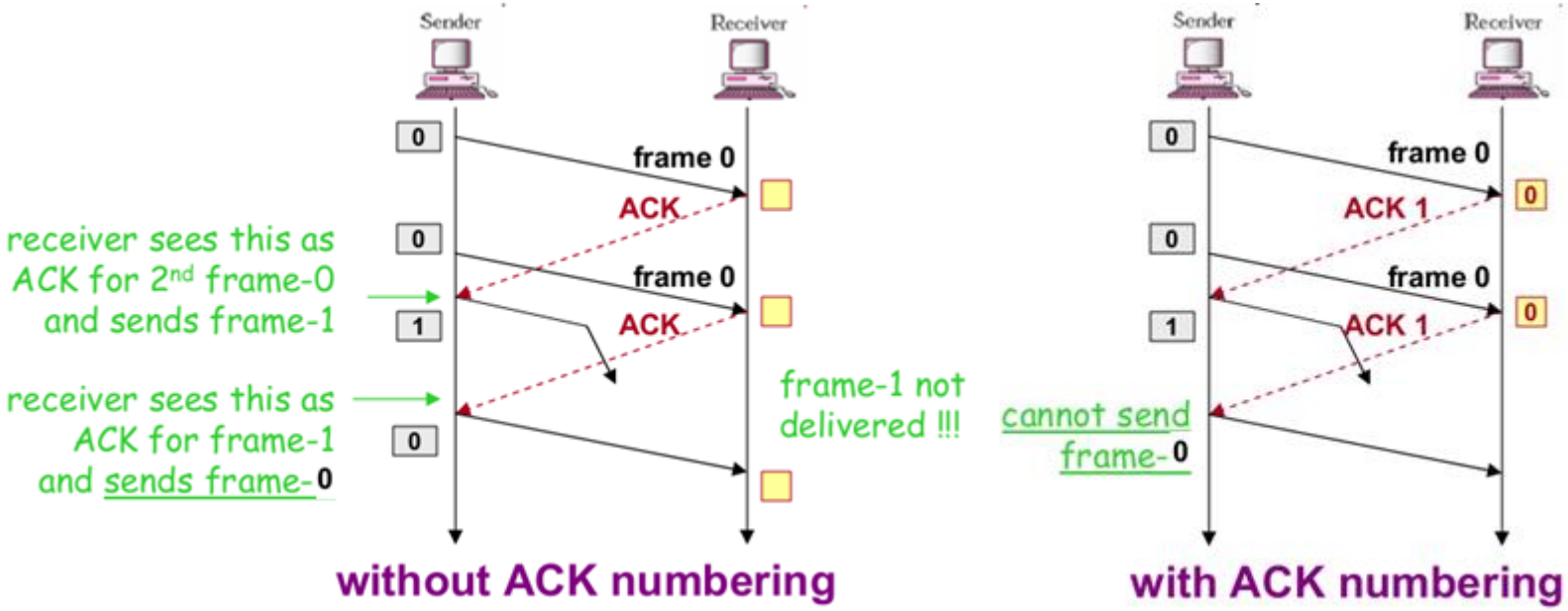
without packet numbering



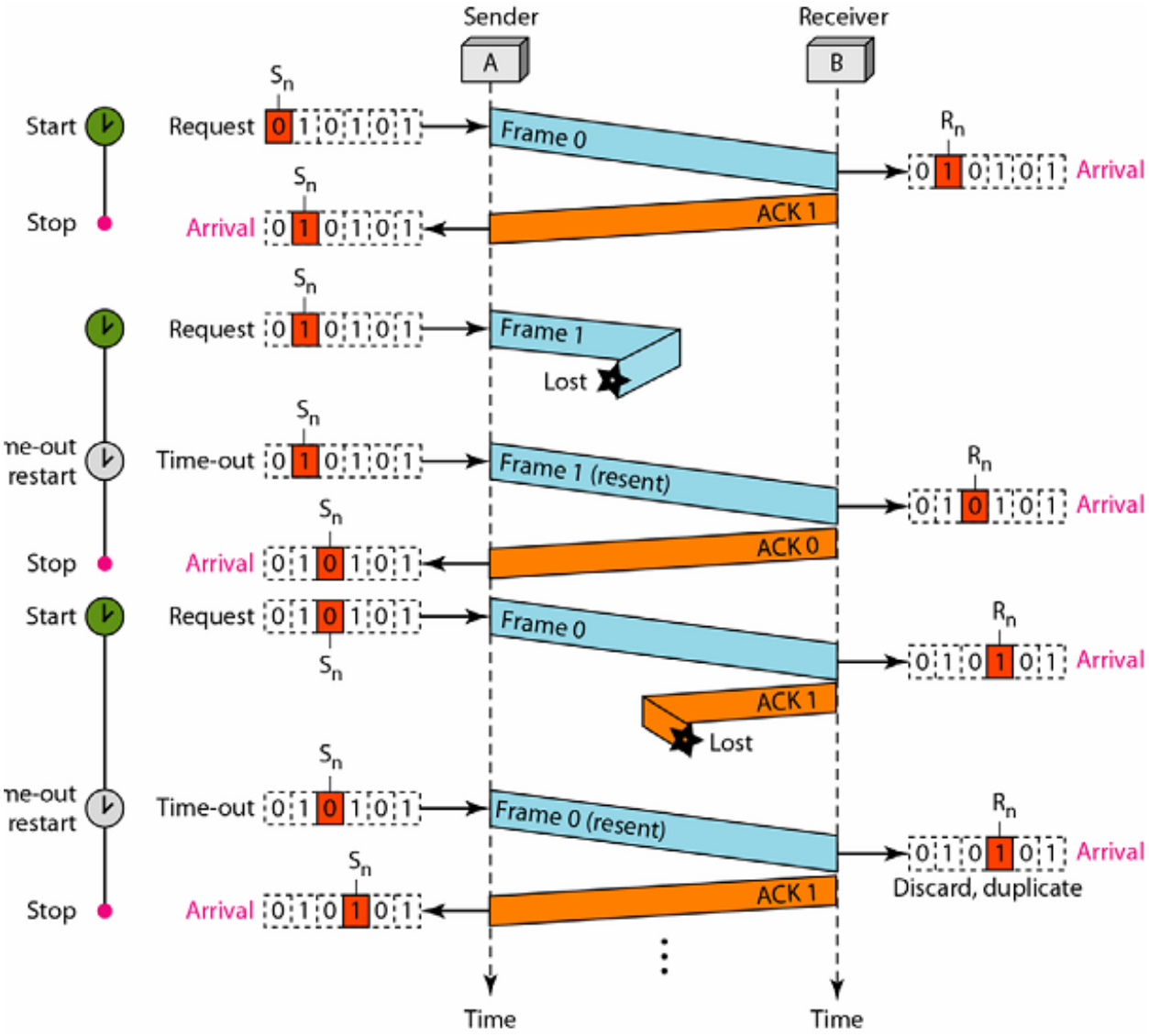
with packet numbering

Stop-and-wait [3]

- ACKs can be delayed due to problems with links or network congestion
  - time-out expires earlier, sender resends frame
  - when delayed ACK arrives, sender assumes that given ACK is for the last frame sent
- ACKs must be numbered to prevent gaps in delivered packet sequence (1 bit required)



Stop-and-Wait [4]



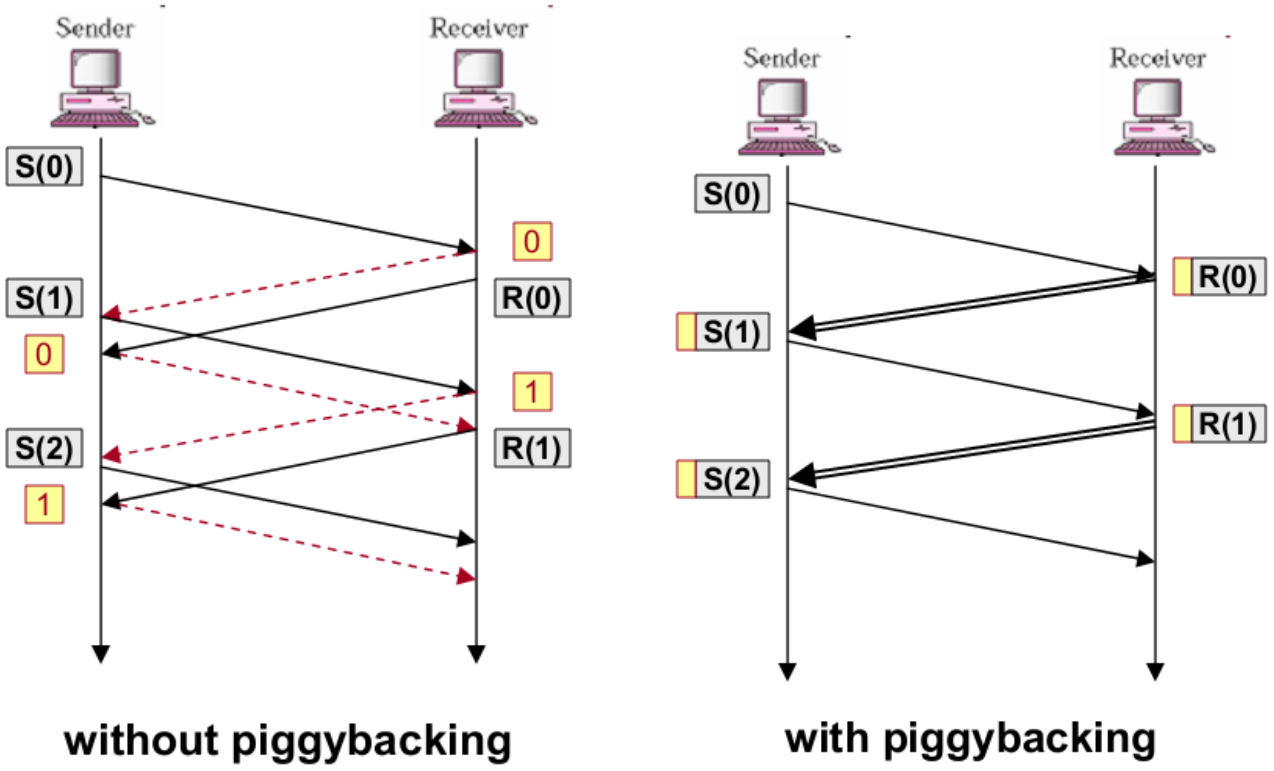
Stop-and-Wait [5]

In summary:

- A frame containing a packet or an ACK can be corrupted, lost or delayed.
- An ARQ protocol needs to function correctly:
  - ACKs (indicates the next expected packet SN)
  - Timeout
  - Sequence number for the packets **and** for the ACK (how many bits?)
- In the case of this protocol, 2 types of frames:
  - Frames from Tx to Rx containing data
  - Frames from Rx to Tx containing ACKs

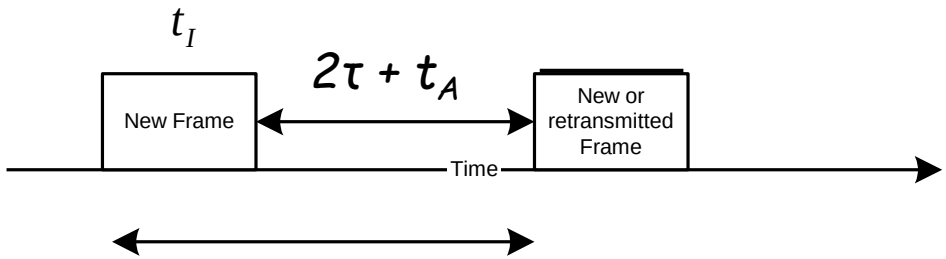
Stop-and-Wait [6]

- In two-way communication, both parties send **data and acknowledgement**
- Piggybacking method: outstanding ACKs are placed in the header of information frames
- Piggybacking can save bandwidth since the overhead from a data frame and an ACK frame (addresses, CRC, etc) can be combined into just one frame



Throughput Analysis of Stop and Wait

- Throughput = # of successful packets received/sec (unit of packet/sec, bit/sec etc.)
- **Assumptions**:
  - Fixed size packets L
  - No errors
  - Infinite retransmissions



Round trip delay: $t_T = t_I + 2\tau + t_A$

- **Goal**: Determine the maximum throughput λ_{max} when A transmits to B.
- **Assumption**: Transmitter A always has something to send. We need this assumption to get λ_{max} .

If no errors:

$$\lambda_{\max} = \frac{1}{t_T}$$

In packet per second

Utilization: Proportion of time the channel is occupied by data

U = (L/C) /t_T (no units, between 0 and 1)

Stop-and-wait: Performance when no errors

land line: **100Km**
DATA PDU: 1500 bytes
ACK PDU: 1500 bytes
data rate: 1 Mb/s
propagation: 200,000Km/s
neglect H (header)

t_I = 1500x8/1000000 = 12ms = t_a
τ = 100/200,000 = 0.0005s = 0.5ms.

total elapsed time = t_T = 25ms
λ_{max} = 1 pckt every 25ms = 0.48 Mb/s
utilization = **12/25** = **48%**

satellite link: **71,600Km**
DATA PDU: 1500 bytes
ACK PDU: 1500 bytes
data rate: 1 Mb/s
propagation: 300,000Km/s
neglect H (header)

t_I = 1500x8/1000000 = 12ms = t_a
τ = 71.6/300 = 0.238666s = 239.7ms.

total elapsed time = t_T = 503.4ms
λ_{max} = 1 pckt every 503.4ms = 0.024 Mb/s
=24kb/s
utilization = **12/503.4** = **2.4%**

Link layer protocol limits use of physical resources!

Stop and wait when errors

- Let ***p*** be the probability that a frame **or** its ACK be corrupted. Let the time-out be the roundtrip delay, i.e.,

$$t_T = t_I + 2\tau + t_A$$

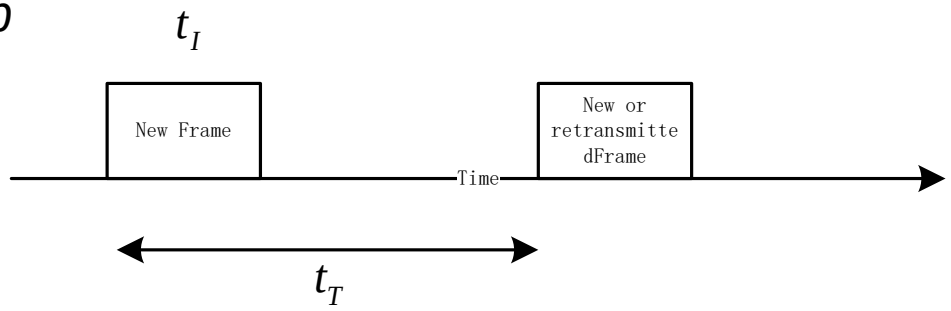
Hence a frame will be received and acknowledged correctly after:

- t_T with prob. **(1-p)**
- **2** t_T with prob. **p(1-p)**
- **k** t_T with prob. $\mathbf{p^{k-1}(1-p)}$

- Let t_v be the average time for a correct transmission and Ack , i.e., = Avg time between successive transmissions of correct frames in the saturated case.

$$t_v = \sum_{k=1}^{\infty} kp^{k-1}(1-p)t_T = \frac{t_T}{1-p}$$

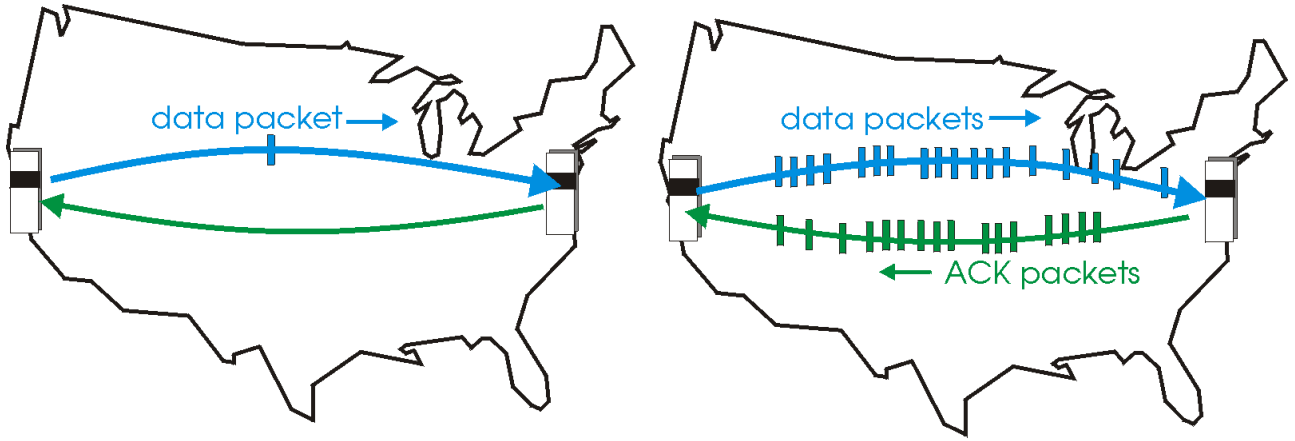
$$\therefore \lambda_{\max} = \frac{1}{t_v} = \frac{1-p}{t_T}$$



Pipelined protocols

Pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged pkts

- range of sequence numbers must be increased
- buffering at sender and/or receiver



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

- Two generic forms of pipelined protocols: *go-back-N, selective repeat*

Pipelining Protocols

Go-back-N: big picture:

- Sender can have up to N un-acked frames in pipeline
- Rcvr does not keep frames out of order (buffer size = 1)
  - Doesn't keep packet if there's a gap
  - Sender retransmits all un-acked packets if timer expires (has timer for oldest un-acked frame) or when window gets full

Selective Repeat: big picture

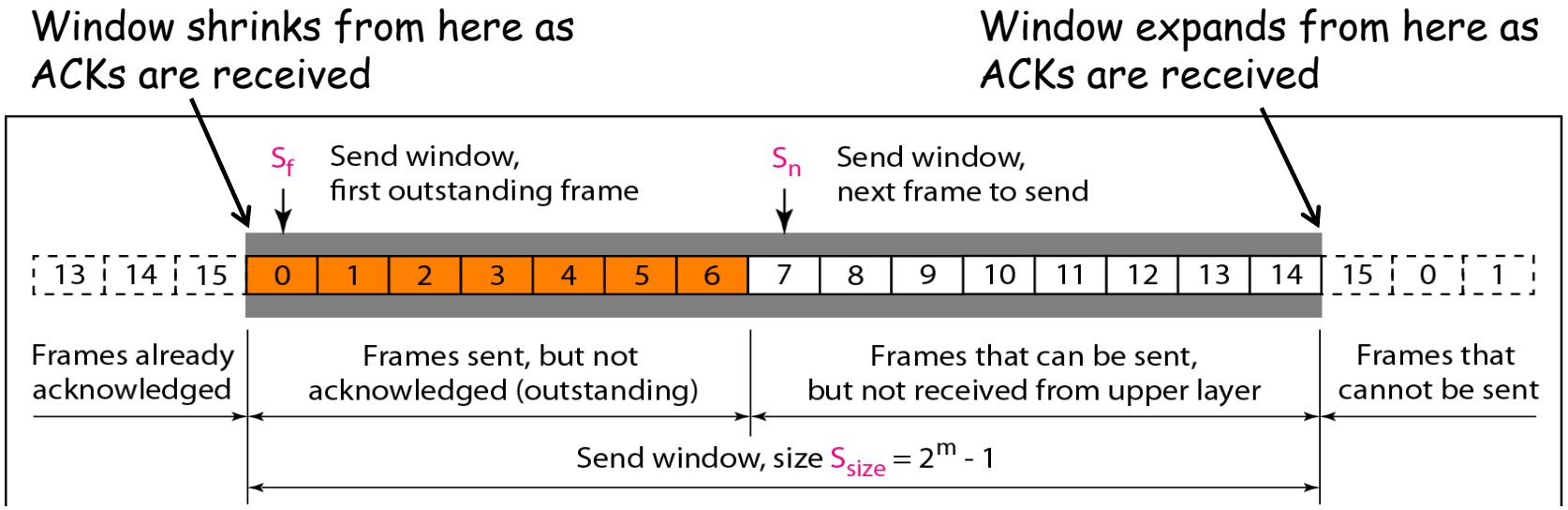
- Sender can have up to N un-acked frames in pipeline
- Rcvr keeps frames out of order
- Sender maintains timer for each un-acked frame
  - When timer expires, retransmit only un-acked frame

Go Back N (GBN)

- Sender allows a window of up to N outstanding (un-ACKed) frames in pipeline, i.e., sender is allowed to send N frames without waiting for ACK
- If the ACK for oldest received frame arrives at sender before window is full, we can continue transmitting
- Resend all outstanding frames either when window gets full or when the timer expires
- Sender's windows can be fixed size or dynamically growing and shrinking
- Receiver does not keep packets out of order → doesn't ack packet if there's a gap.

Sliding Window: The sender

- The sender's window size can be $2^m - 1$, with S_f, S_n, and S_{size}.
- Sender window size (S_{size}) can buffer up to S_{size}
- First outstanding frame (S_f): increase S_f when ACK arrives
- Next frame to send (S_n): increase S_n when a frame is sent
- The window slides to include new unACKed/unsent frames
- Once the window gets full (max # of outstanding frames is reached), entire window gets resent



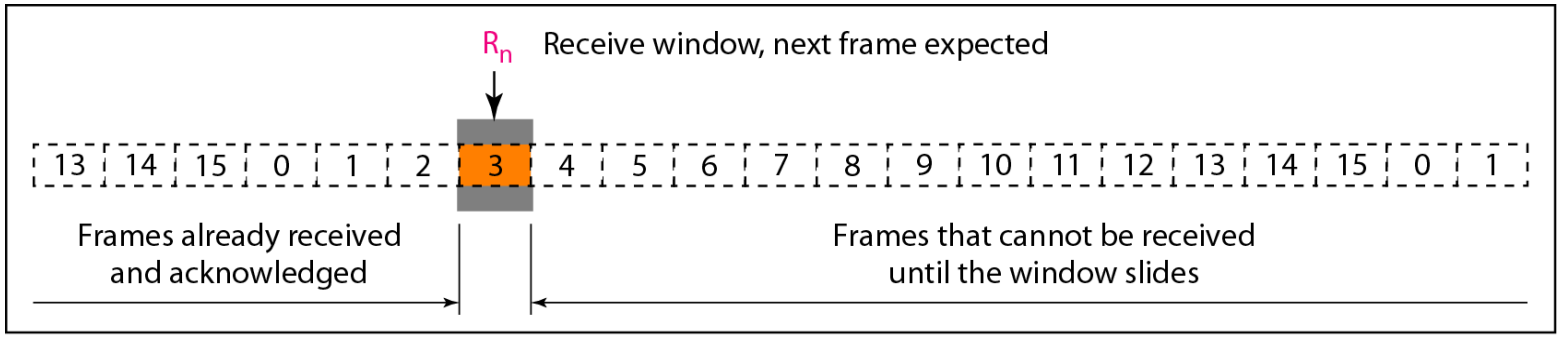
Sliding Window: The sender

- **Frame sequence numbers**: Frames from a sender are numbered sequentially and included in the frame header
- Sequence number is bounded by the length of "sequence number field" in the header, e.g., m bits → Frames are numbered modulo 2^m
- If the header of the frame allows m bits for sequence number, the sequence numbers range from 0 to $2^m - 1$. for m = 3, sequence numbers are: 0, 1, 2, 3, 4, 5, 6, 7.
- We can repeat the sequence number:

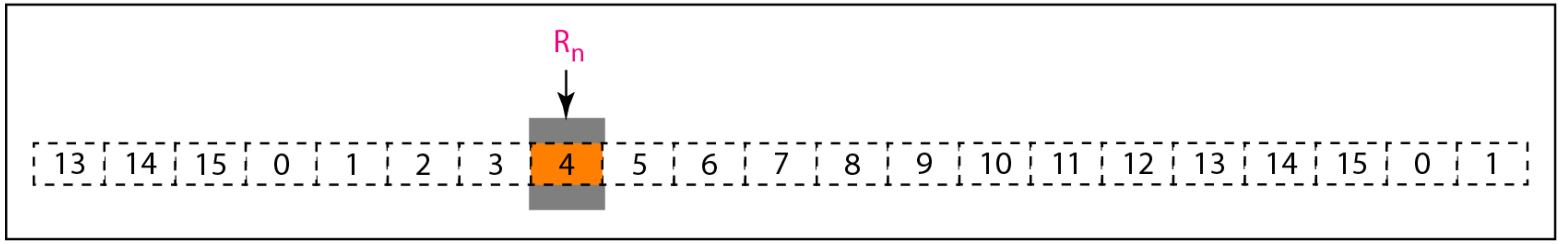
  0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, …

Sliding Window: The receiver

- The receiver window is of size one with a single variable R_n.
- The window slides when a correct frame has arrived; sliding occurs one slot at a time.
- **ACKs' sequence number always defines the SN of the next expected frame**. In Go-Back-N, receiver can send one cumulative ACK for several frames



a. Receive window



b. Window after sliding

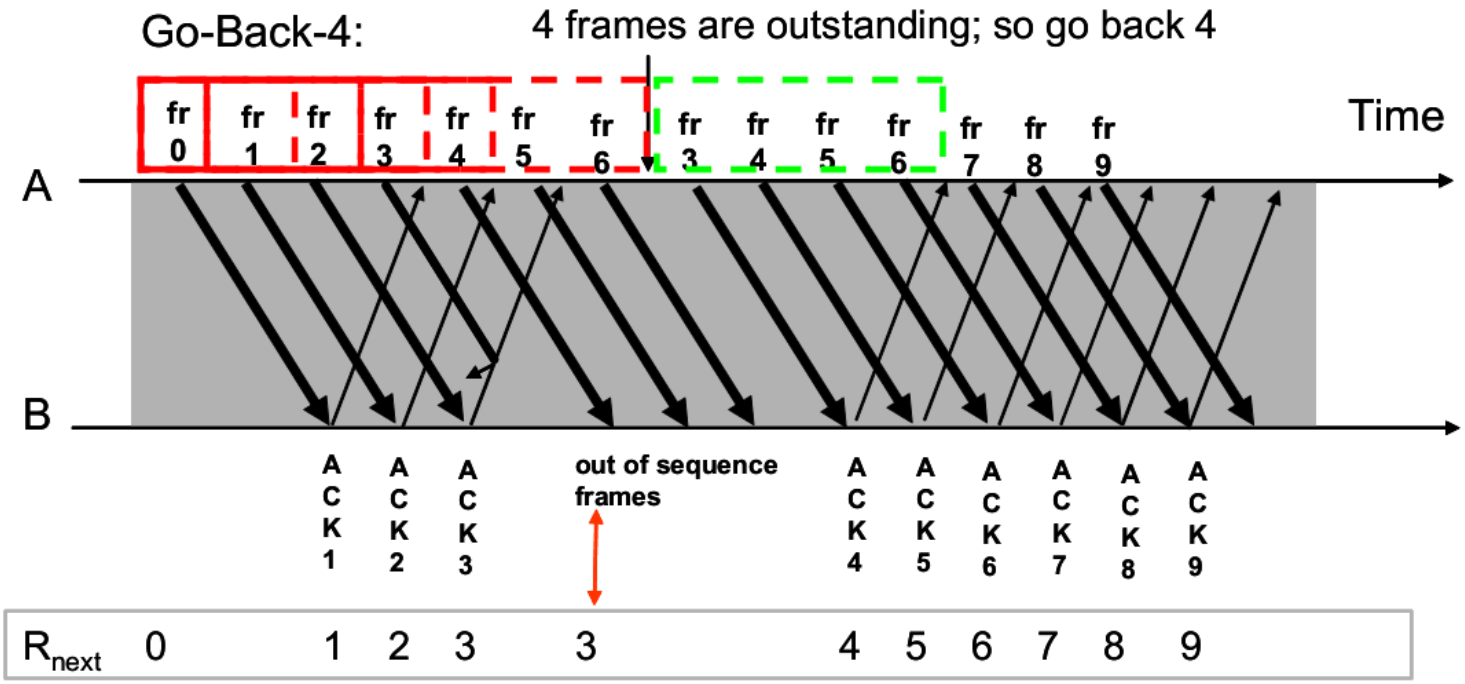
Go Back N (GBN)

Assume: Sliding window size (W_s) = 4

1) sender sends frames one by one

2) frame 3 undergoes transmission error – receiver ignores all subsequent frames after frame 3

3) sender eventually reaches max number of outstanding frames, and takes following action: go back N=W_s frames and retransmit all frames from 3 onwards



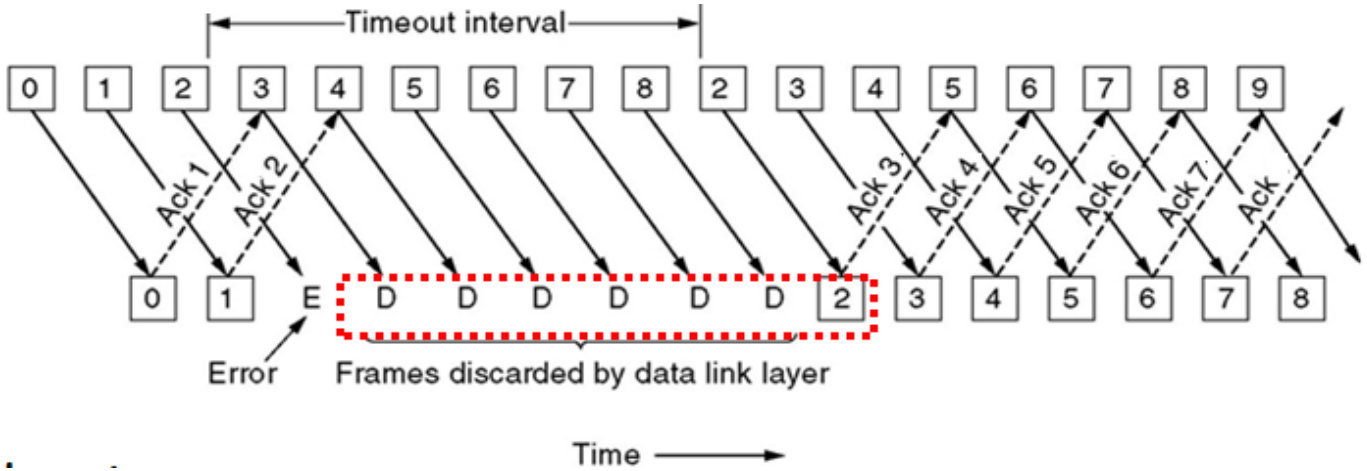
Selective Repeat ARQ

- Go-Back-N is NOT suitable for 'noisy links'
  - in case of a lost/damaged frame a whole window of frames need to be resent

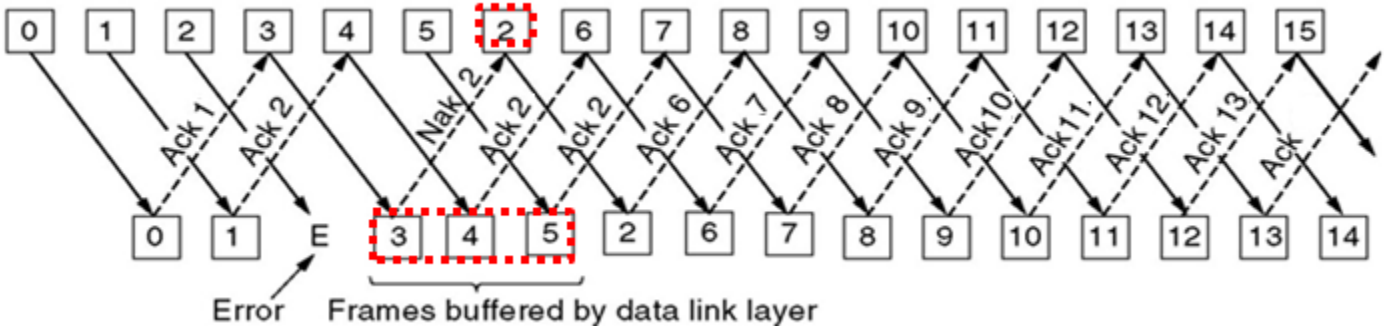
- Selective Repeat ARQ introduces 2 new features:
  (1) receiver window > 1 frame, so that out-of-order but error-free frames can be accepted
  (2) retransmission mechanism is modified – only individual frames are retransmitted

Selective Repeat ARQ

1. Go-back-N

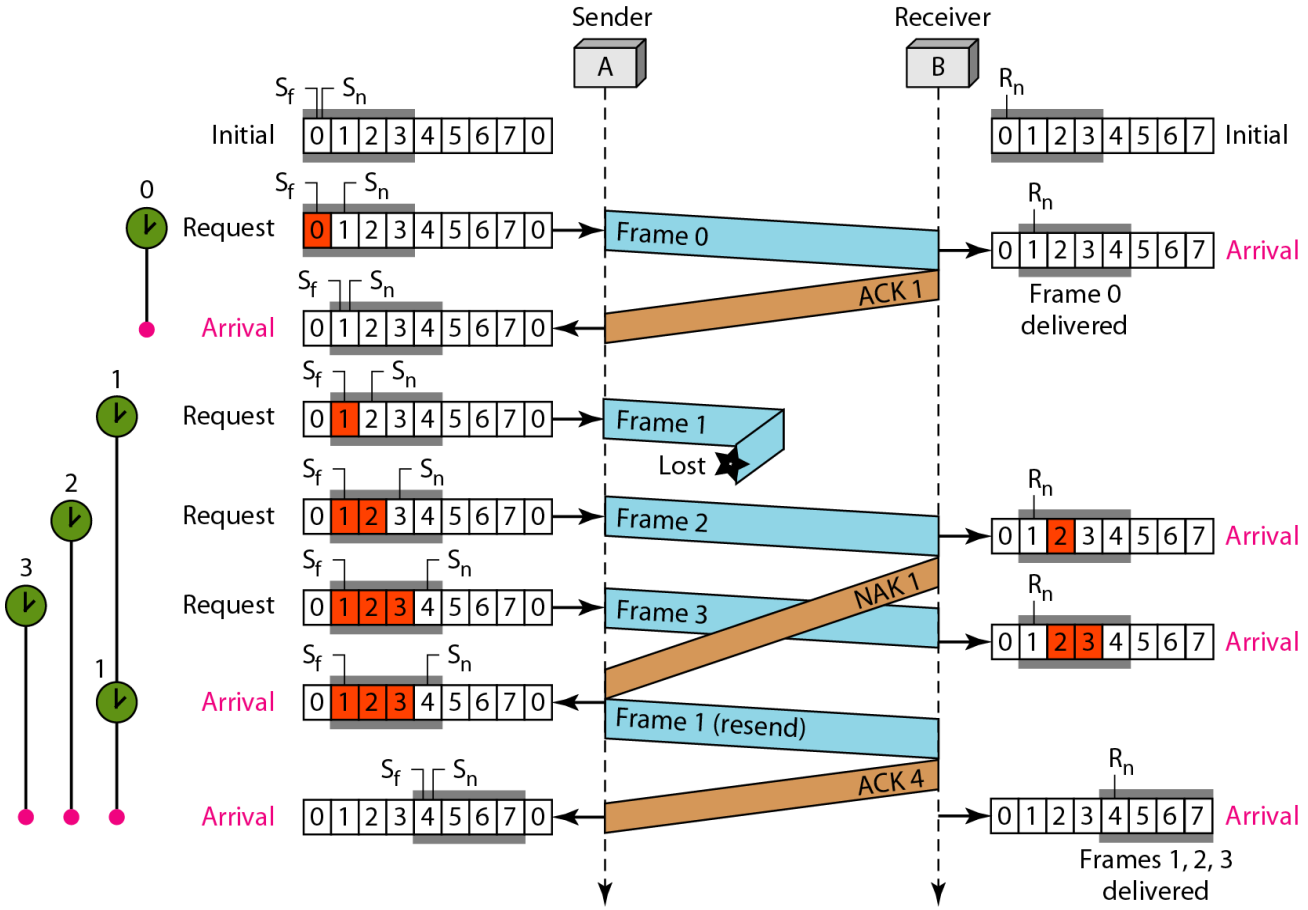


2. Selective repeat



- Out-of-order frames are buffered
- An NAK frame is sent to trigger the retransmission of the frame at the sender.

Selective Repeat ARQ



- Receiver window slides whenever next in-order frame arrives
- Out-of-order frames are accepted only when they are in the range specified by the receiver window.
- a negative ACK (NAK) with sequence number R_{next} is sent whenever an out-of-sequence frame is observed

A Word on Flow Control

- The receiver can control the source: for its own sake or the network's sake
- By delaying ACKs,
- By negotiating the right window size at set-up,
- By changing the window size during the session.

Link Layer

3.1 Introduction and services

3.2 Framing

3.3 Error detection and correction

3.4 Retransmission

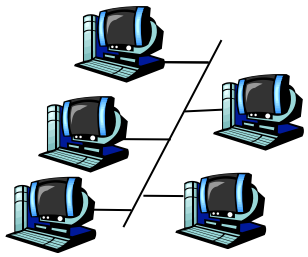
3.5 Multiple access protocols

3.6 Link-layer Addressing

Multiple Access Links and Protocols

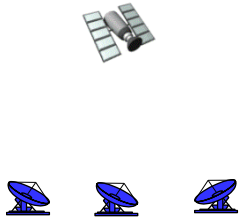
Two types of "links":

- point-to-point
  - PPP for dial-up access
- broadcast (shared wire or medium)
  - Ethernet
  - 802.11 wireless LAN (WiFi)



shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

Multiple Access protocols

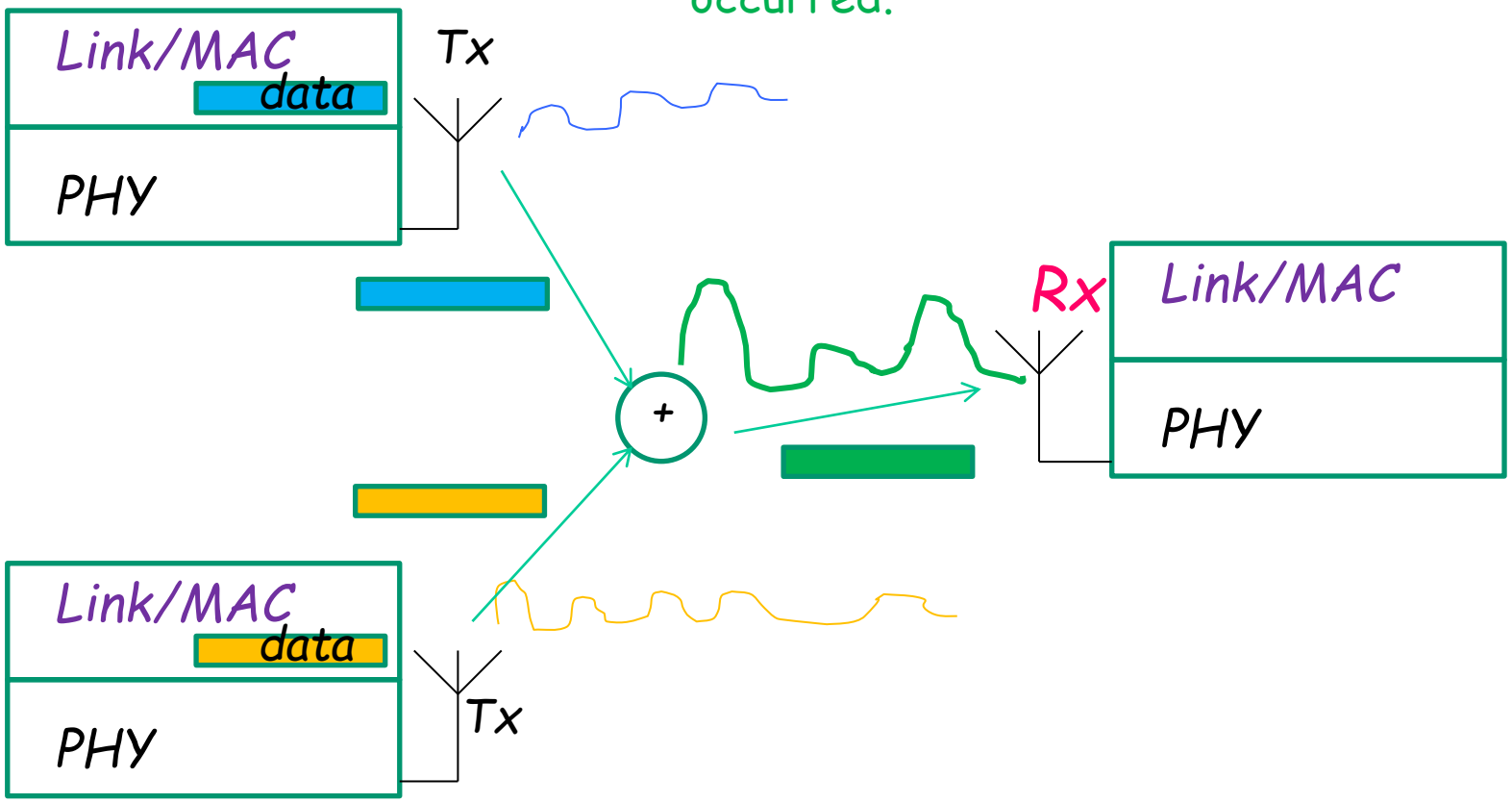
- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - collision if node receives two or more signals at the same time

multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine **when** a node can transmit
- communication about channel sharing must use channel itself!
  - no separate control channel for coordination

Packet Collision

Packet collision occurs at the receiver, but transmitters must know that collision has occurred.



Ideal Multiple Access Control 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
4. simple (plug-and-play, no complex hardware, …)

MAC Protocols: a taxonomy

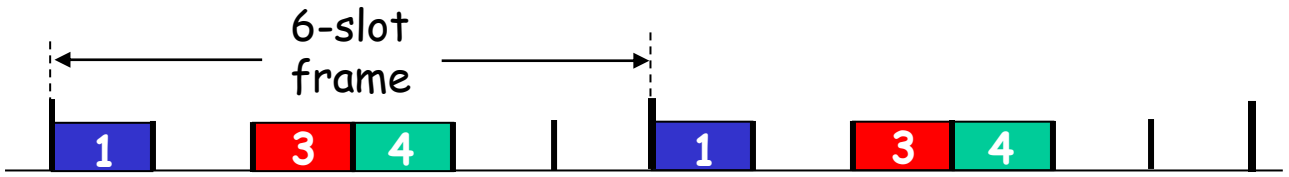
Three broad classes:

- Channel Partitioning (Commonly done in cellular networks)
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- Random Access
  - channel not divided, allow collisions
  - "recover" from collisions
- "Taking turns"
  - nodes take turns

Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

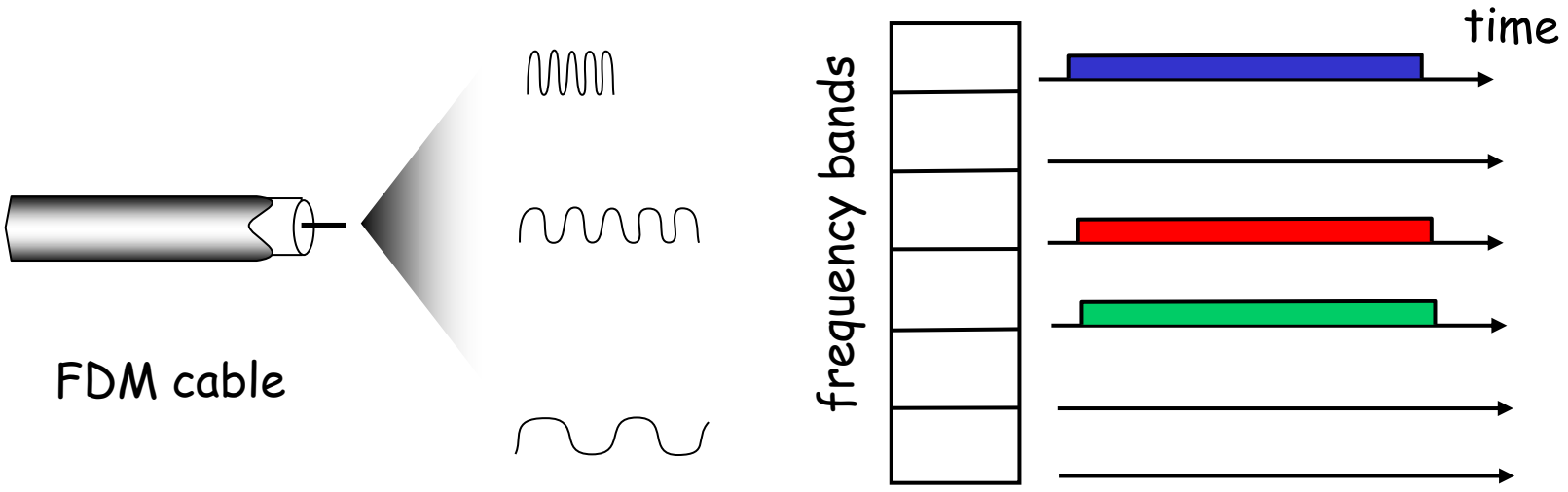
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Random Access Protocols

- When node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision",
- random access MAC protocol specifies:
  - **how** to detect collisions
  - **how** to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
  - ALOHA and slotted ALOHA
  - CSMA/CD, CSMA/CA

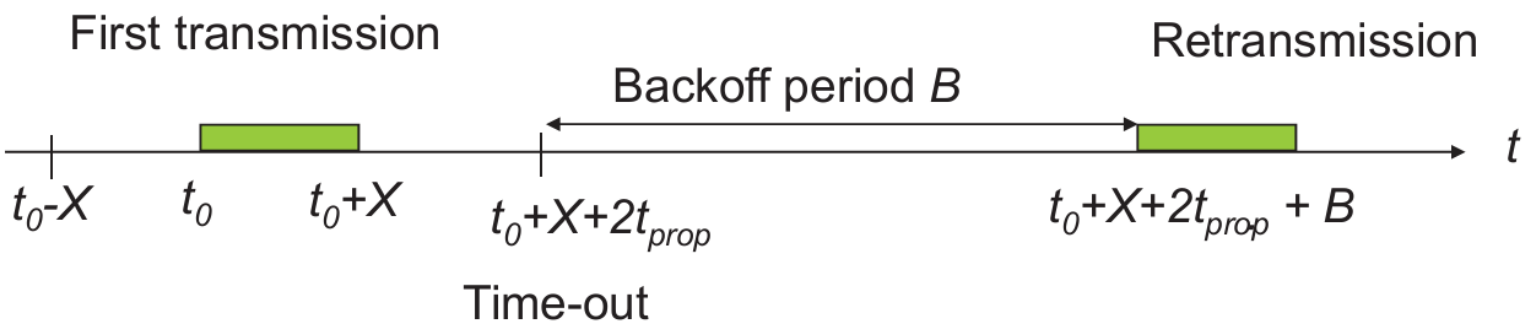
CSMA: Carrier Sense Multiple Access

- CD: Collision **Detection** ← Reactive scheme
- CA: Collision **Avoidance** ← Preventive scheme

Pure ALOHA (unslotted ALOHA)

Protocol:

- A user transmits whenever it has packets to transmit
- When two or more packet transmissions overlap in time, a collision occurs and all the packets involved in the collision are destroyed.
- If ACK not received within timeout, then a user picks random backoff time (to avoid repeated collision)
- User retransmits packet after backoff time



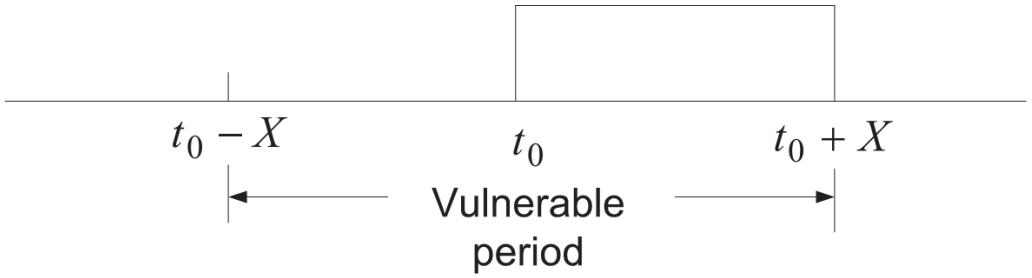
Analysis (Throughput)

1. Definitions and assumptions

X: packet transmission time (assume constant time slot)
S: throughput (average # successful packet transmissions per X sec)
G: load (average # transmission attempts per X sec)
Traffic : traffic (new arrivals+retransmissions) is a Possion process with rate G packet/time slot

$$\left.\begin{array}{l} L = \text{packet length} \\ R = \text{transmission rate} \end{array}\right\} \Rightarrow X = \frac{L}{R} = \text{transmission time} = \text{slot}$$

2. The probability of a successful transmission is the probability that there are no additional packet transmission in the vulnerable period.



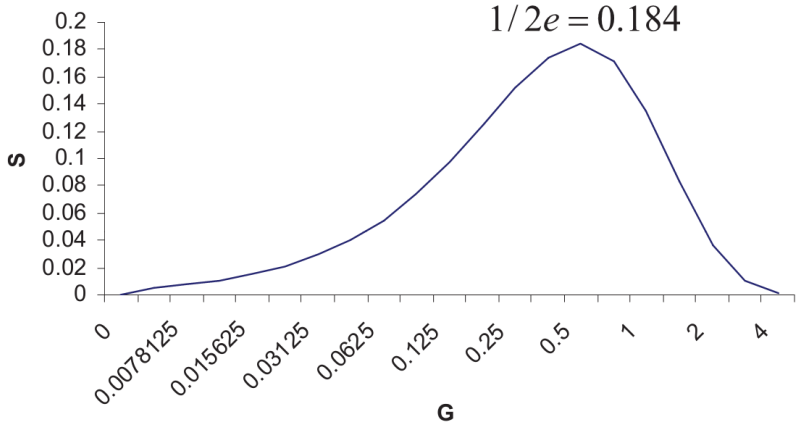
Analysis (Throughput)

For G: $P(k \text{ transmissions in } 2X \text{ seconds})$

$$= \frac{(2G)^k}{k!} e^{-2G}, \quad k = 0, 1, 2, \ldots \text{ (on average, } 2G \text{ arrivals}/2X \text{ seconds)}$$

The throughput $S = GP(\text{no collision})$

$$= GP[\text{0 transmission in 2X seconds}] = G\frac{(2G)^0}{0!}e^{-2G} = Ge^{-2G}$$



Throughput S versus load G for pure ALOHA

S reaches a peak value of 1/2e at G=0.5, and then declines back toward 0.

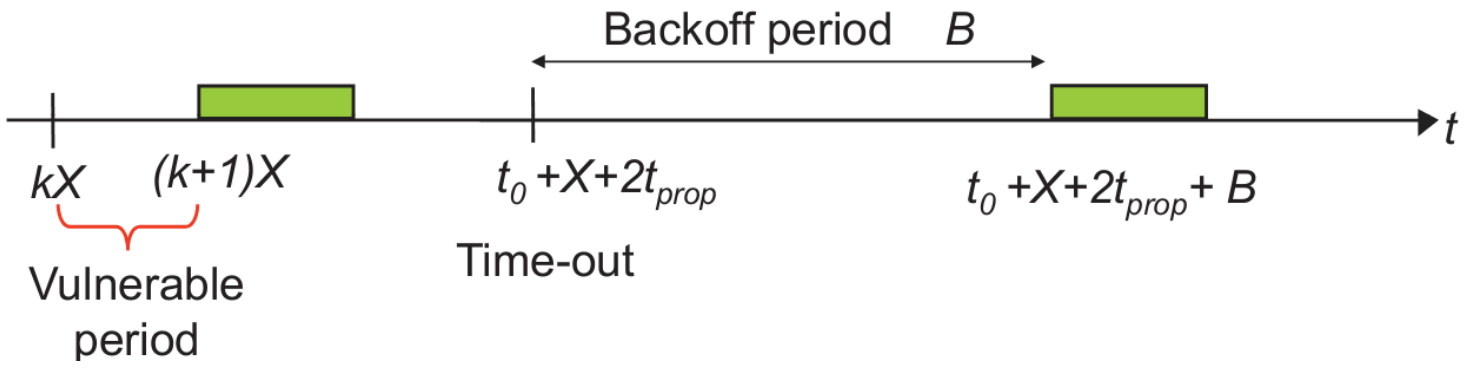
For a given value of S, say, S=0.05, there are two associated values of G.

For small G, S $\approx$ G. For large G, there are many backlogged users.

ALOHA system cannot achieve throughput higher than 18.4 percent (1/2e).

Slotted ALOHA

- Slotted ALOHA is to constrain the user to transmit in synchronized fashion.
- Time is divided into slots of size X=L/R (one frame time)
- Users start to transmit only at the beginning of a time slot.
- When node has a fresh frame to send, it waits until next frame slot and transmits
- If there is a collision, node retransmits the frame after a backoff-time

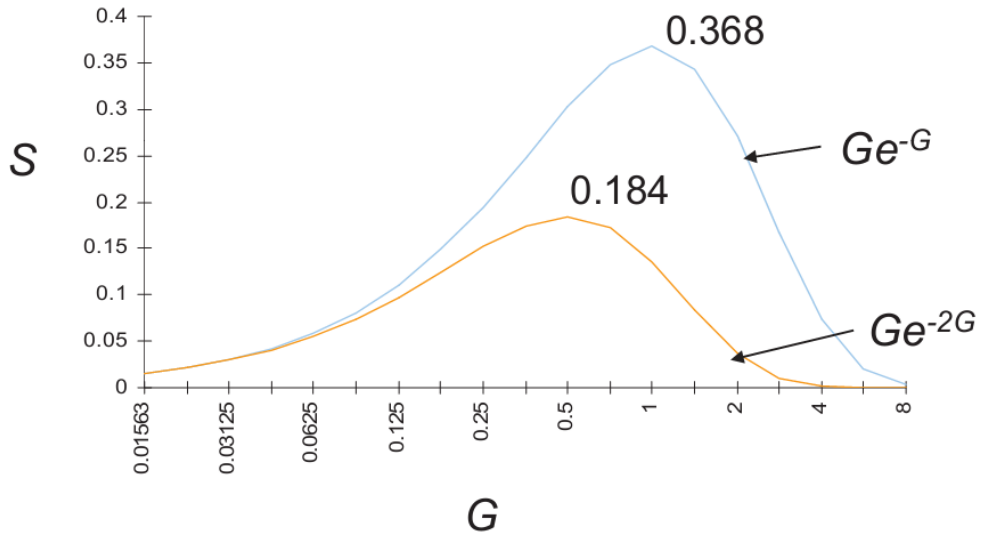


Only packets that arrive during prior X seconds collide.

Throughput of Slotted ALOHA

$$S = GP[\text{no collision}] = GP[\text{no arrivals in X seconds}]$$

$$= G \cdot \frac{(G)^0}{0!} e^{-G} = G \cdot e^{-G}$$

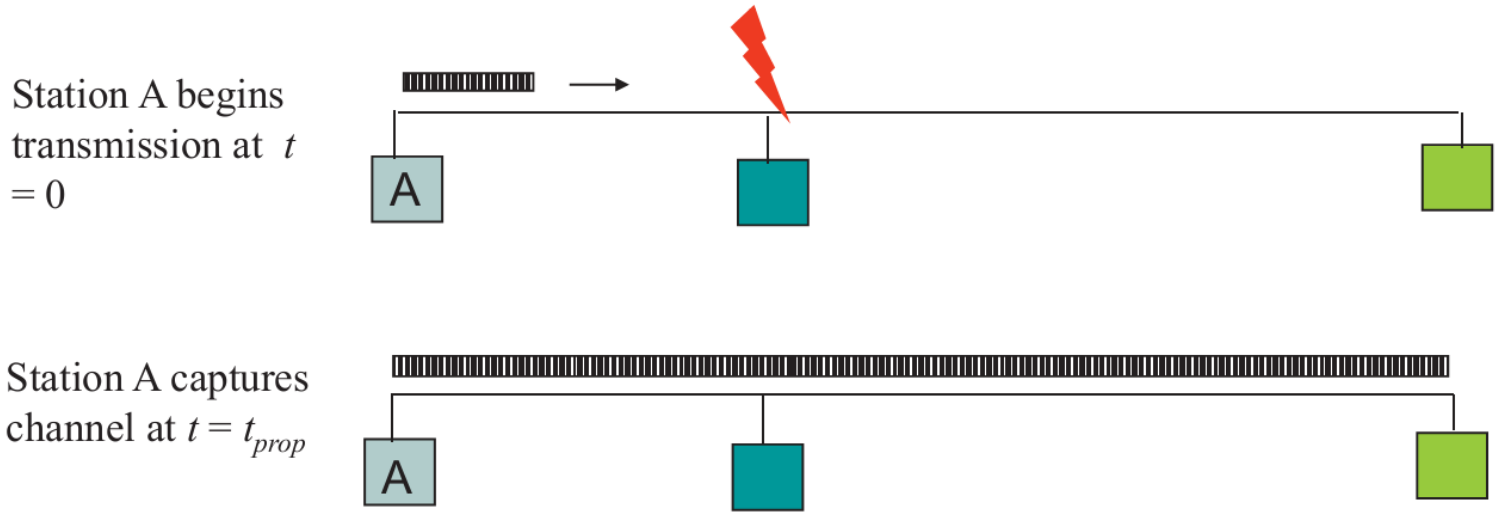


$S_{max}=1/e=36.8\%$

Efficiency=throughput · 36% (upper bound=36%)

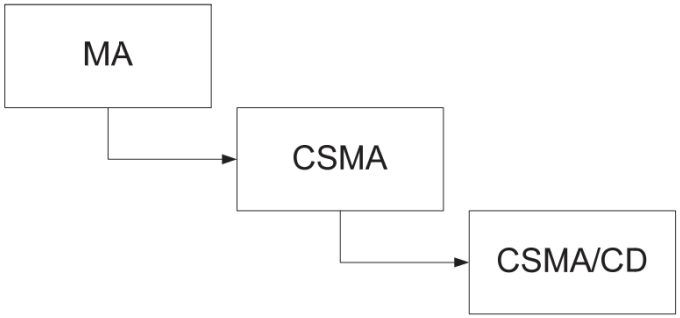
Carrier Sensing Multiple Access (CSMA)

A station senses the channel before it starts transmission
- If busy either wait or schedule backoff (different options)
- If idle, start transmission
- Vulnerable period is reduced to t_{prop}
- When collisions occur they involve entire frame transmission times
- If $t_{prop} > X$, no gain compared to ALOHA or slotted ALOHA



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

- The access mechanism used in an Ethernet is called CSMA/CD, standardized in IEEE 802.3.
- CSMA/CD is the result of an evolution from multiple access (MA) to carrier sense multiple access (CSMA), and finally, to CSMA/CD.
- In a CSMA system, any user (work station) wishing to transmit must first listen to the existing traffic on the line. A device listens by checking for a voltage. No voltage means the line is idle. CSMA cuts down on the number of collisions but does not eliminate them.
- In CSMA/CD system, the station listens again after each packet transmission. The extremely high voltages indicate a collision.



Analysis of CSMA/CD Protocol

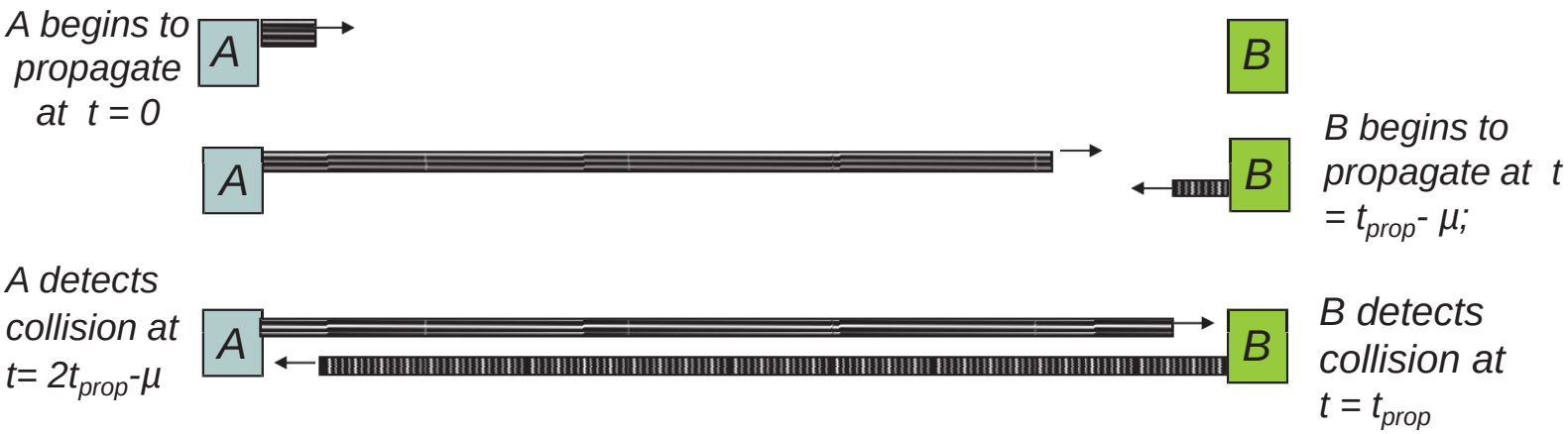
- In CSMA/CD protocol, a node with a packet to transmit must proceed as follows:
  1. Wait until the channel is idle;
  2. When the channel is idle, transmit and listen while transmitting
  3. In case of a collision, stop the packet transmission, and then wait for a random delay and go to 1.

- Note: when a node "goes back to (1)" after its waiting time, it senses the signal from the other nodes and must then wait until the end of that transmission before transmitting.

- In CSMA collisions result in wastage of X seconds spent transmitting an entire frame.

- CSMA-CD reduces wastage to time to detect collision and abort transmission

CSMA/CD reaction time



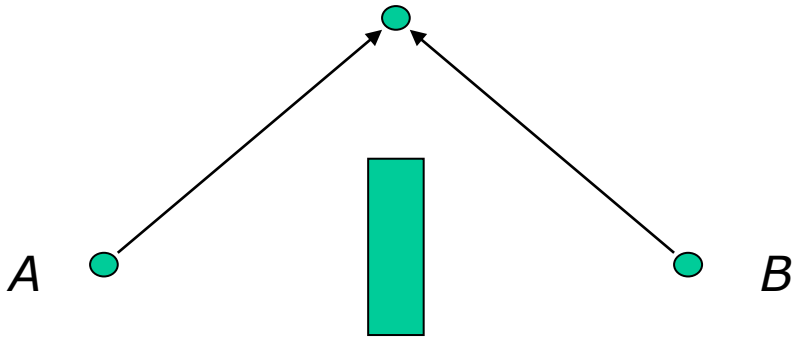
It takes approximately $2t_{prop}$ to find out if channel has been captured

Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA)

- Procedure
  - Similar to CSMA but instead of sending packets control frames are exchanged
  - RTS = request to send
  - CTS = clear to send
  - DATA = actual packet
  - ACK = acknowledgement

Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA)

- Advantages
  - Small control frames lessen the cost of collisions (when data is large)
  - RTS + CTS provide "virtual" carrier sense which protects against hidden terminal collisions (where A can't hear B)



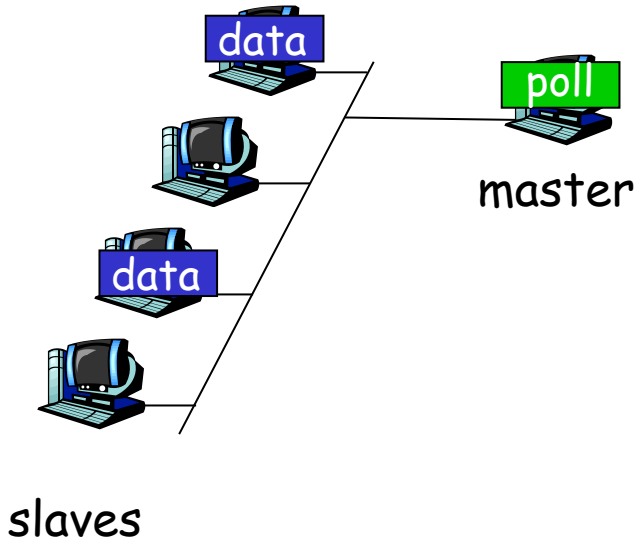
- *Disadvantages*

  Not as efficient as CSMA-CD

"Taking Turns" MAC protocols

Polling:

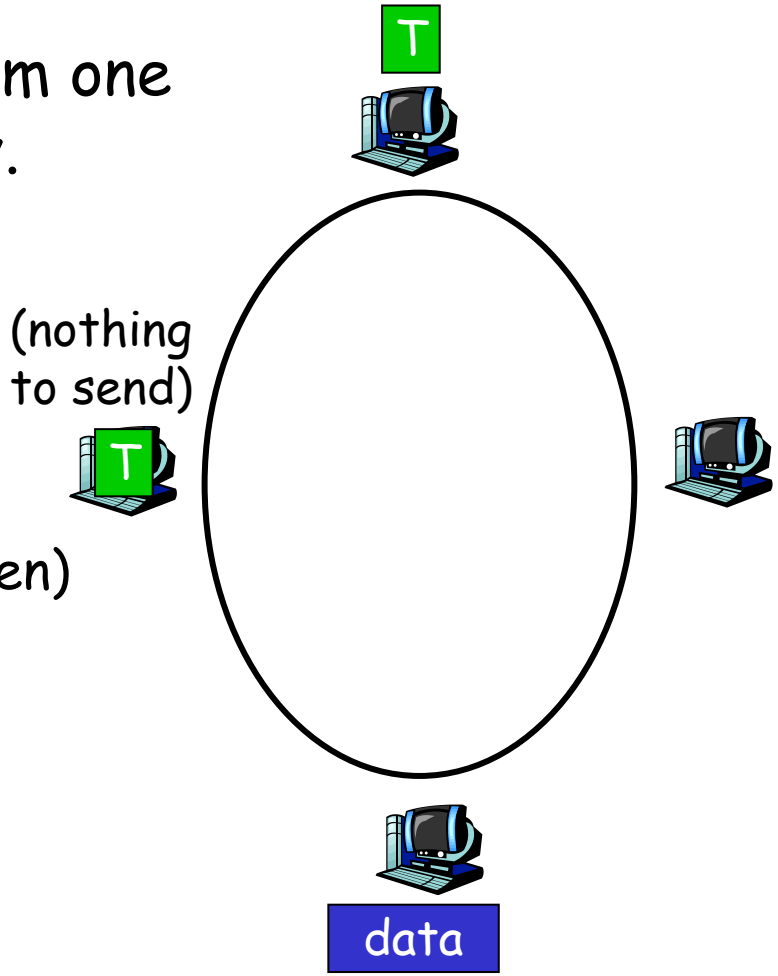
- master node
  "invites" slave nodes to transmit in turn
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"Taking Turns" MAC protocols

Token passing:

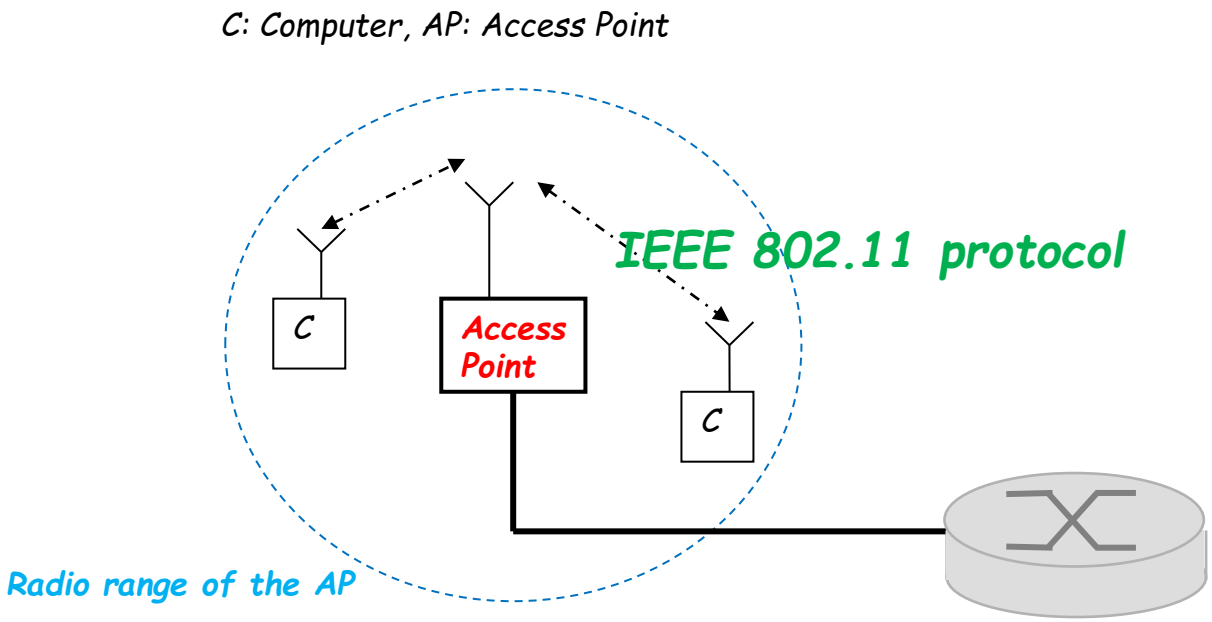
- control **token** passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



Wireless LAN

IEEE 802.11/a/b/g

WLAN View



Basic Service Set (BSS): BSSID = MAC address of AP

Independent BSS (IBSS)= BSS - AP

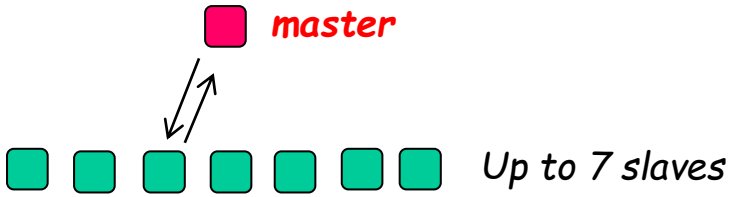
Extended Service Set (ESS): A collection of BSS connected by a Distribution System

IEEE 802.11/a/b/g/n Family

IEEE	Technique	Frequency Band	Rate **(Mbps)**
802.11	DSSS FHSS	2.4 GHz 2.4 GHz	2 2
802.11a	OFDM	5 GHz	54
802.11b	DSSS	2.4 GHz	11
802.11g **802.11n** **802.11ac** **(Draft/Nov. 2011)**	OFDM OFDM OFDM	2.4 GHz 2.4/5 GHz 5 GHz	54 600 **6.9 Gbps**

Summary of MAC protocols

- *channel partitioning* by time, frequency or code
  - Time Division, Frequency Division, Code Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in wireless
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing



Link Layer

3.1 Introduction and services

3.2 Framing

3.3 Error detection and correction

3.4 Retransmission

3.5 Multiple access protocols

3.6 Link-layer Addressing

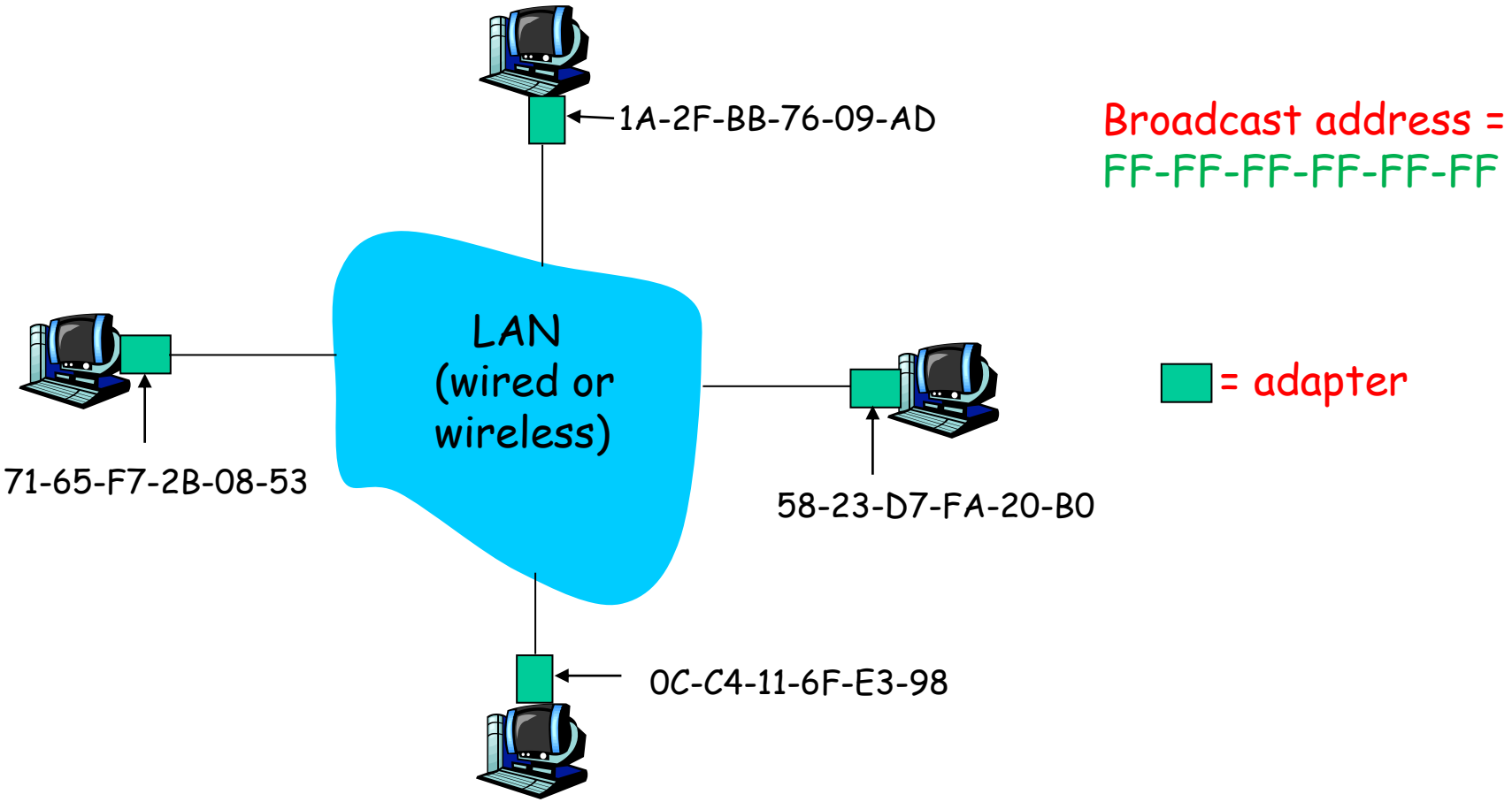
MAC Addresses and ARP (Address Resolution Protocol)

- 32-bit IP address:
  - *network-layer* address
  - used to get datagram to destination IP subnet

- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address
  - burned in NIC ROM, also sometimes software settable

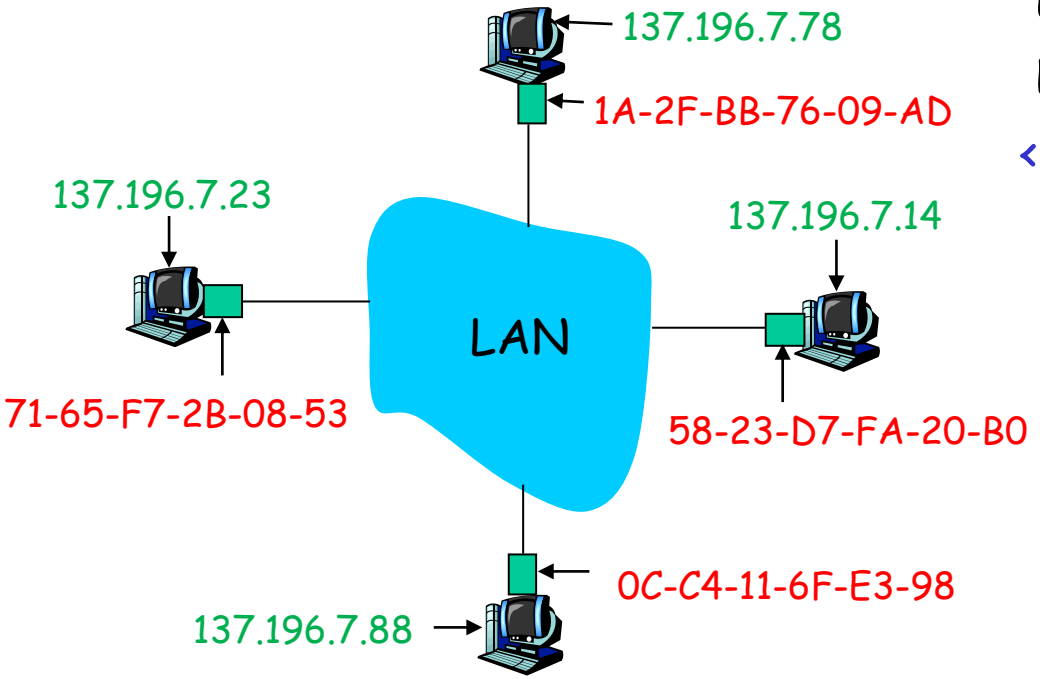
LAN Addresses and ARP

Each adapter on LAN has unique LAN address（hexadecimal notation）



ARP: Address Resolution Protocol

Question:
how to determine B's MAC address knowing B's IP address?



- Each IP node (host, router) on LAN has ARP table
- ARP table: IP/MAC addr. mappings for some LAN nodes

  < IP address; MAC address; TTL>

  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

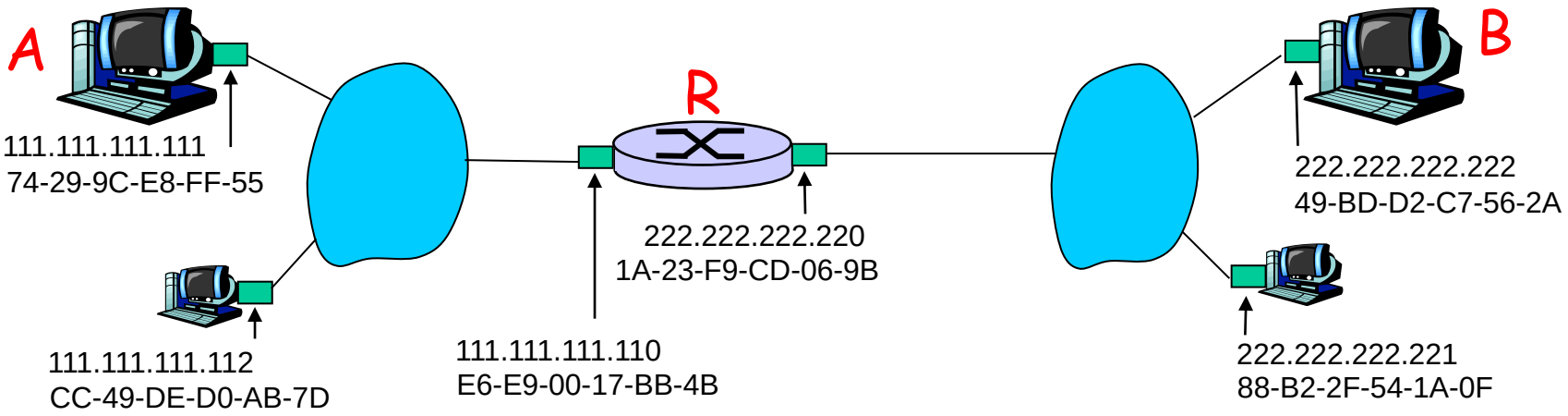
ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A broadcasts ARP query pkt containing B's IP addr
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP query, replies to A with its (B's) MAC addr
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC addr pair in its ARP table until this becomes old (times out)
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

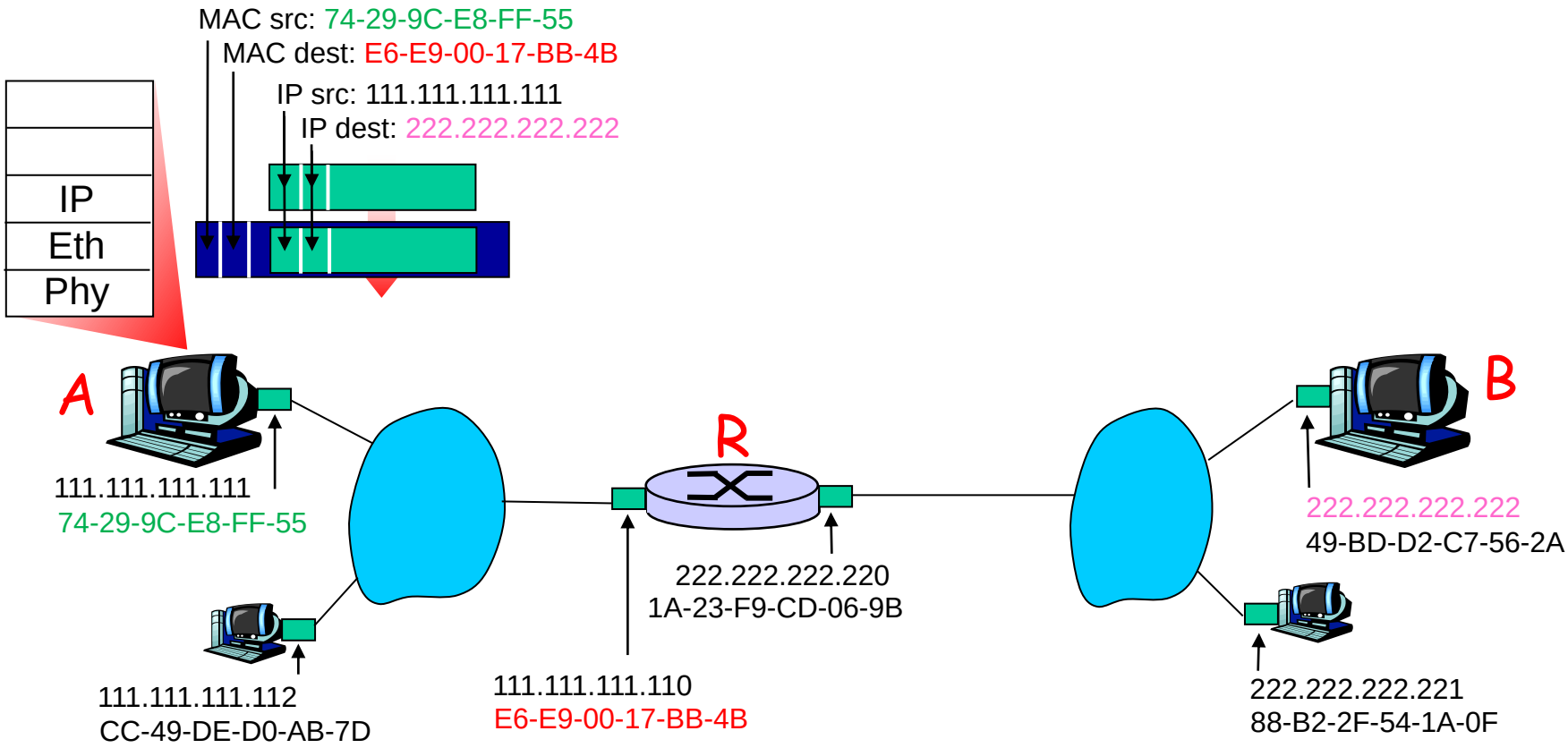
walkthrough: send datagram from A to B via R.

- focus on addressing - at both IP (datagram) and MAC layer (frame)
- assume A knows B's IP address



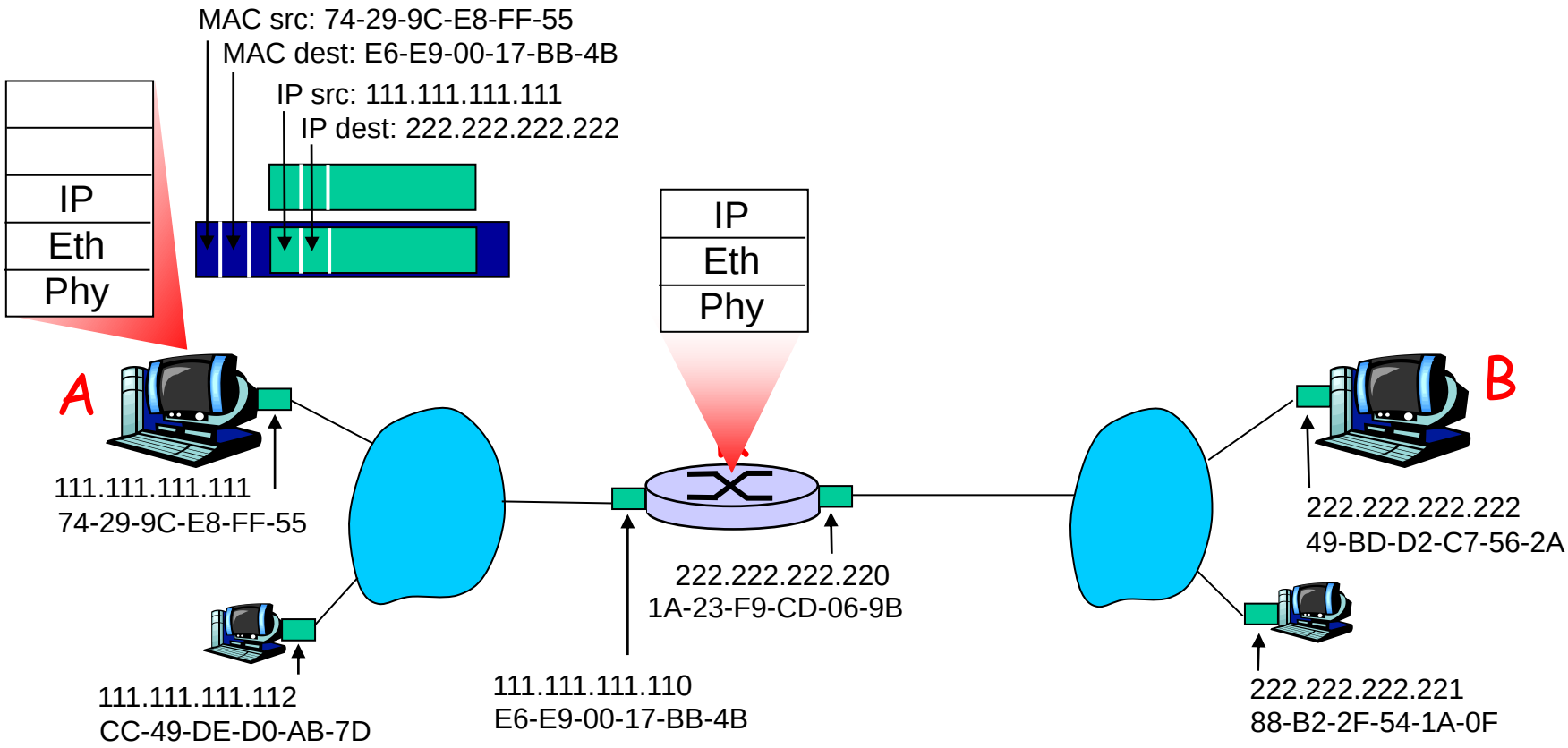
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



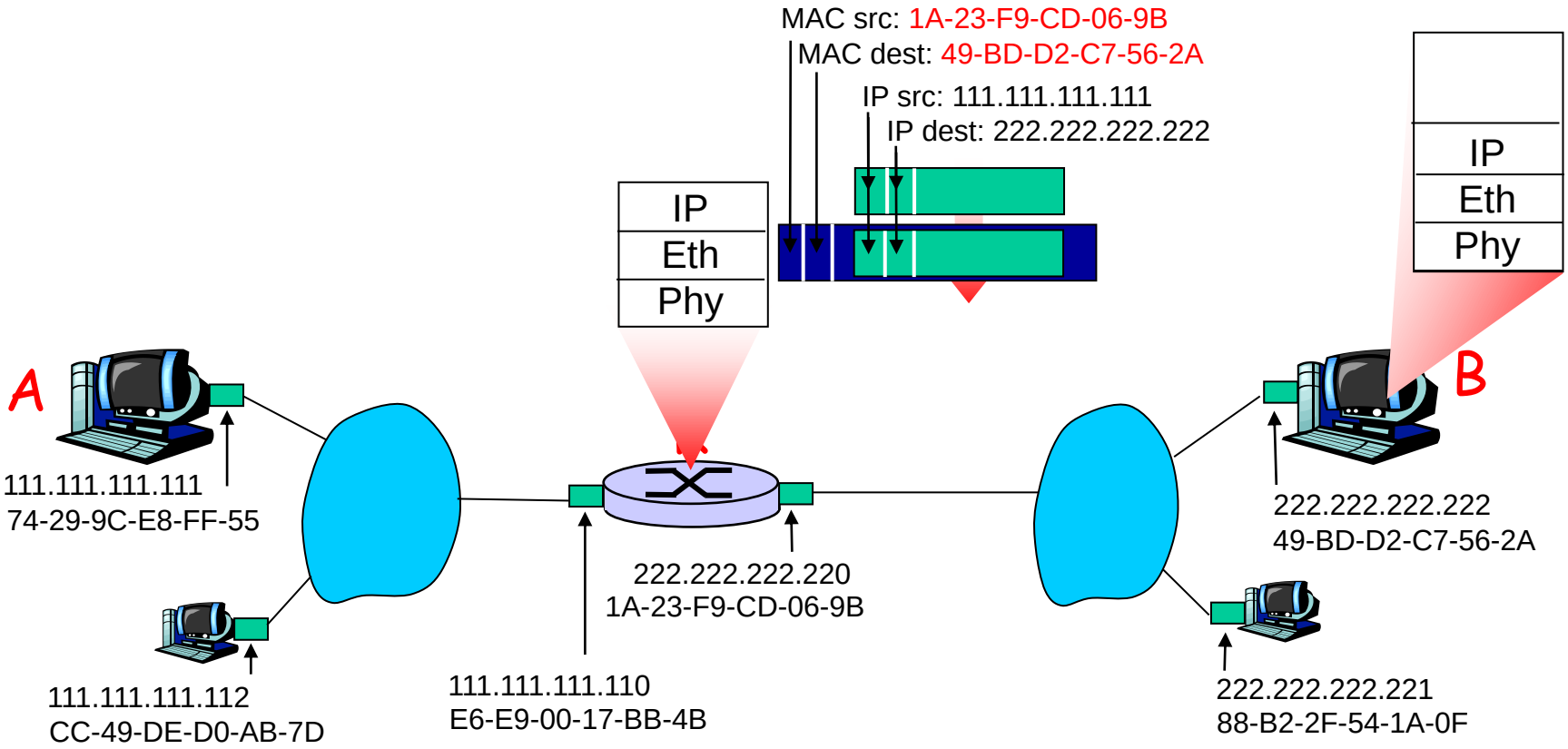
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



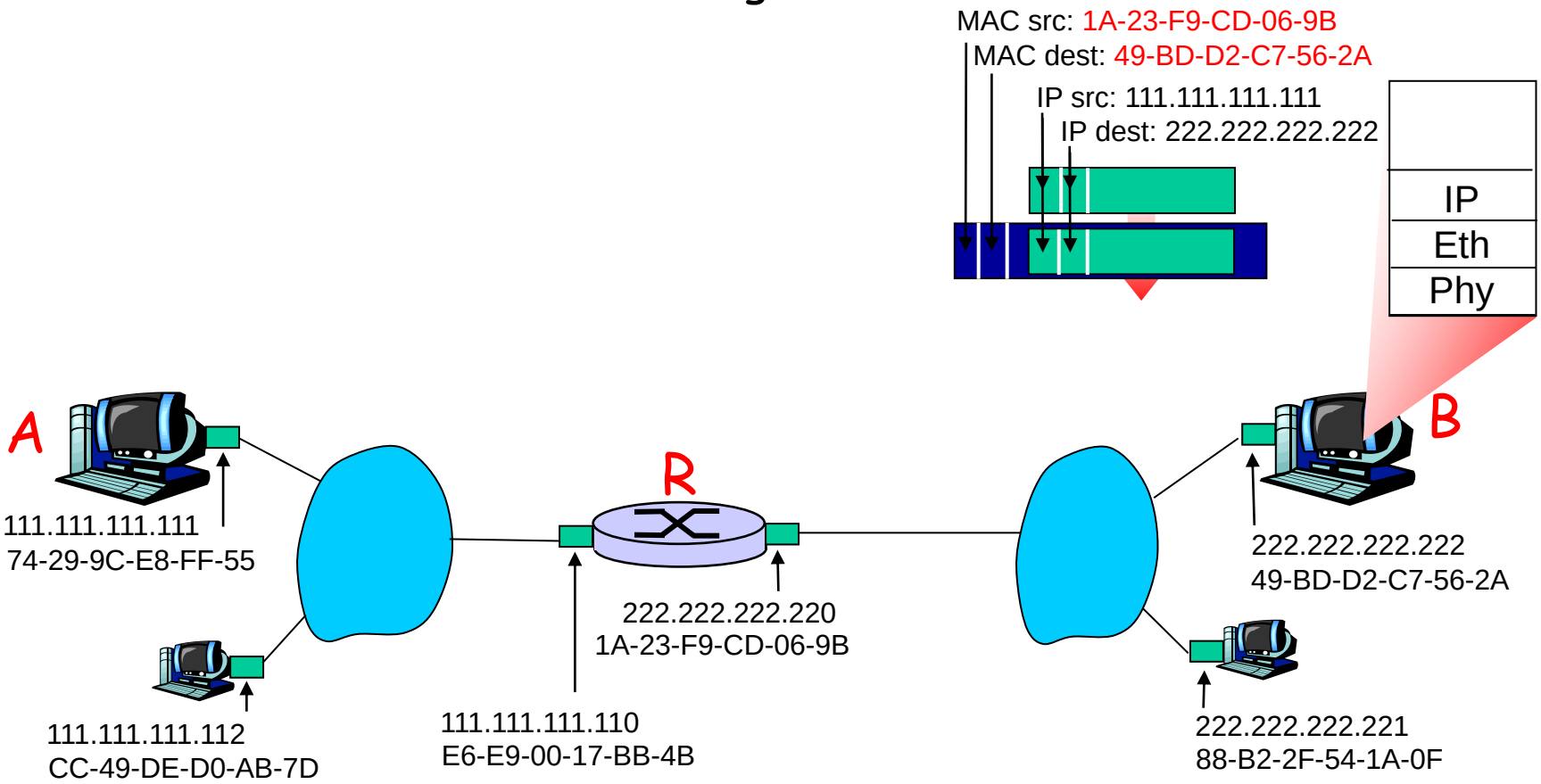
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram

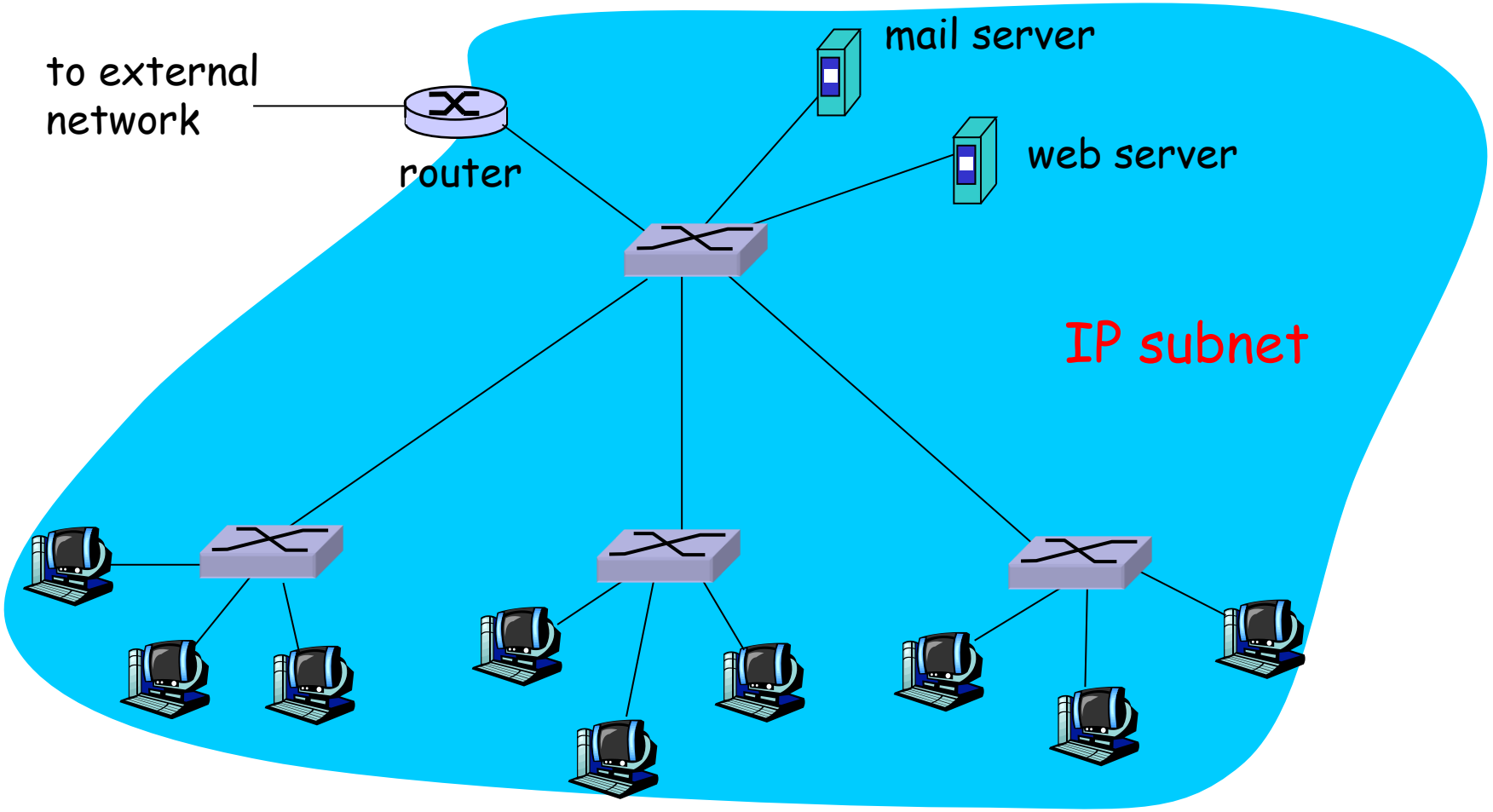


Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Institutional network



Switches vs. Routers

- both store-and-forward devices
  - routers: network-layer devices (examine network-layer headers)
  - switches are link-layer devices (examine link-layer headers)
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms