BLG 337E- Principles of Computer Communications

Assist. Prof. Dr. Berk CANBERK

11/11/ 2014

-Medium Access Layer - 2

References:

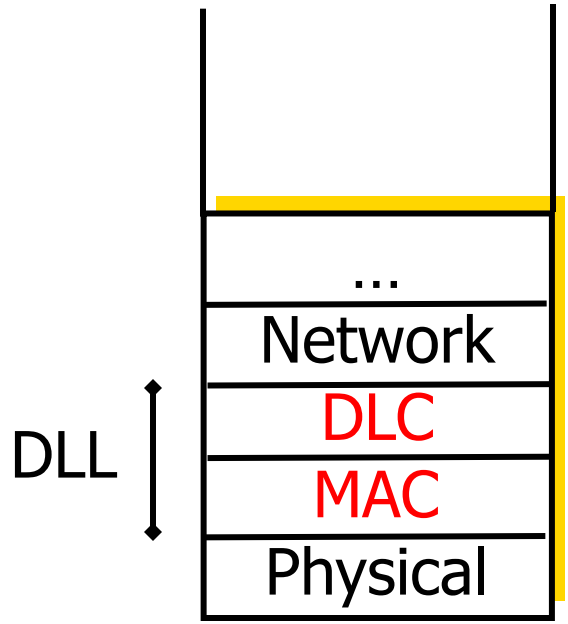
Data and Computer Communications, William Stallings, Pearson-Prentice Hall, 9th Edition, 2010.

-*Computer Networking, A Top-Down Approach Featuring the Internet*, James F.Kurose, Keith W.Ross, Pearson-Addison Wesley, 6th Edition, 2012.

-Google!

Data Link Layer

- Services Provided to the Network Layer
  - Framing
  - Medium Access Control
  - Error Control: Dealing with transmission errors
  - Flow Control: Slow receivers not swamped by fast senders

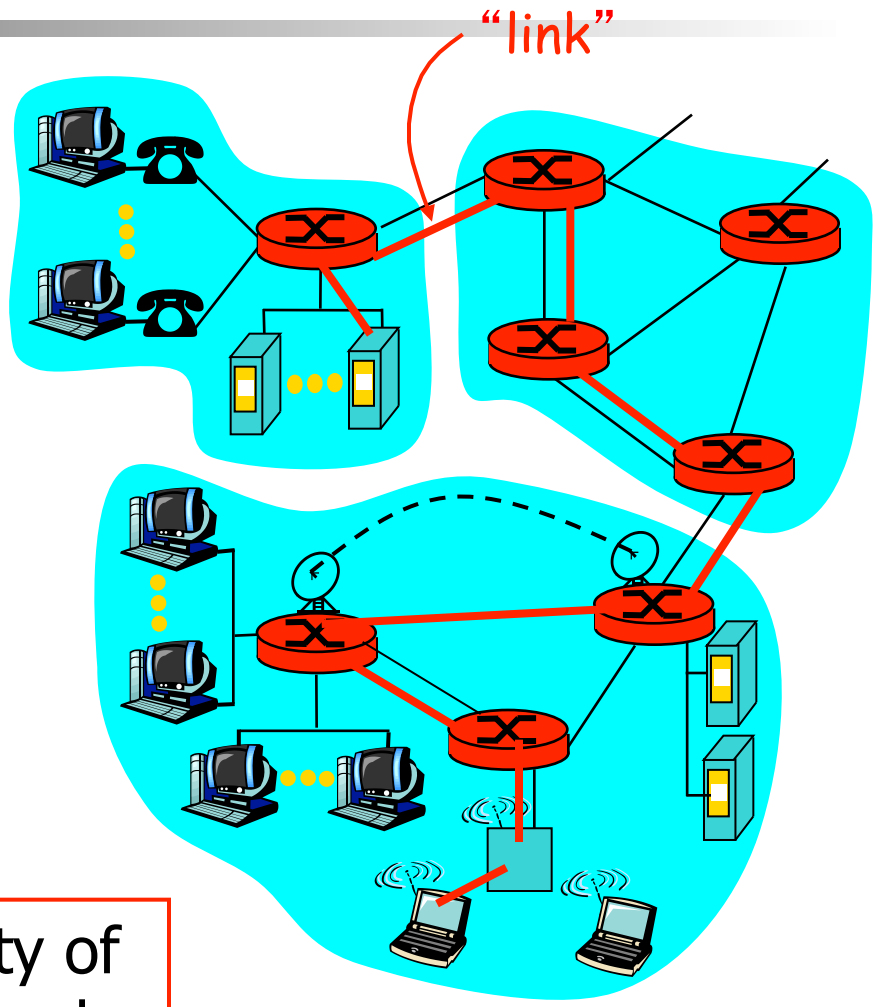


DLL: Data Link Layer
DLC: Data Link Control
MAC: Medium Access Control

Data Link Layer

Some terminology:

- hosts and routers are **nodes**
- communication channels that connect adjacent nodes along communication path are **links**
  - wired links
  - wireless links
- layer-2 packet is a **frame**, encapsulates datagram



Data link layer has responsibility of transferring datagram from one node to adjacent node over a link

Data Link Layer

- Datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- Each link protocol provides different services
  - e.g., may or may not provide reliability over link

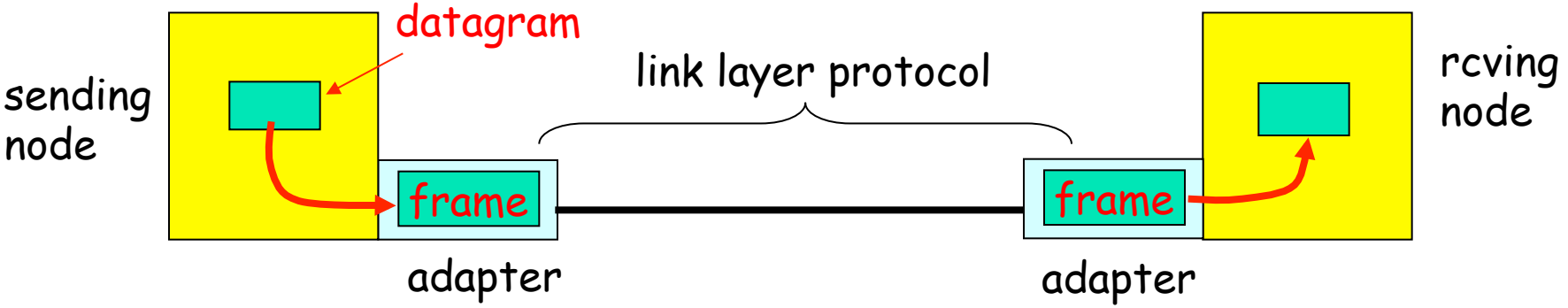
transportation analogy

- trip from METU to New Jersey
  - car: METU to Esenboga
  - plane: ESB to JFK
  - train: New York to New Jersey
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

Link Layer Services

- Framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- Reliable delivery between adjacent nodes
  - seldom used on low bit error link (fiber, some twisted pair)
  - wireless links: high error rates
- Error Detection:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- Error Correction:
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- Flow Control:
  - pacing between adjacent sending and receiving nodes

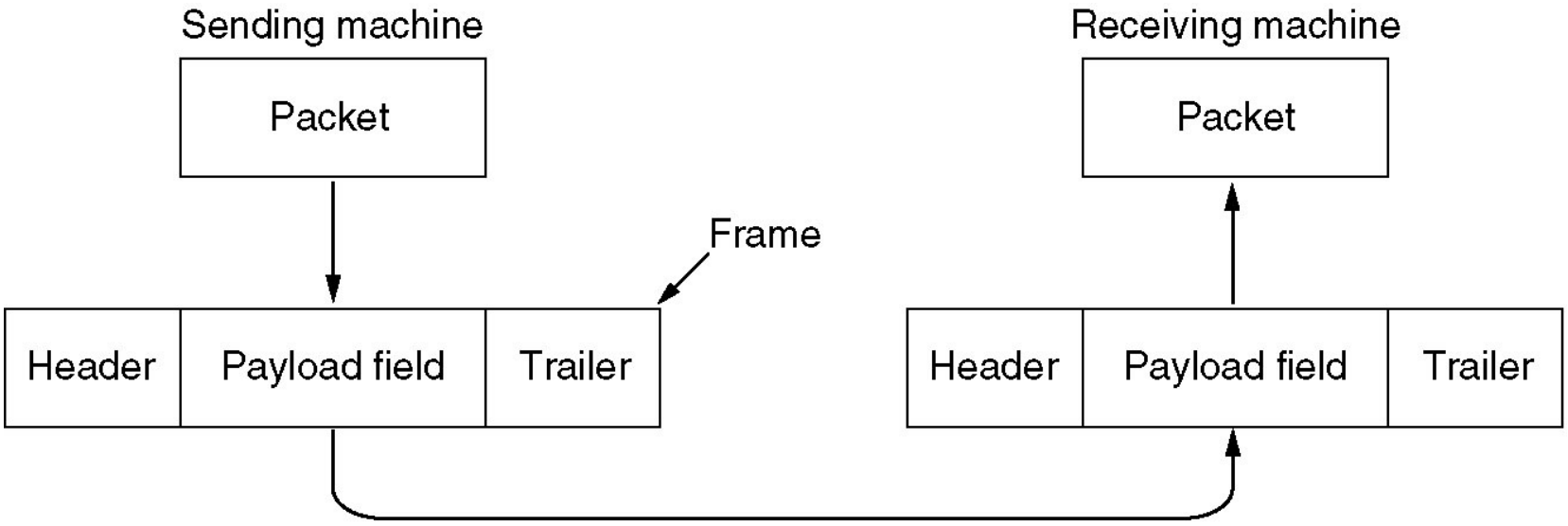
Adaptors Communicating



- link layer implemented in "adaptor" (NIC)
  - Ethernet card, PCMCI card, 802.11 card
- sending side:
  - encapsulates datagram in a frame
  - adds error checking bits, flow control, etc.
- receiving side
  - looks for errors, flow control, etc
  - extracts datagram, passes to rcving node
- adapter is semi-autonomous
- link & physical layers

Functions of Data Link Layer

Relationship between packets and frames.



MAC Addresses and ARP

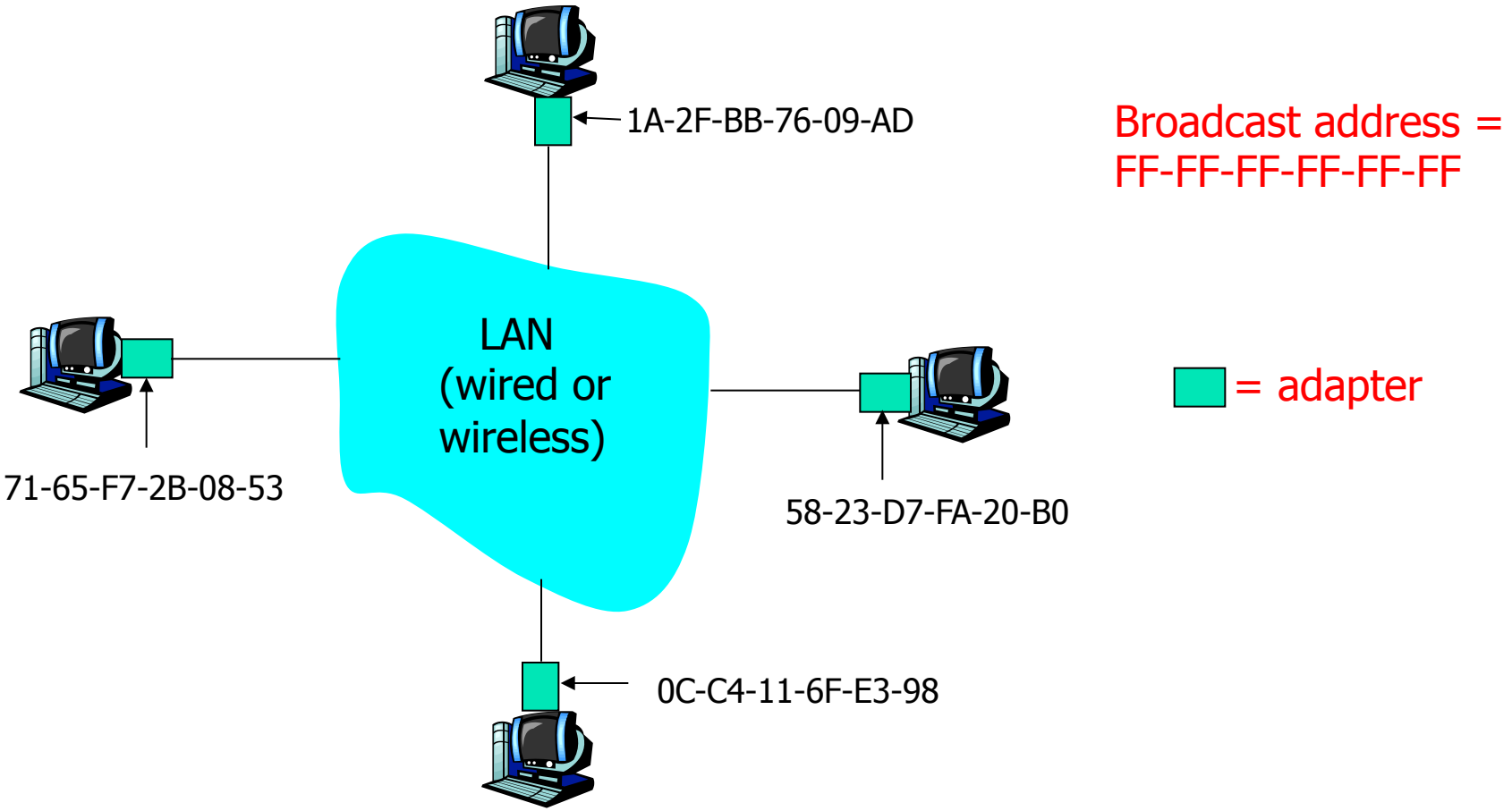
- 32-bit IP address:
  - network-layer address
  - used to get datagram to destination IP subnet
- MAC (or LAN or physical or Ethernet) address:
  - used to get datagram from one interface to another physically-connected interface (same network)
  - *48 bit MAC address (for most LANs) burned in the adapter ROM*
  - *MAC address allocation administered by IEEE*
- Analogy:

  (a) MAC address: like Social Security Number

  (b) IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - depends on IP subnet to which node is attached

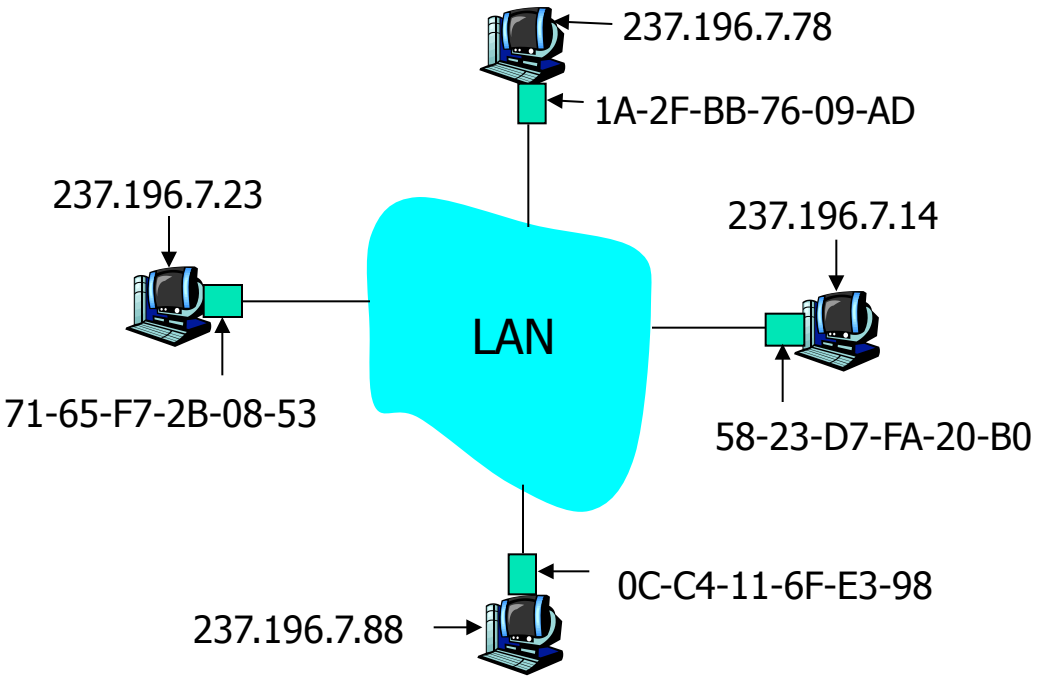
LAN Addresses and ARP

Each adapter on LAN has unique LAN address



ARP: Address Resolution Protocol

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



- Each IP node (Host, Router) on LAN has ARP table
- ARP Table: IP/MAC address mappings for some LAN nodes

  < IP address; MAC address; TTL>

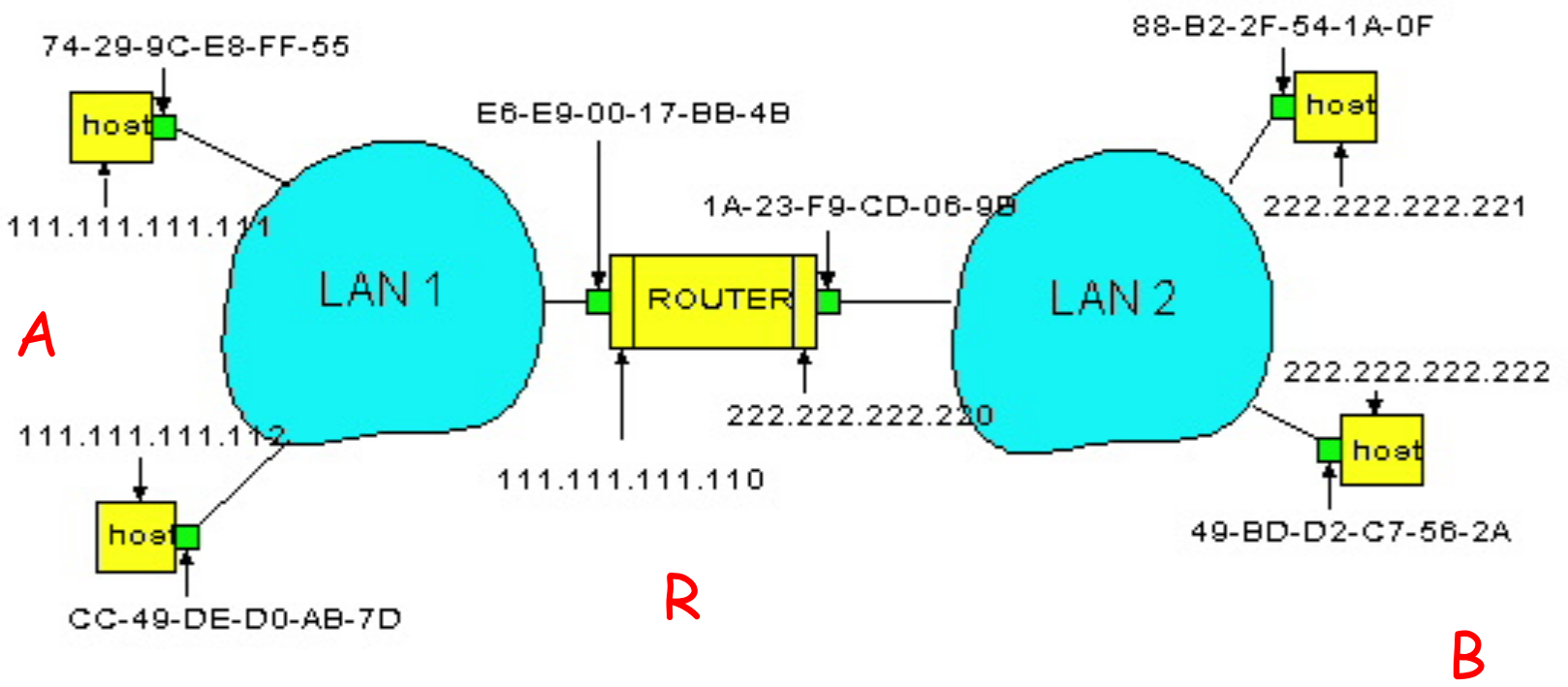
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

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 packet, containing B's IP address
  - Dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables without intervention from net administrator

Routing to another LAN

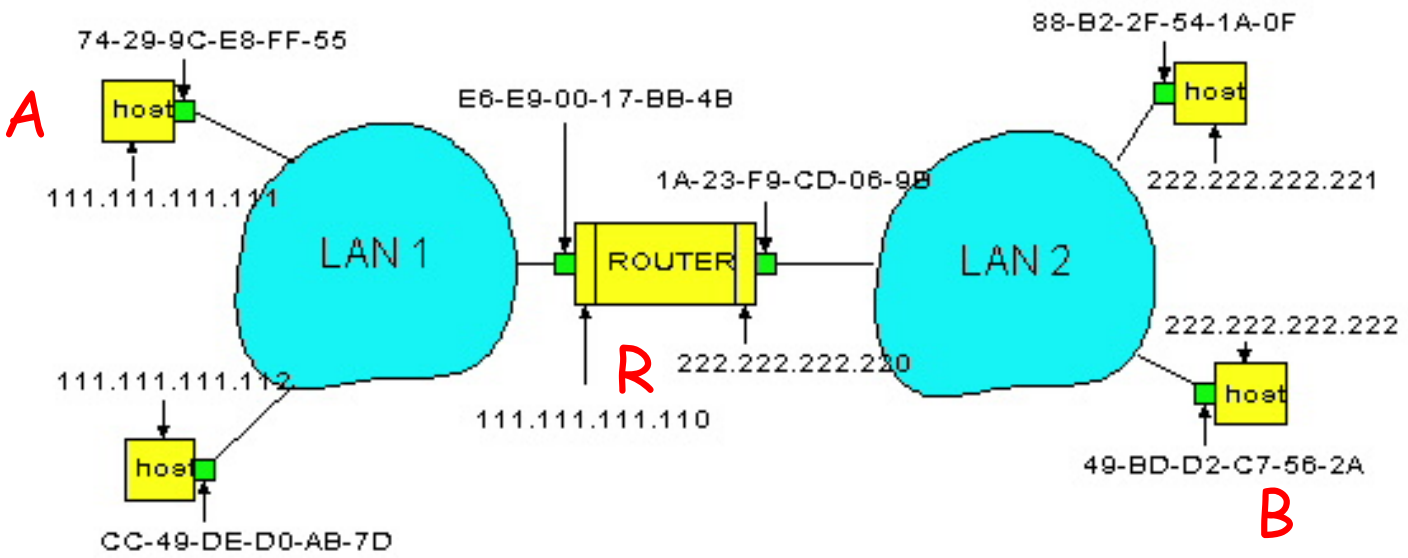
walkthrough: send datagram from A to B via R assume A knows B's IP address



- Two ARP tables in router R, one for each IP network (LAN)
- In routing table at source Host, find router 111.111.111.110
- If exists in ARP table at source, find MAC address E6-E9-00-17-BB-4B
- A creates datagram with source A, destination B

Routing to another LAN

- A uses ARP to get R's MAC address for 111.111.111.110
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
- A's adapter sends frame
- R's adapter receives frame
- R removes IP datagram from Ethernet frame, sees its destined to B
- R uses ARP to get B's MAC address
- R creates frame containing A-to-B IP datagram sends to B

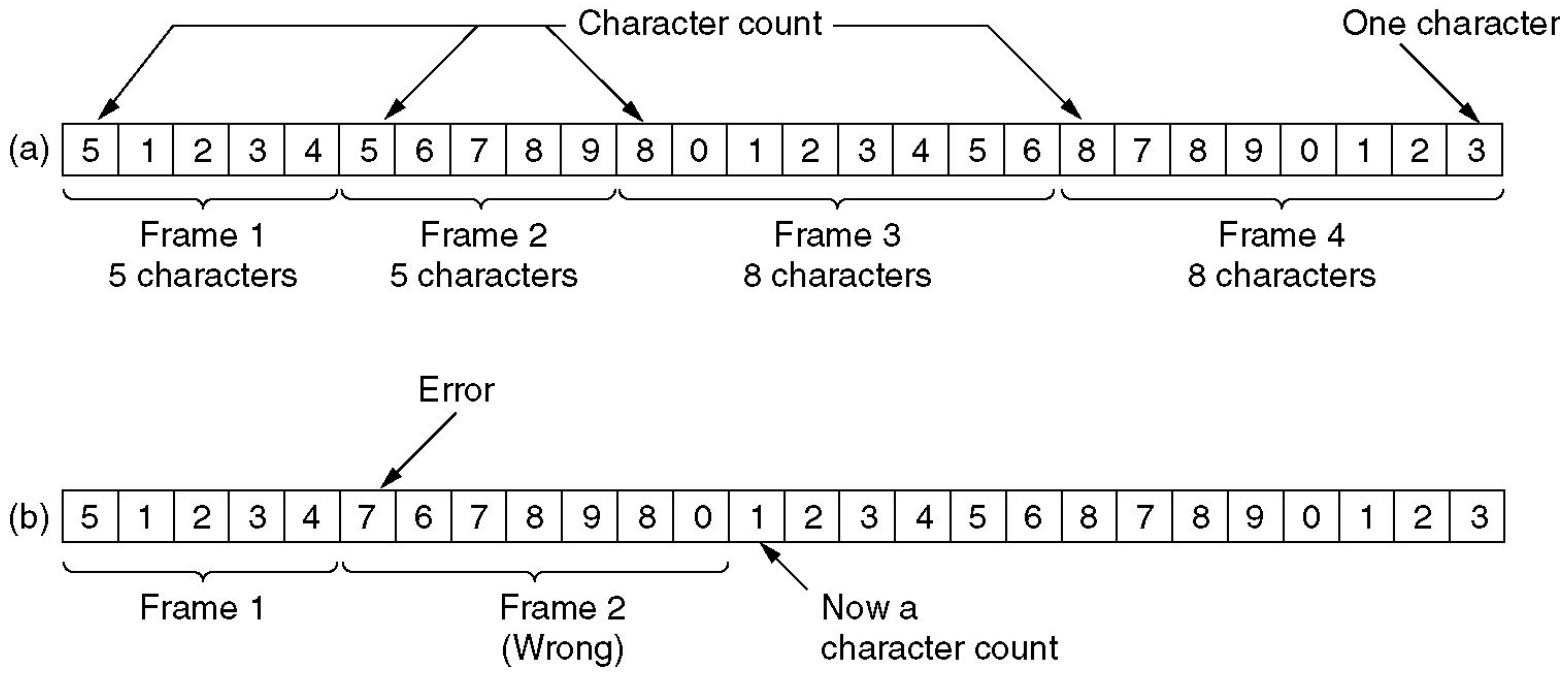


Framing

- DLL breaks bit stream into discrete frames
- Computes checksum of each frame
- Start + end of frame determination:
  - Character count
  - Start/end characters with character stuffing
  - Start/end flags with bit stuffing
  - Physical layer coding violations

Framing (Character Count)

A character stream. (a) Without errors. (b) With one error.



Problem occurs when control field is corrupted!

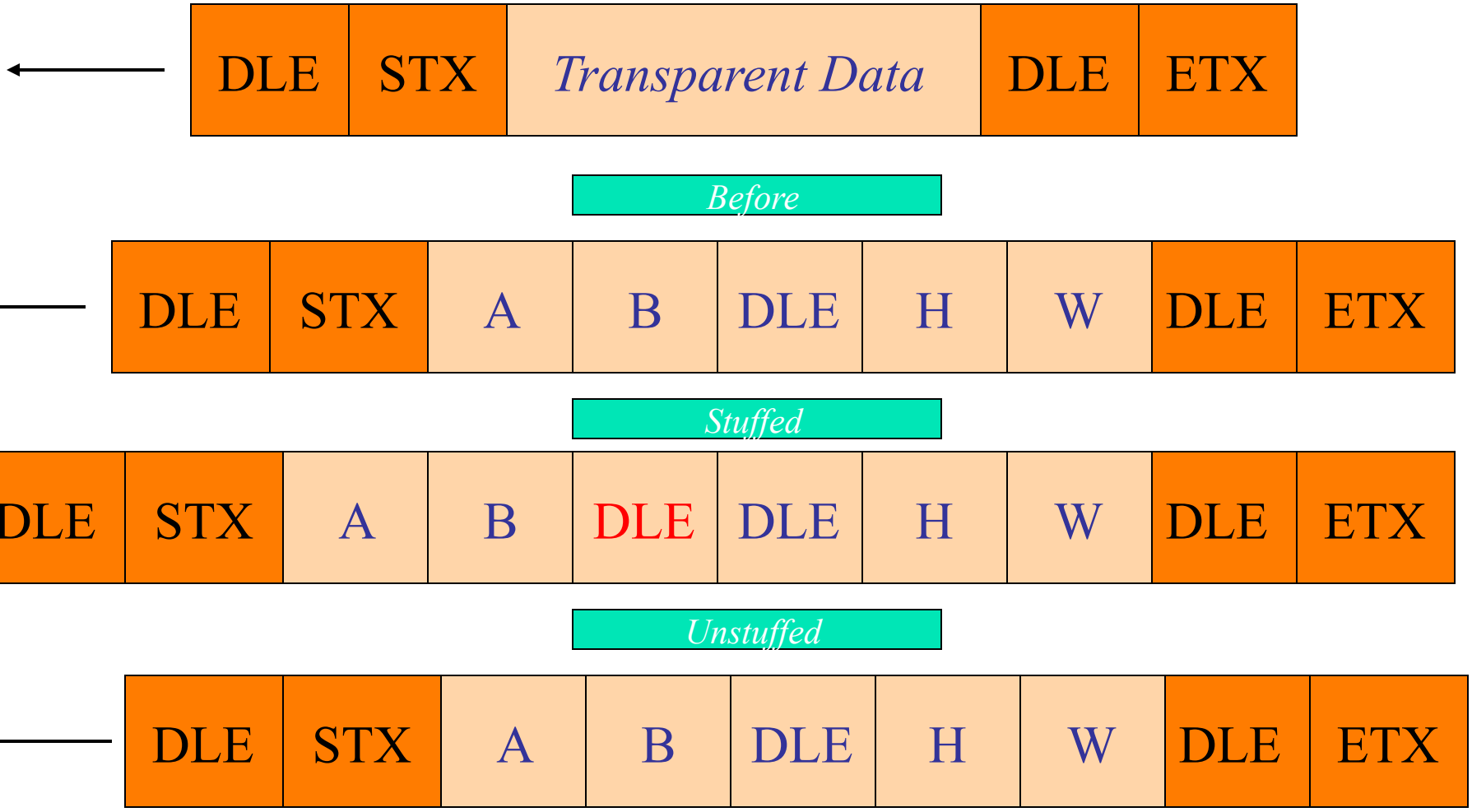
Framing (Character Stuffing)

- Also referred to as Byte stuffing.
- ASCII characters are used as framing delimiters (e.g. DLE STX and DLE ETX)
- Each frame has a special sequence of letters to
  - start the frame <DLE><STX> - Data Link Escape, Start of Text
  - end the frame <DLE><ETX> - Data Link Escape, End of Text
- The data link layer on the receiving end unstuffs the DLE before giving the data to the network layer.

The problem occurs when these character patterns occur within the data!

Solution: sender stuffs an **extra DLE** into the data stream just before each occurrence of an "accidental" DLE in the data stream.

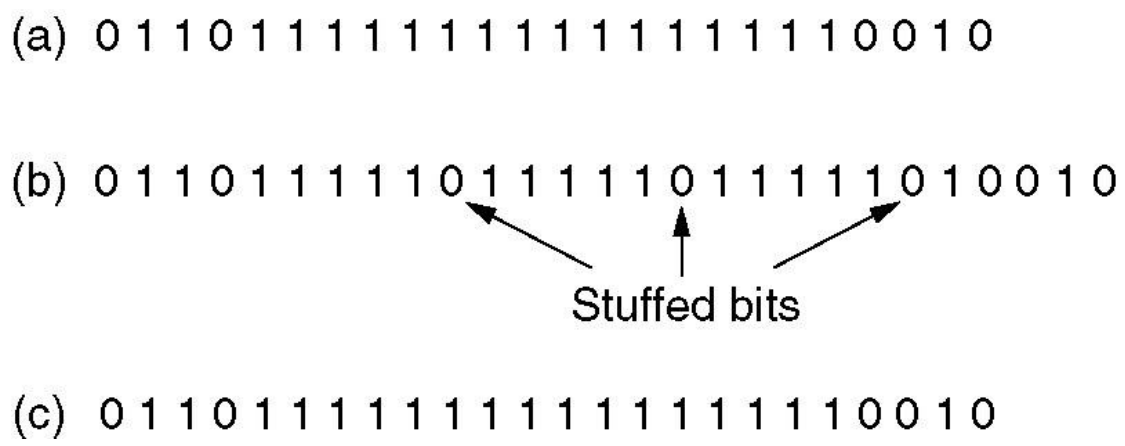
Framing (Character Stuffing)



Framing (Bit Stuffing)

- Each frame begins and ends with a special bit pattern called a flag byte [01111110].
- Whenever sender data link layer encounters *five consecutive ones* in the data stream, it automatically stuffs a 0 bit into the outgoing stream.
- When the receiver sees *five consecutive incoming ones followed by a 0 bit*, it automatically destuffs the 0 bit before sending the data to the network layer.

Framing (Bit Stuffing)



Bit stuffing

(a) The original data.

(b) The data as they appear on the line.

(c) The data as they are stored in receiver's memory after destuffing.

Error Control

- Applications require certain reliability level
  - Data applications require error-free transfer
  - Voice & video applications tolerate some errors
- Transmission errors exist
  - Single bit errors
  - Burst errors (which one is better???)
  - Lost frames vs. Damaged frames (when??)
- ***Error detection***
  - Error-detecting codes: CRC, checksum etc.
  - Would suffice (along with a retransmission-based strategy) in relatively reliable channels
- ***Error correction***
  - Error-correcting codes: Hamming codes…
  - Required for error-prone channels
- Two basic approaches:
  - Error ***detection*** & retransmission (ARQ: Automatic Repeat reQuest)
  - Forward error ***correction*** (FEC)

Error Control

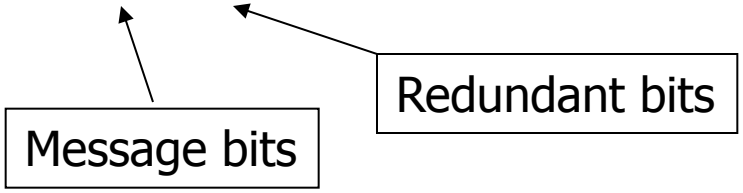
- Error rate
  - Bit error rate (BER) : probability of a transmitted bit being received wrong
    - e.g., 10^{-7} for satellite, 10^{-9} for MW, 10^{-11} for fiber
  - Packet/frame error rate
- For a given BER and frame length n

  P[frame correct]= $(1\text{-BER})^n$

  P[frame has error]=$1- (1\text{-BER})^n \cong n \times BER$

Error Detection

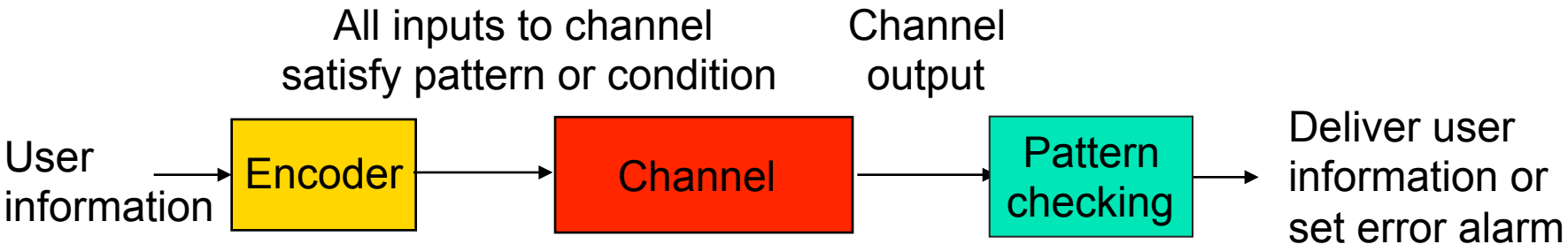
- To detect or correct errors
  - Additional (redundant) bits added by transmitter to the original data to form codewords
  - Codeword length n=m+r



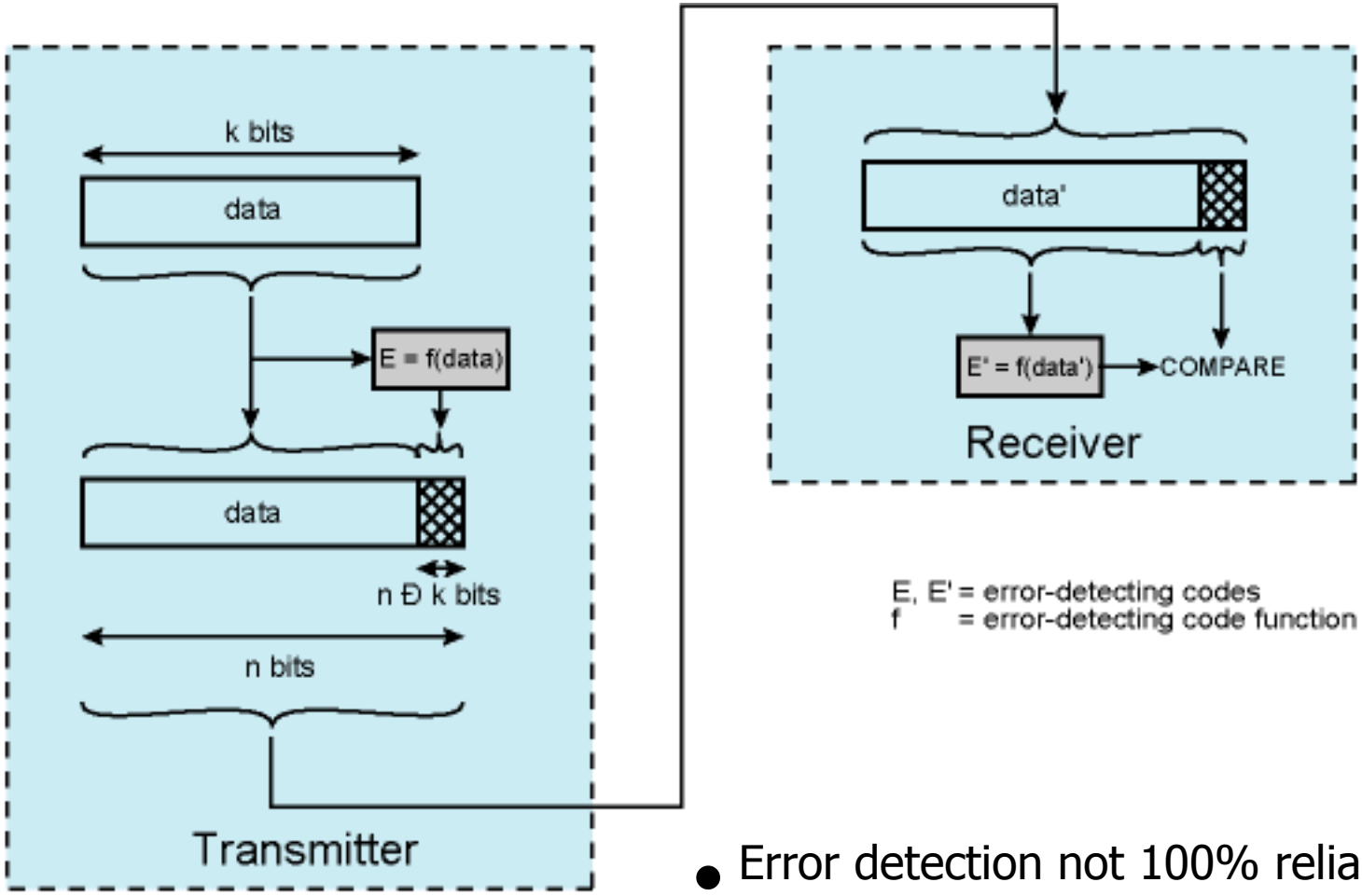
  - e.g. Parity
    - Value of parity bit is such that character has even (even parity) or odd (odd parity) number of ones

General Idea

- All transmitted data blocks ("codewords") satisfy a pattern
- If received block does not satisfy pattern, it is in error
- Redundancy: Only a subset of all possible blocks can be codewords
- Blindspot: when channel transforms a codeword into another codeword



Error Detection Process



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

Single Parity Check

- Append an overall parity check to *k* information bits

Info Bits: $b_1, b_2, b_3, \ldots, b_k$

Check Bit: $b_{k+1} = b_1 + b_2 + b_3 + \ldots + b_k \mod 2$

Codeword: $(b_1, b_2, b_3, \ldots, b_k, b_{k+1})$

- All codewords have even # of 1s
- Receiver checks to see if # of 1s is even
  - All error patterns that change an odd # of bits are detectable
- Parity bit used in ASCII code

All even-numbered error patterns are undetectable!!!

Example of Single Parity Code

- Information (7 bits): (0, 1, 0, 1, 1, 0, 0)
- Parity Bit: $b_8 = 0 + 1 + 0 + 1 + 1 + 0 = 1$
- Codeword (8 bits): (0, 1, 0, 1, 1, 0, 0, 1)

- If single error in bit 3 : (0, 1, 1, 1, 1, 0, 0, 1)
  - # of 1's =5, odd
  - Error detected

- If errors in bits 3 and 5: (0, 1, 1, 1, 0, 0, 0, 1)
  - # of 1's =4, even
  - Error not detected

Other Error Detection Codes

- Many applications require very low error rate
- Need codes that detect the vast majority of errors
- Single parity check codes do not detect enough errors
- The following error detecting codes used in practice:
  - CRC Polynomial Codes
  - Internet Check Sums (at Transport Layer)

Polynomial Codes

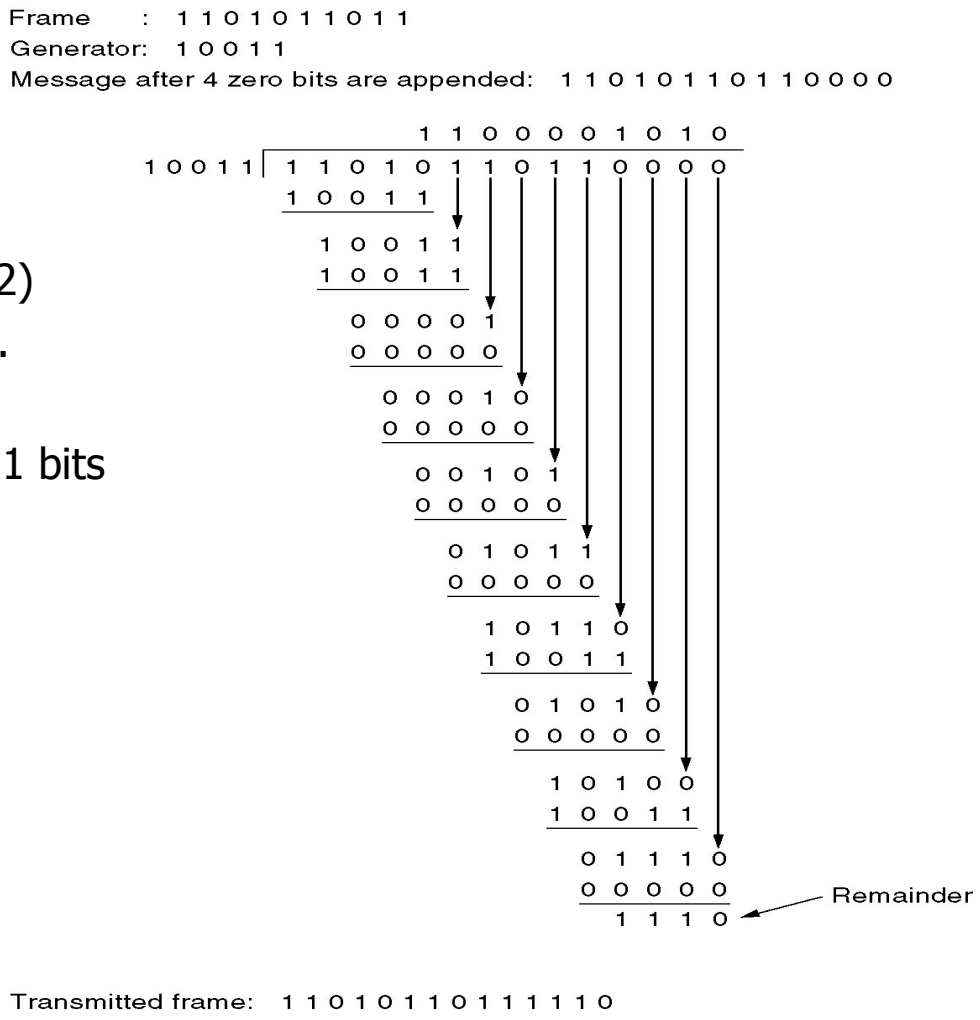
- Polynomials for codewords and polynomial arithmetic
- Implemented using shift-register circuits
- Also called *cyclic redundancy check (CRC)* codes
- Most data communications standards use polynomial codes for error detection
  - For a block of *k* bits transmitter generates *n* bit sequence
  - Transmit *k+n* bits which is exactly divisible by some number
  - Receiver divides frame by that number
    - If no remainder, assume no error

Error-Detecting Codes (CRC)

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G.
  - If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice (ATM, HDLC)

Divide $D \cdot 2^r$ by G, get remainder R, and transmit <D,R>

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



Standard Generator Polynomials

CRC = cyclic redundancy check

- CRC-8:

  $= x^8 + x^2 + x + 1$ — ATM

- CRC-16:

  $= x^{16} + x^{15} + x^2 + 1$
  $= (x + 1)(x^{15} + x + 1)$ — Bisync

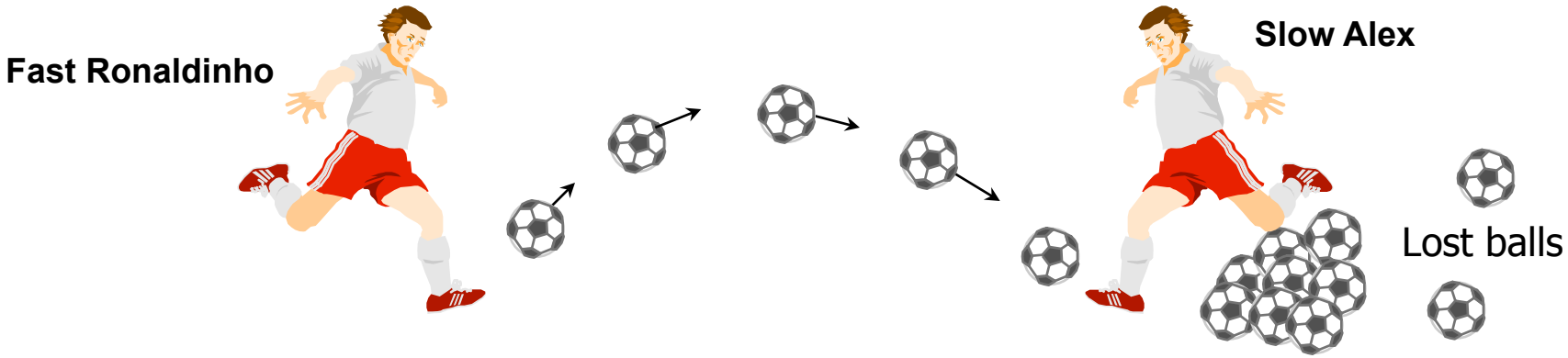
- CCITT-16:

  $= x^{16} + x^{12} + x^5 + 1$ — HDLC, XMODEM, V.41

- CCITT-32: — IEEE 802, DoD, V.42

  $= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Flow Control

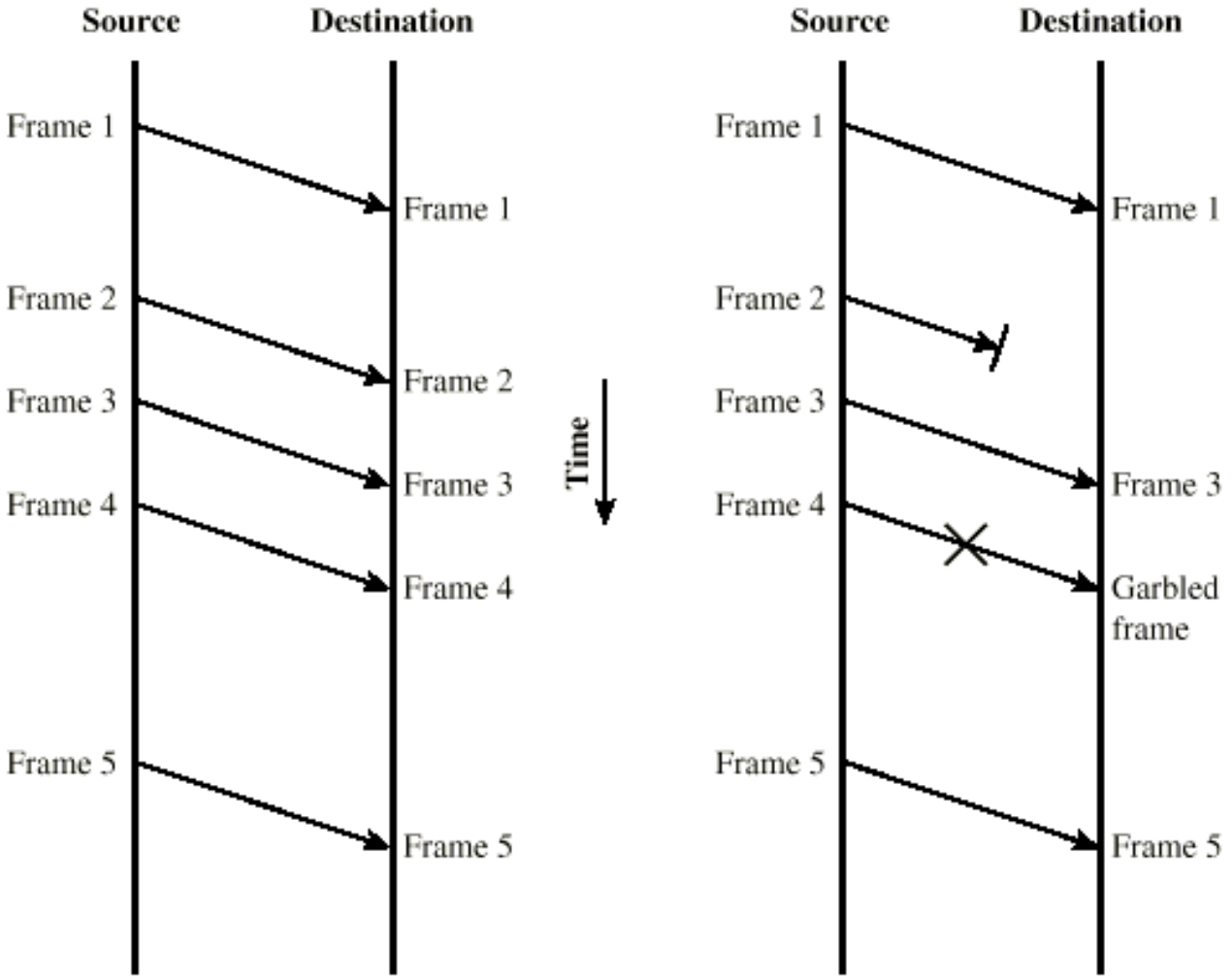


- What to do with a sender that wants to transmit frames faster than the receiver can accept them ???
- Even if transmission is error free, the receiver may be unable to handle the frames as they arrive and lose
  - Might be possible for the sender to simply insert a delay to slow down sufficiently to keep from swamping the receiver
- Two approaches for flow control
  - **Feedback-based flow control:** the receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing.
  - **Rate-based flow control:** the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

Flow Control

- Ensuring the sending entity does not overwhelm the receiving entity
  - Preventing buffer overflow
- Transmission time (t_{frame})
  - Time taken to emit all bits into medium
- Propagation time (t_{prop})
  - Time for a bit to traverse the link

Model of Frame Transmission



(a) Error-free transmission

(b) Transmission with losses and errors

Some Flow Control Algorithms

- Simplex protocols
  - Flow control for the ideal network
  - Stop and Wait for noiseless channels
  - Stop and Wait for noisy channels
- Full duplex protocols
  - Sliding window with Go Back N
  - Sliding window with Selective Repeat

Simplex Flow Control

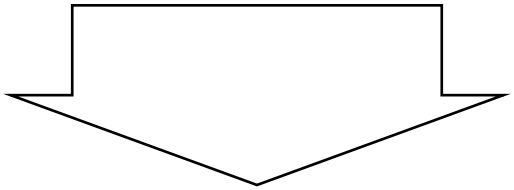
- Data only flows in one direction
- Acknowledgement (ACK) stream may flow in the other direction

Flow control in the ideal network:

Assumptions:

Error free transmission line,

Infinite buffer at the receiver



No acknowledgment of frames necessary

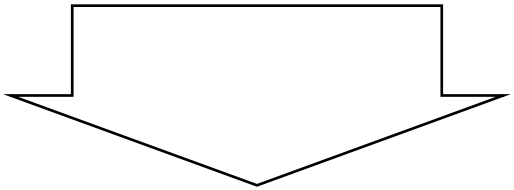
Since the transmission line is error-free and the receiver can buffer as many frames as it likes, no packet will ever be lost

Stop and Wait with Noiseless Channels

Assumptions:

Error free transmission line,

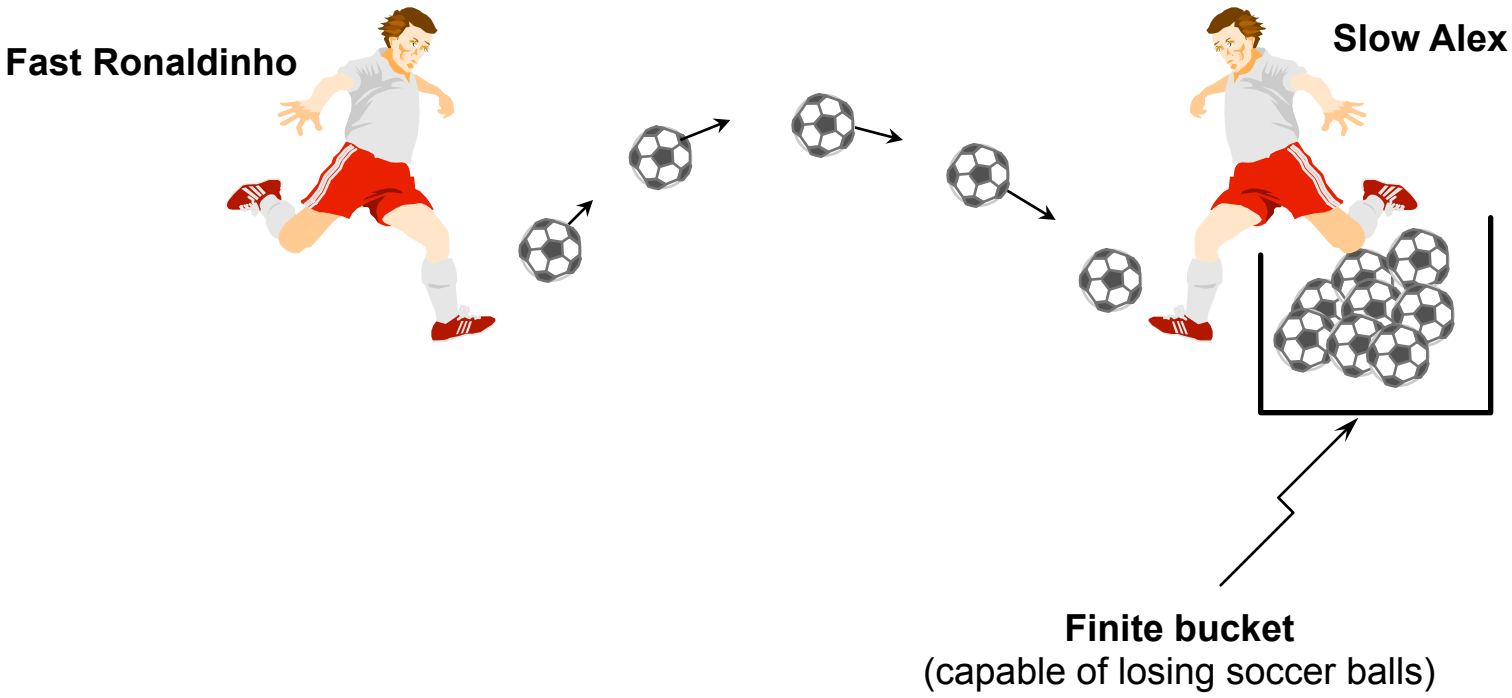
Finite buffer at the receiver



Problem of Buffer overflow at the receiver

- Buffer overflow may happen at the receiver when the sender sends frames at a rate faster than the receiver

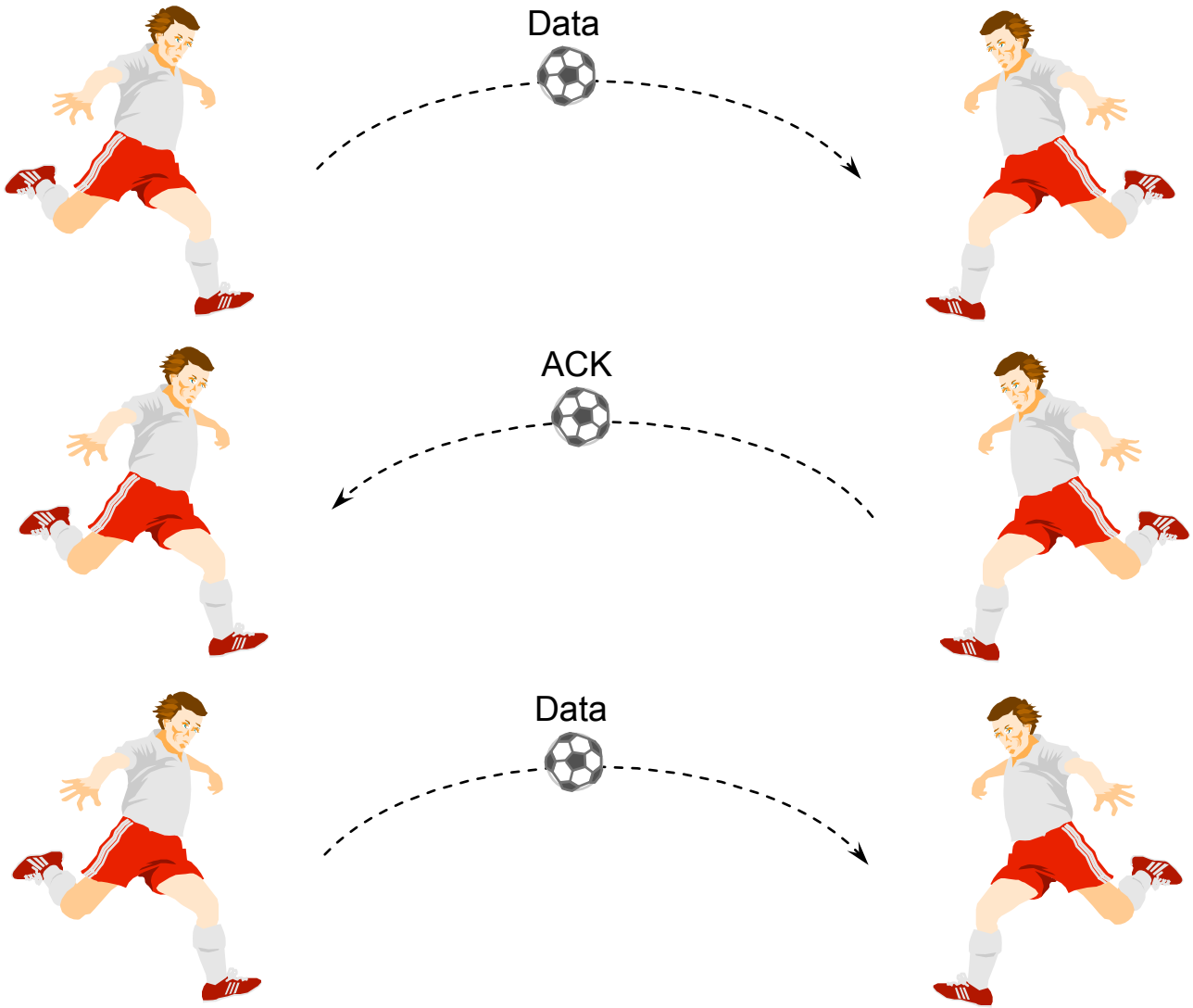
Stop and Wait with Noiseless Channels



Stop and Wait

- Source transmits frame
- Destination receives frame and replies with ACK
- Source waits for ACK before sending next frame
- Destination can stop flow by not sending ACK
- Works well for a few large frames

Stop and Wait with Noiseless Channels



Fragmentation

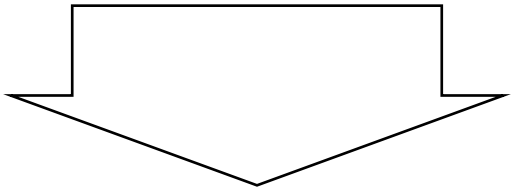
- Large block of data may be split into small frames
  - Limited buffer size
  - Errors detected sooner (when whole frame received)
  - On error, retransmission of smaller frames is needed
  - Prevents one station occupying medium for long periods
- Stop and wait becomes inadequate

Stop and Wait for Noisy Channels

Assumptions:

Transmission line may cause errors,

Finite buffer at the receiver



ACK frames may now be lost

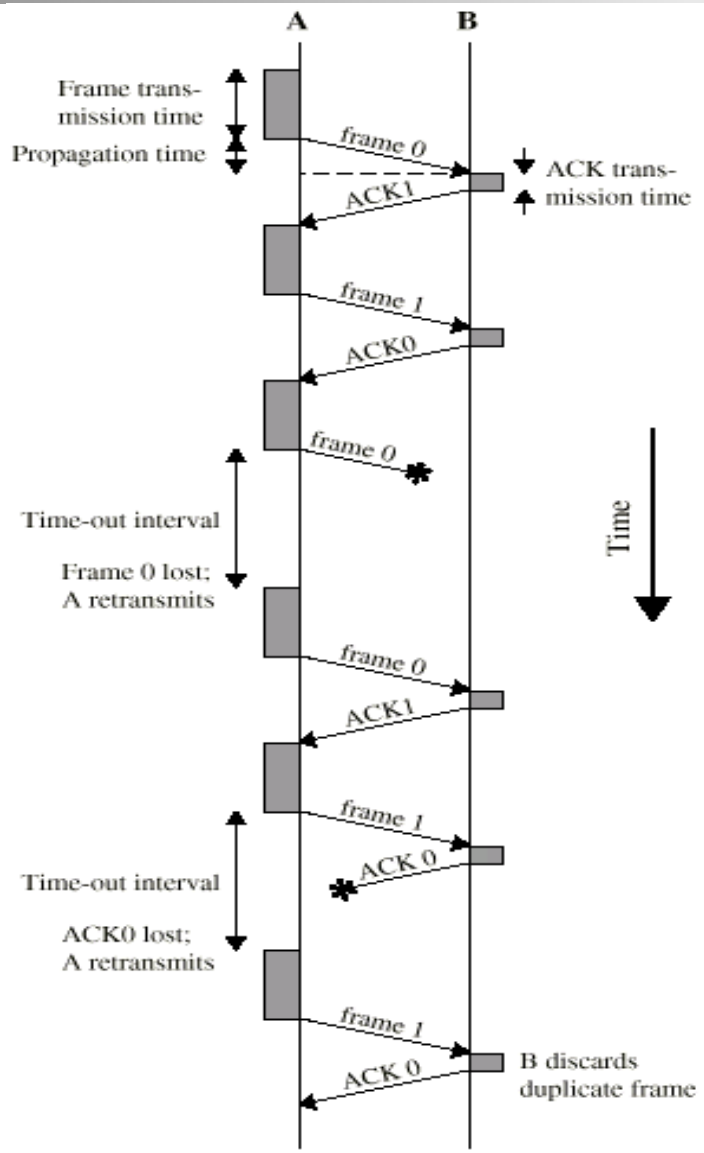
Stop and Wait

- Source transmits single frame
- Keep timeout for every frame transmitted
- Wait for ACK
- If received frame damaged, discard it
  - Transmitter has timeout
  - If no ACK within timeout, retransmit
- If ACK damaged/lost, transmitter will not recognize it
  - Transmitter will retransmit
- If receiver gets two copies of frame (duplicate frames) → discards
  - Use ACK0 and ACK1
- Use 1-bit sequence number (mod 2)

Stop and Wait Operation

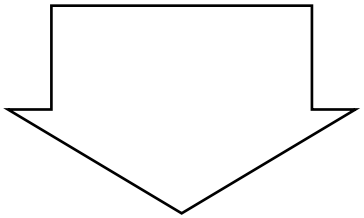
- Simple
- Inefficient

Automatic Repeat reQuest (ARQ) protocol!



Full Duplex Flow Control Protocols

Data frames are transmitted in both directions



Sliding Window

Flow Control Protocols

Sliding Window Flow Control

- Allow multiple frames to be in transit
- Receiver has buffer W long (receiver window)
- Transmitter can send up to W frames without ACK (sender window)
- Each frame is numbered (sequence number)
- ACK includes number of next frame expected
  - Ack for frame n = I am expecting frame n+1
    (not "I received fame n")
- Sequence number bounded by size of field
  - e.g. k bits: Frames are numbered modulo 2^k

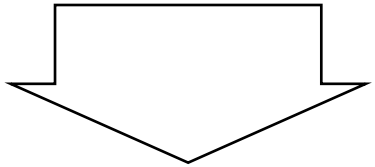
Sliding Window Protocols
Definitions

- Sequence Number: Each frame is assigned a sequence number that is incremented as each frame is transmitted
- Sender's Window: Keeps sequence numbers of frames that have been sent (or can be sent) but not yet acknowledged
- Sender Window size: The number of frames the sender have already sent and may transmit before receiving ACKs
- Receiver's Window: Keeps sequence numbers of frames that the receiver is allowed to accept
- Receiver Window size: The maximum number of frames the receiver may receive out of order

- The sending and receiving windows do not have to be the same size
- Any frame which falls outside the receiving window is discarded at the receiver

Sliding Window Protocols
Piggybacking Acknowledgements

Since we have full duplex transmission, we can "piggyback" an ACK onto the header of an outgoing data frame to make better use of the channel

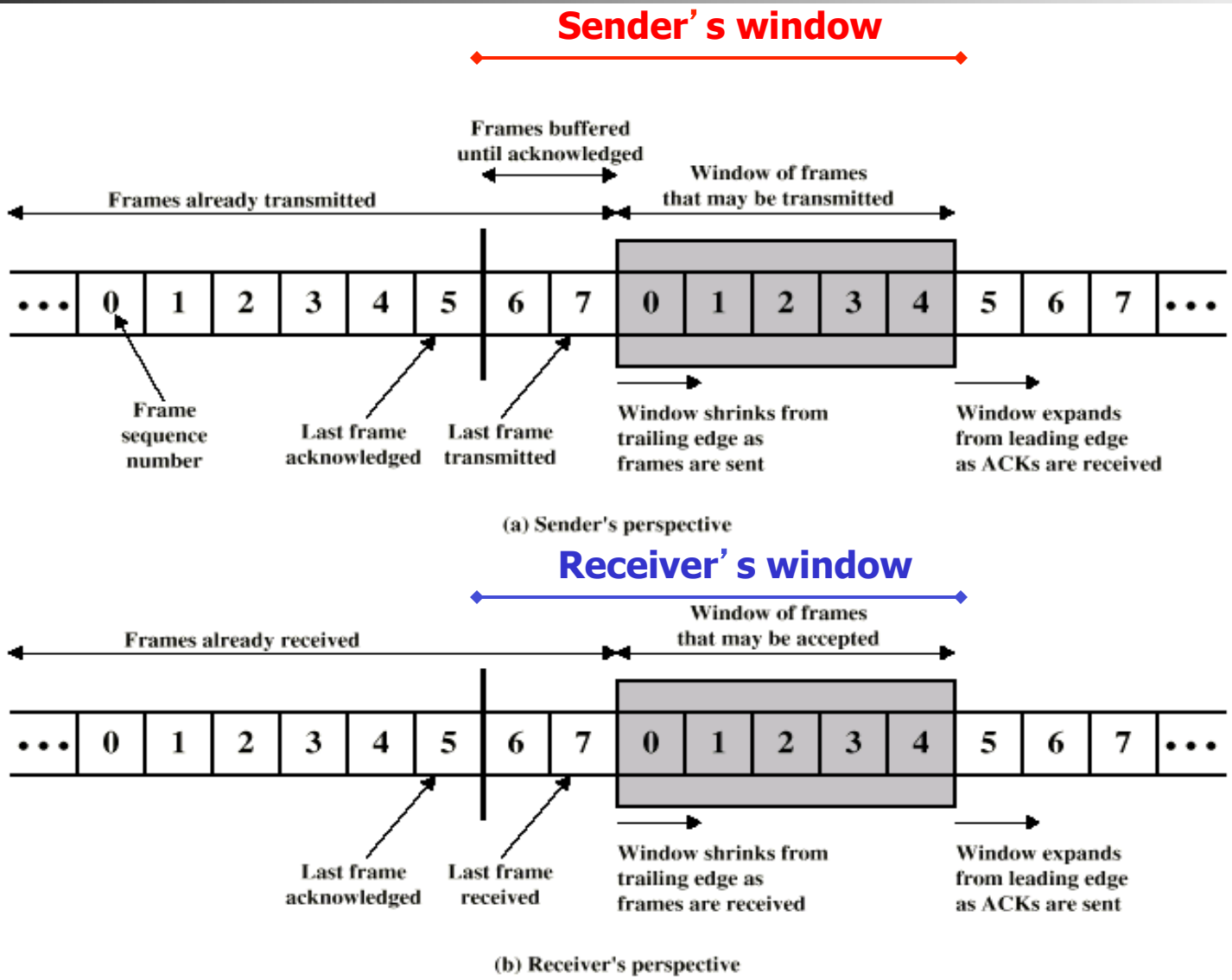


When a data frame arrives, instead of immediately sending a separate ACK frame, the receiver waits until it is passed the next data frame to send. The ACK is attached to the outgoing data frame.

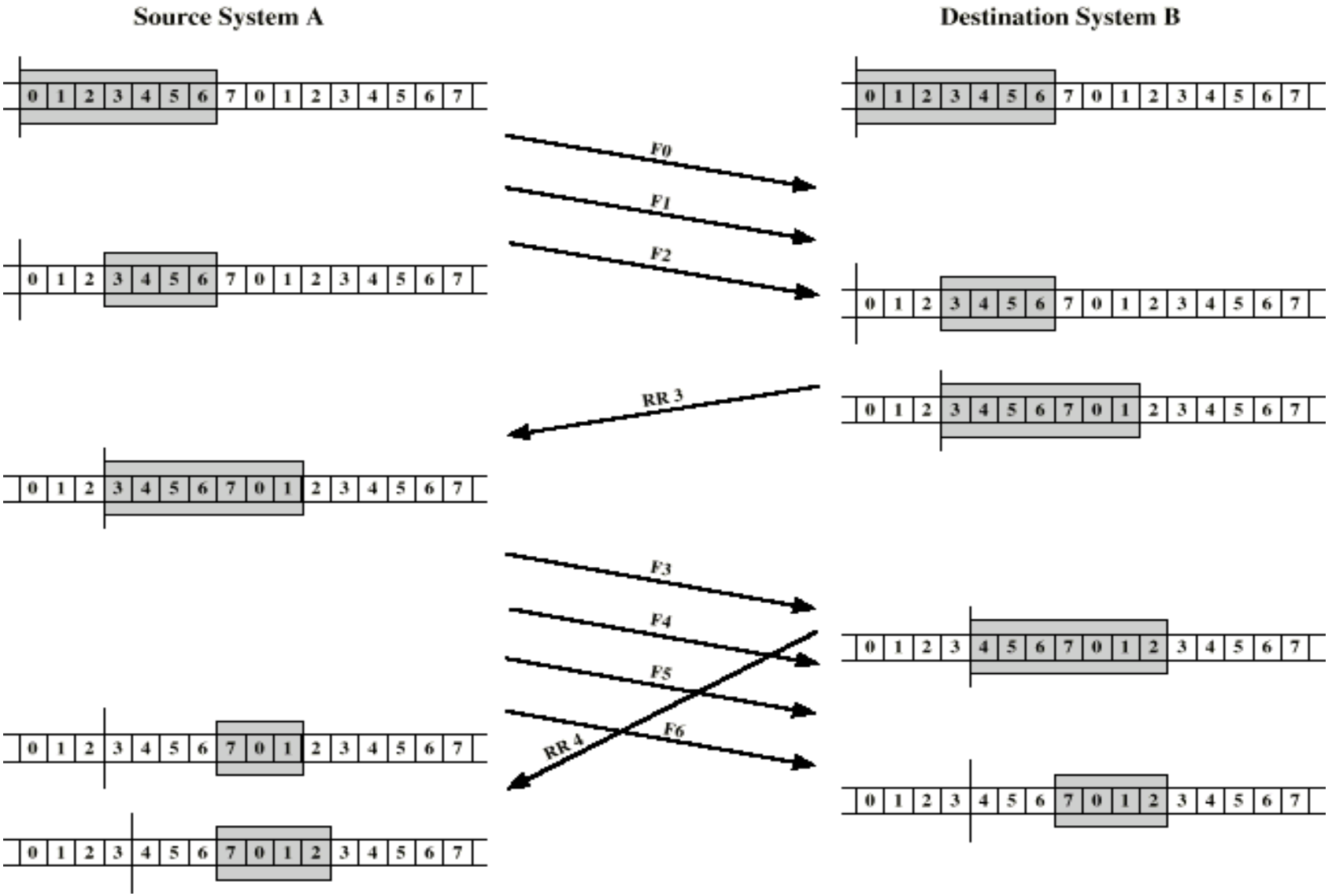
Sliding Window with Window Size W

- With a window size of 1
  - the sender waits for an ACK before sending another frame
  - This protocol behaves identically to stop and wait for a noisy channel!
- With a window size of W, the sender can transmit up to W frames before "being blocked"
- We call using larger window sizes **Pipelining**

Sliding Window Diagram

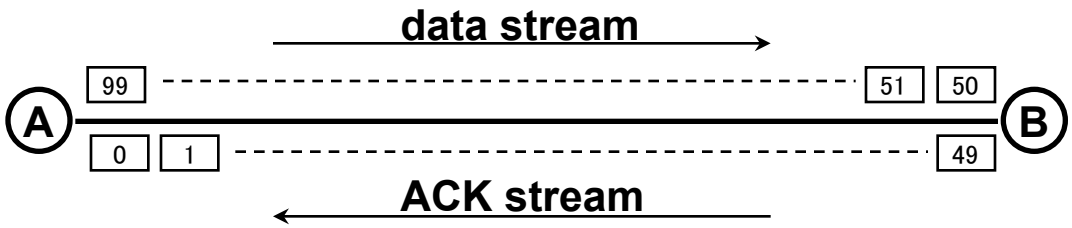


Example Sliding Window



Why do Pipelining?

In other words, why have a window size greater than 1?



By allowing several frames into the network before receiving ACK, pipelining keeps the transmission line from being idle !

Sliding Window Flow Control

- Example: (Large bandwidth x delay channels)
  50Kb/s satellite link, RTT=500ms, frame=1000 bits

  t=0 send
  t=20 frame sent
  t=270 frame is at receiver
  t=520 ACK is back

  → Efficiency=20/520
  In general, channel= b b/s, frame=l, RTT=R
  → frame time=l/b R: send and receive ACK
  → Efficiency= (l/b)/(l/b)+R=l/(l+bR)
  bR : bandwidth x delay (bandwidth-delay product)

  Fill the pipe → pipelining to increase utilization!

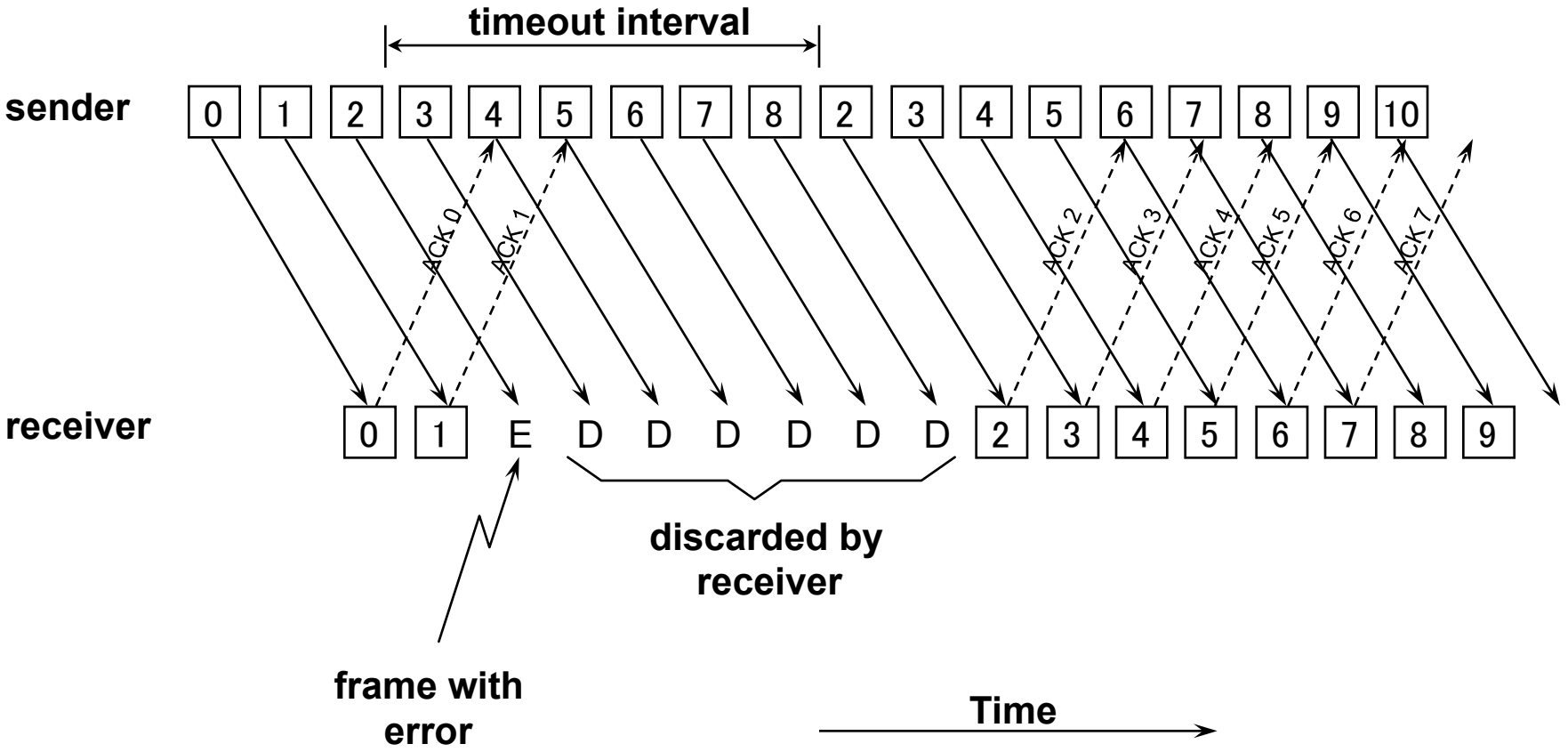
Automatic Repeat Request (ARQ)

- Stop and Wait
- What if a data or ACK frame is lost when using a sliding window protocol?
  - Sliding Window with Go back N
  - Sliding Window with Selective Repeat (Selective Reject or Retransmission)

Sliding Window with Go Back N

- Based on sliding window
- If no error, ACK as usual with next frame expected
- Use window to control number of outstanding frames
- If error, reply with rejection
  - Discard that frame and all future frames until error/lost frame received correctly
  - Transmitter must go back and retransmit that frame and all subsequent frames

Go Back N



- Assume ACK per frame by receiver (receiver window=1)
- Go Back N can recover from erroneous or missing packets
- Inefficient → if there are a lot of errors, the sender will spend most of its time retransmitting

Go Back N

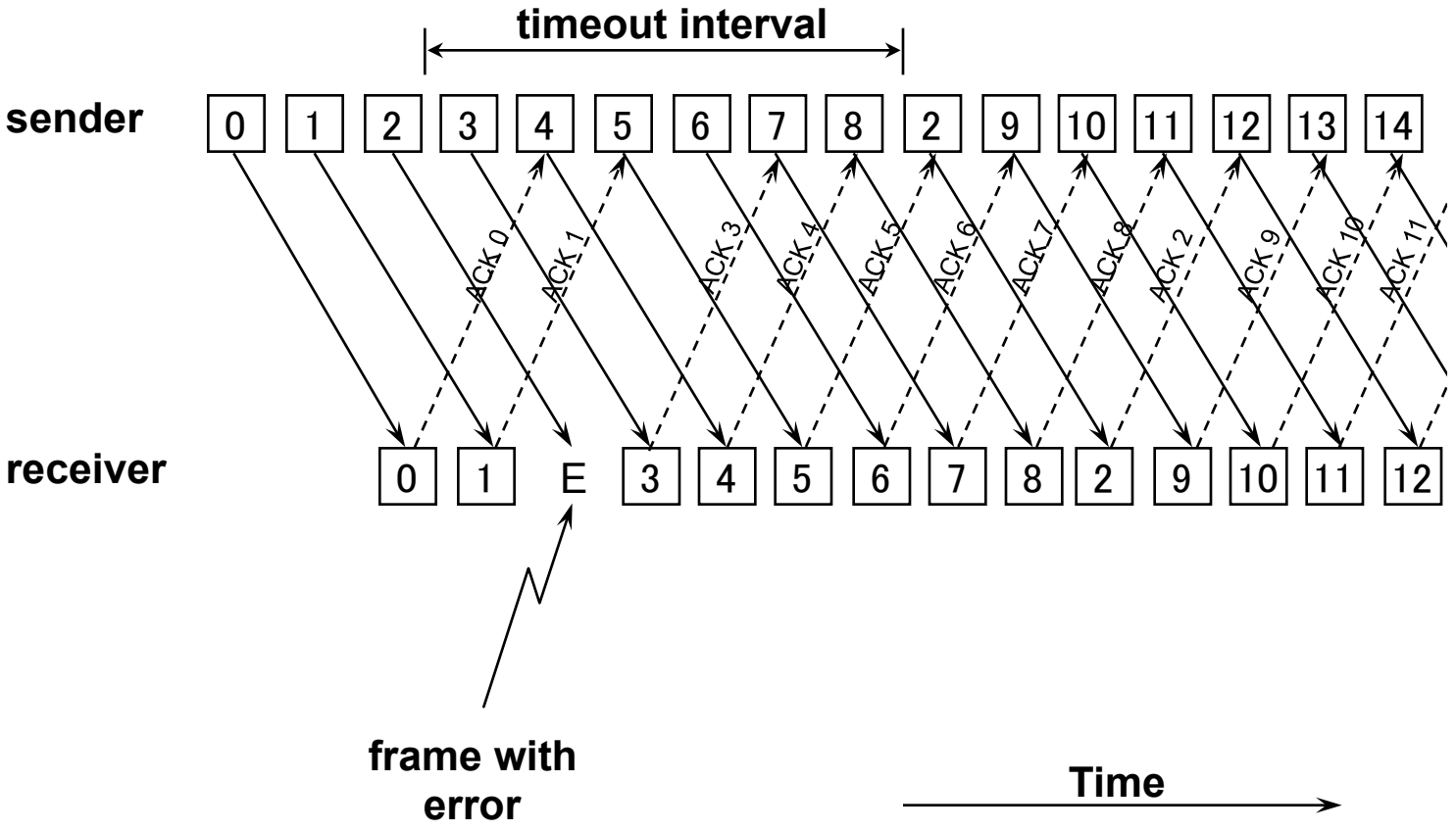
- Consider the scenario with MAX_SEQ = 7
  - Sender sends frames 0 through 7
  - A piggybacked ACK for frame 7 comes back to sender
  - Sender sends another 8 frames, again 0 through 7
  - Now another piggybacked ACK for frame 7 comes in
- Did all eight frames belonging to the second batch arrive successfully, or did all eight get lost (counting discards following an error as lost)?
  - In both cases the receiver would be sending frame 7 as ACK
  - The sender has no way of telling

Maximum of MAX_SEQ frames and not MAX_SEQ + 1 frames may be outstanding at any instant, even though there are MAX_SEQ + 1 distinct sequence numbers: 0, 1, 2, ..., MAX_SEQ.

Sliding Window with Selective Repeat

- The sender retransmits only the frame with errors
- The receiver stores all the correct frames that arrive following the bad one (receiver window > 1)
- Requires a significant amount of buffer space at the receiver
- When the sender notices that something is wrong, it just retransmits the one bad frame, not all its successors
- Might be combined with Negative Acknowledgment (NACK)

Selective Repeat



Selective Repeat with NACK