Lecture 3 - Data Link Layer

Computer Communication Networks
CS35201 - (001/ 002)
Fall 2022

Kent State University

Department of Computer Science
Betis Baheri
bbaheri@kent.edu

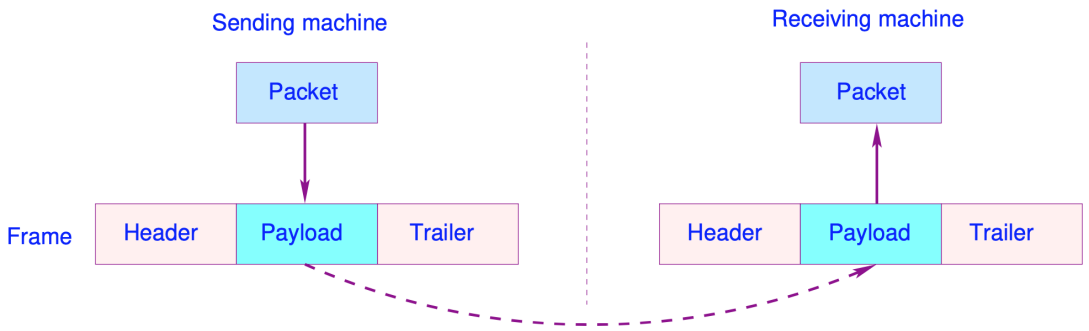
September 26, 2022

Outline I

Data Link Layer Design Issues

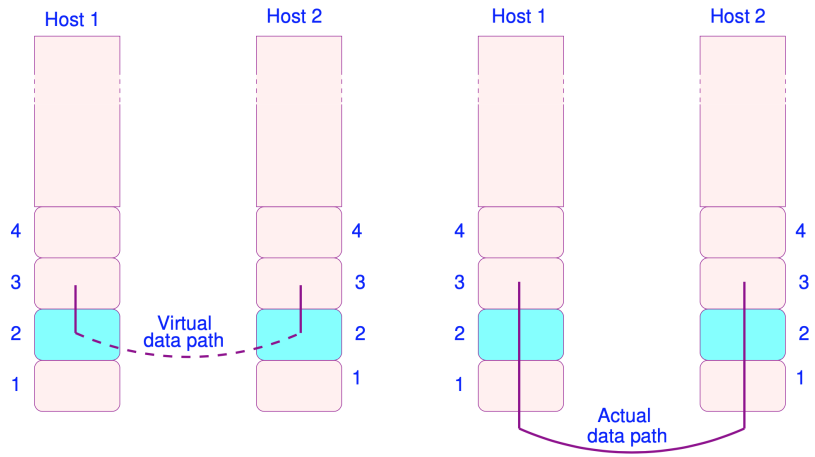
1. Network Layer Services
2. Framing
3. Error Control
4. Flow Control

Packets vs. Frames



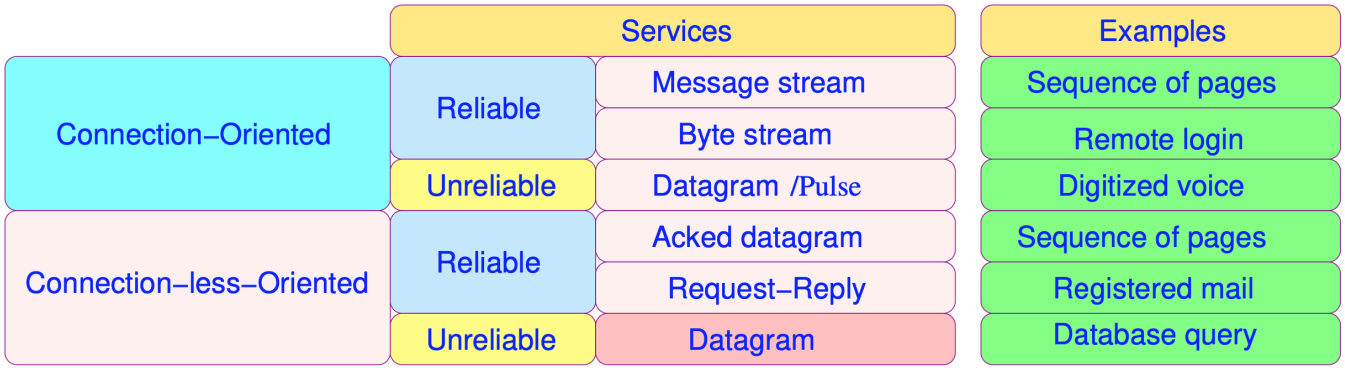
- Packets
  - Layer 3 data units
  - Mostly variable size $\Rightarrow 20 \leq$ IP $\leq$ 65,535 bytes $=$ 64k bytes
  - Carry routing information
- Frames
  - Layer 2 data units
  - Fix size $\Rightarrow$ Ethernet 1500 bytes, WiFi 2346 bytes
  - No or minimal routing information $\Rightarrow$ forwarding

Virtual vs. Actual Connections



- Layer 3 doesn't know how layer 2 delivers its packets
- Layer 2 and 4 are designed to provide reliable connections
  - Connection-oriented
  - Acknowledgements
  - Request-Reply
- Layer 3 is designed to provide best effort delivery
  - Connection-less-oriented
  - Unreliable

Possible Services Offered



1. Unacknowledged connectionless service
   - User Data Protocol (UDP)
2. Acknowledged connectionless service
   - Ethernet frames, Internet Protocol (IP)
3. Acknowledged connection-oriented service
   - Transport Control Protocol (TCP)

Outline I

How can reliability be provided?

1. Acknowledgment,
2. Partial acknowledgment (cumulative acknowledgment) $\Rightarrow$ TCP
3. Automatic-repeat and request (ARQ)
4. Byte-stream
5. Message sequencing

What is the Problem?

When two devices exchange data:

- Data transmitted one bit at a time
- Transmitting and receiving data rates must be the same The transmitter/receiver must recognize the start/end of each bit
  - To make sure sampling can be done correctly
  - Also, a drift in time results errors

Two Approaches: Framing Methods

1. Asynchronous data transmission
2. Synchronous data transmission

Transmission Methods

1. Asynchronous data transmission
   - Timing problem is avoided by not sending long streams of bits
   - Data is transmitted one character at a time
     - Byte (character-oriented) protocols
   - Synchronization bits are required
     - Bits are synchronized, data is not
   - Good for low-rate channels

2. Synchronous data transmission
   - Bit-oriented protocols
   - Transmitter and receiver clocks must be synchronized
     - Using separate timing circuit $\Rightarrow$ out-of-band signaling
     - Manchester coding
   - There is a minimum frame length

Framing

What is a Frame?

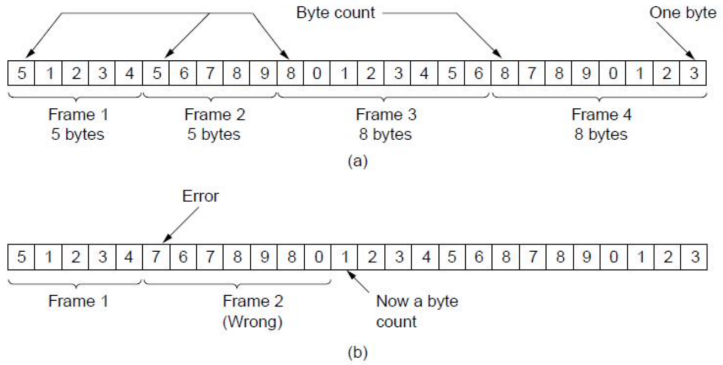
- A basic transmission unit
- Typically implemented by network adapter
- Adapter fetches/deposits frames out/in to memory
- The receiver must determine first and last bit of the frame

Basic Framing Techniques

1. Byte count
2. Flag bytes with byte stuffing
3. Flag bits with bit stuffing
4. Physical layer coding violations
5. Clock-based Framing $\Rightarrow$ SONET

Framing Methods I

1. Byte Count Framing ⇒ Byte-Oriented Protocols
   - Each frame starts with a byte that contains the frame size (in bytes)
   - What would happen when an error occurs in a byte count?
     - Synchronization is lost
     - Correct start/end of the frames will be lost
     - Even if the checksum detects such an error, it cannot correct it
   - A byte stream. (a) Without errors. (b) With one error



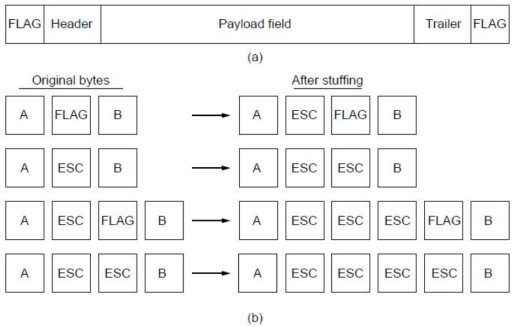
   - Example: Digital Data Communication Message Protocol (DDCMP)

8	8	8	14		16
SYN	SYN	Class	Header	Body	CRC

Framing Methods II

2 Framing with Flags

- A frame delimited by flag bytes
- What would happen if the *FLAG* appears in the text?
  - Use a escape character
  - Four examples of byte sequences before and after byte stuffing.

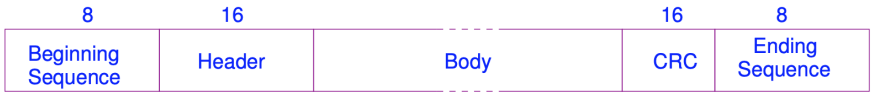


- Example: Point-to-Point Protocol (PPP)

- Example: BISYNC (Binary sync. comm.) ⇒ IBM

Framing Methods III

3. Bit-oriented protocols
   - High-Level Data Link Control (HDLC) (OSI)
     - Synchronous Data Link Control (SDLC) - IBM



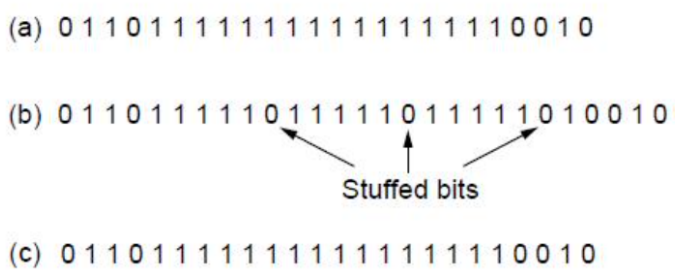
   - Delineate frame with a special bit-sequence: 01111110

Bit Stuffing Protocol

- Sender: any time five consecutive 1s have been transmitted from the body of the message, insert a 0.
- Receiver: should five consecutive 1s arrive, look at next bit(s):
  - if next bit is a 0: remove it $\Rightarrow$ un-stuff it
  - if next bits are 10: end-of-frame marker
  - if next bits are 11: error

Framing Methods IV

Example 3.1 (Bit Stuffing)



(a) The original data. (b) The data as they appear on the line
(c) The data stored in the receiver's memory after de-stuffing

Other Bit-Oriented Protocols

- Link Access Procedure (LAP) $\Rightarrow$ CCITT
- Link Access Procedure - Balanced (LAPB) $\Rightarrow$ ITU-T (for X.25)
- Link Access Procedure for the D channel (LAPD) $\Rightarrow$ ITU-T (for ISDN)
- Hewlett Packard Data Link Control (HPDLC: HP)
- Advanced Data Communications Control Procedures (ADCCP) - ANSI
- Logical Link Control (LLC) $\Rightarrow$ IEEE 802.2

Outline I

Error Detection

- Reliability requires error detection and possibly correction
  - We cannot correct all the time
  - Two many bit errors may not be correctable
- Many techniques
  - From dictionary search to in-line (real-time) detection and possibly correction
  - Not all good equally for different noise (error) rate
- In general we extra bits to the data bits for the purpose of:
  - Detecting errors
  - Correcting errors if possible
- We start with simple techniques, and we cover more robust techniques
  - Used by NASA for various missions

Major Error Detection Techniques

1. Parity
2. Checksum
3. Cyclic Redundancy Checks

Parity Bit

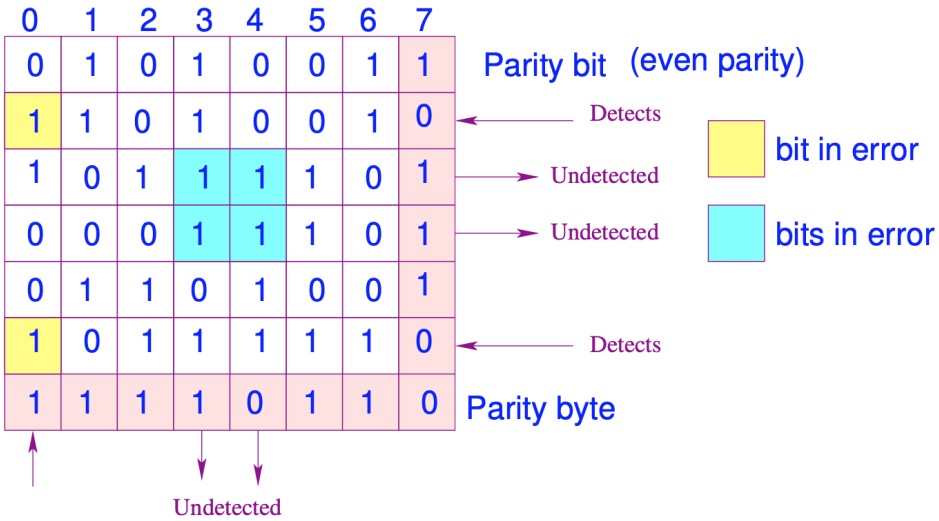
- Detecting a single bit error $\Rightarrow$ parity bit
  - Memory read/write (R/W) $\Rightarrow$ 8 data bits + 1 parity bit

0	1	2	3	4	5	6	7	
1	0	1	1	0	1	1		
1	0	1	1	0	1	1	0	odd parity, odd number of 1's
1	0	1	1	0	1	1	1	even parity, even number of 1's

  - Even number of bit errors goes undetected $\Downarrow$
    - Keeps the parity bit unchanged
- In general, we decode m bits to $n = m + r$ bits $\Rightarrow$ codewords
  - r bits overhead $\Rightarrow$ next
- Odd/even parity is weak $\Rightarrow$ 50% of errors cannot be detected

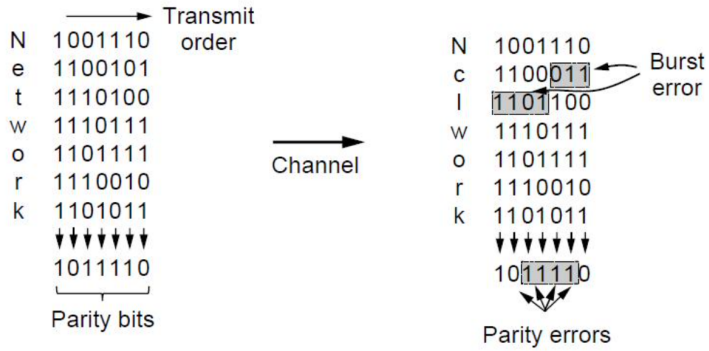
Two-Dimensional Parity Check

- Multiple bytes are sent in matrix form $\Rightarrow$ blocks
- Each bit of data is checked by two parity bits
  - Row and Column
- What happens if the parity itself received in error?
- 2D parity detects better but still misses a lot of errors undetected



Interleaving Parity Bits

- Errors often occur in bursts as a result of:
  - Transit-time noise ⇒ unpredictable
  - Random statistical fluctuations of the electric current ⇒ shot noise ⇒ unpredictable
  - ...
- Interleaving parity bit with data bits allows spread the bits
  - Better detection, and also help correction
- Data is horizontal, parity is vertical ⇒ scrambles the bits



- Used in fiber optics ⇒ minimal computations

Checksum

- Simple, but weak
- The sum of bytes computed as parity bits $\Rightarrow$ 1's complement

Data	Checksum	Data	Checksum
0001	1	0011	3
0010	2	0000	0
0011	3	0001	1
0001	1	0011	3
Total	7	Total	7

- Does not detect all common erros $\Rightarrow$
  weak error detection (used in IP)
- Used in networks where speed is more important than errors
  - Other layers take care of error correction/detection

Cyclic Redundancy Check (CRC) I Data Link Layer

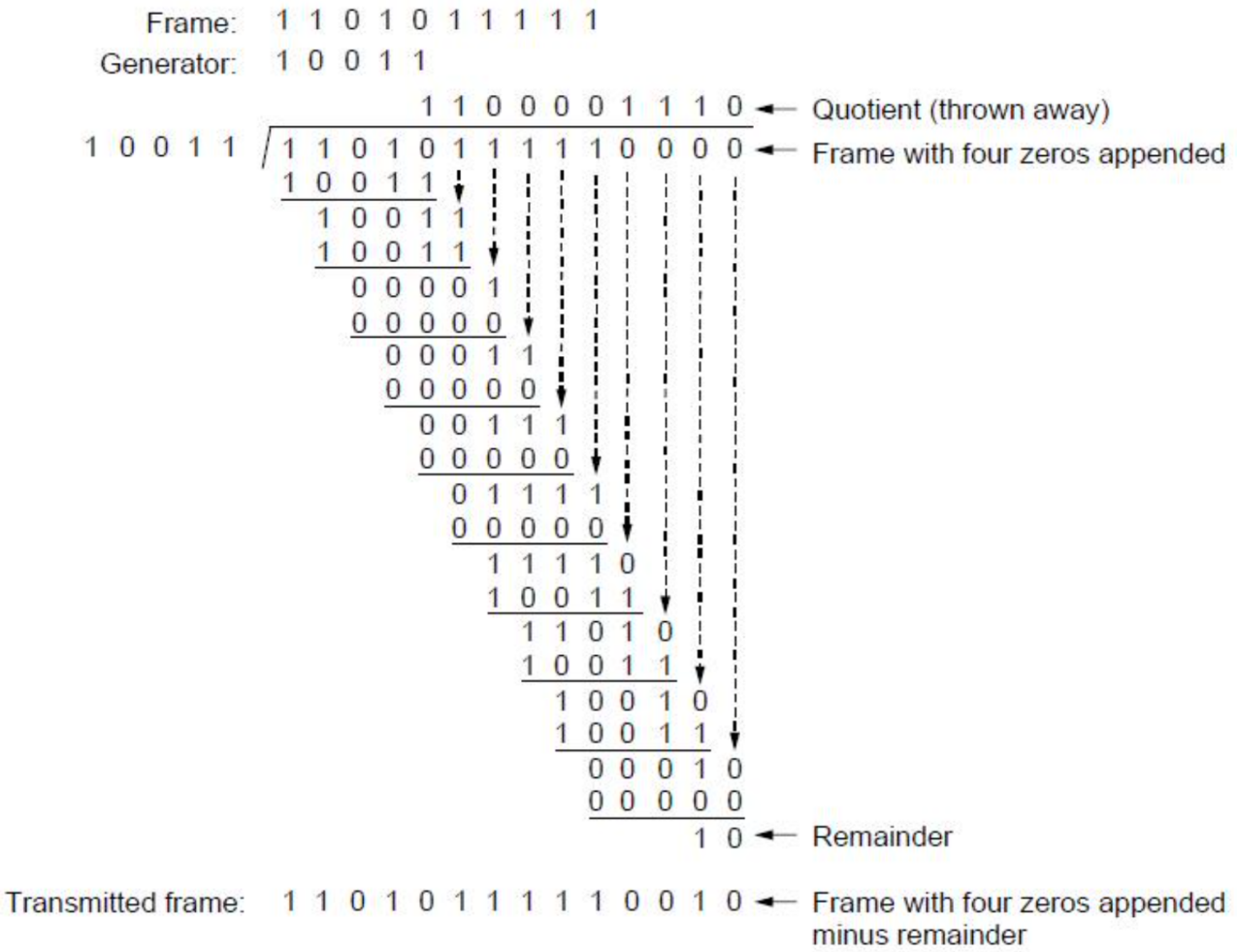
- Given a m-bit message, generate an $m + r$-bit frame such that $m + r$ bit frame evenly divisible by some predefined number

Message Bits (m bits)	Parity Bits (r bits)

- How does it work?
  1. Let $M(x)$ be the message 1101011111
     $M(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$
  2. Choose $G(x)$ with degree r $G(x) = x^4 + x + 1 = 10011 = 19$
  3. Append r zero bits to end of the frame $\Rightarrow x^r M(x)$
     $x^r M(x) = X^{13} + x^{12} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 = 11,0101,1111,0000$
  4. Let $E(x)$ be the reminder of $x^r M(x)/G(x)$ $\quad E(x) = x = 10$
  5. Transmit $T(x) = x^r M(x) - E(x)$
     $T(x) = x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^6 x^5 + x^4 + x = 11,0101,1111,0010$
  6. Recipient divides $T(x)$ by $G(x)$; with remainder $E(x)$
     if $E(x) = 0 \Rightarrow$ no Error
     if $E(x) \neq 0 \Rightarrow$ Error

Just Detection, no Correction

Cyclic Redundancy Check (CRC) II



⇒ All done in Binary, Hardware, with Shift Registers + Adder

Cyclic Redundancy Check (CRC) III

- CRC codes can detect the following
  - All error bursts of length $m - r$ or less
  - All errors with an odd number of errors if generator $G(x)$ has an even number of nonzero coefficient
- Common CRC polynomials

CRC	Generator $g(x)$
CRC-8	$x^8 + x^2 + x^1 + 1$
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11}$ $x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Outline I

Error Correction

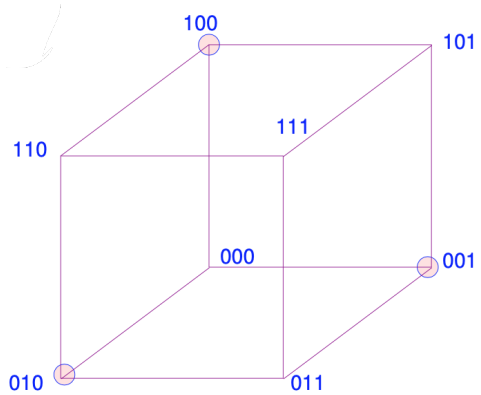
Major Techniques

1. Hamming codes
2. Binary convolutional codes
3. Read-Solomon codes
4. Low-Density Parity Check codes

Hamming Code I

Definition 3.1 (Hamming Distance)
Given two codewords x and y, $d(x, y)$ = number of places that x and y differ = # of 1's in $x \oplus y$

Example 3.2



$d(100, 101) = 1 \quad d(100, 111) = 2 \quad d(100, 011) = 3$
$d(101, 010) = 101 \oplus 010 = 111 \equiv 3$

$d \Rightarrow$ distance in number of hops
$d \Rightarrow$ shows how far codewords are part

Hamming Code II

- We like codewords that are far apart $\Rightarrow$ how many can we have?
- The higher the distance, the better we can detect errors

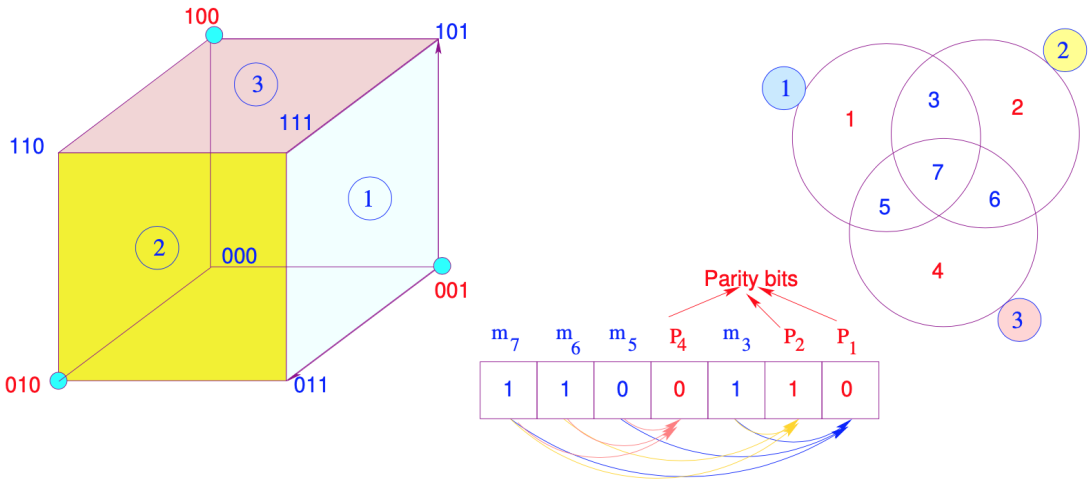
Theorem 3.1 (Minimum Distance)

To detect $l-$bit errors, $d_min \geq l + 1$

Hamming Code III

Example 3.3 (Hamming Code, $H(7,4)$ $n = 7$, $r = 3$, $m = 4$)

- 7-bit codeword, 3-bit error control

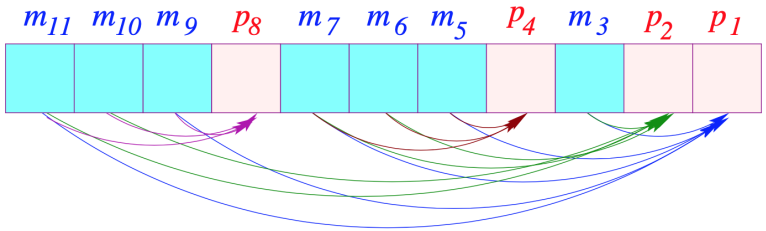


- Counting # of 1's in each circle (dimension)
  - $p_1 = m_3 \oplus m_5 \oplus m_7 = 2 \mod 2 = 0 \Rightarrow$ positions with 1st bit $= 1$
  - $p_2 = m_3 \oplus m_6 \oplus m_7 = 3 \mod 2 = 1 \Rightarrow$ positions with 2nd bit $= 1$
  - $p_4 = m_5 \oplus m_6 \oplus m_7 = 2 \mod 2 = 0 \Rightarrow$ positions with 3rd bit $= 1$

Hamming Code IV

General Algorithm of Hamming Code

1. Write the bit positions in binary; 001, 010, 011, etc.
2. Parity bits are power of 2 positions; 1, 2, 4, 8, etc.
3. All other bits are data bits
4. Each data bit is included in a unique set of parity bits
   1. Parity bit 1 covers bits positions with 1st bit = 1, (1, 3, 5, 7, 9, ...)
   2. Parity bit 2 covers bits positions with 2nd bit = 1, (2, 3, 6, 7, 10 ...)
   3. Parity bit 4 covers bits positions with 3rd bit = 1, (4-7, 12-15, 20-23, ...)
   4. Parity bit 8 covers bits positions with 4th bit = 1, (8-15, 24-31, 40-47, ...)
   5. In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero
5. Set a parity bit to 0 if the total number of 1's in the positions it checks is even
   - Otherwise, 1



Hamming Code V

Example 3.4 (Hamming Code $H(7,4) n = 7,\ r = 3,\ m = 4$

- Let the codeword be $m_7\ \ m_6\ \ m_5\ \ p_4\ \ m_3\ \ p_2\ \ p_1$
- Parity check equations:

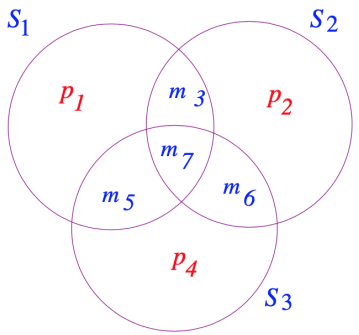
  $S_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 = 0$

  $S_2 = m_3 \oplus p_2 \oplus m_6 \oplus m_7 = 0$

  $S_3 = p_4 \oplus m_5 \oplus m_6 \oplus m_7 = 0$

  There is an even parity in each circle S_1, S_2, S_3 is called the syndrome

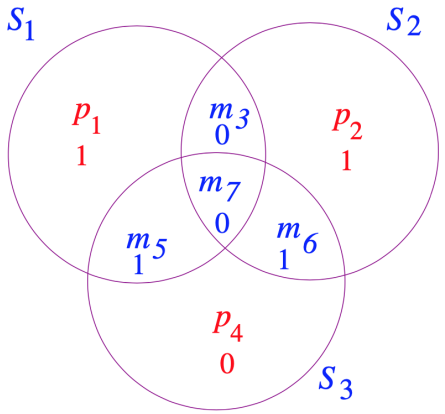
- There is an even number of 1's in each circle $\Rightarrow$ even parity
- Note:
  - if $A \oplus B = C$, then $A = B \oplus C$



Hamming Code VI

Example 3.5 (Hamming Code: Encoding)

- Let the message digits be $m_7 \;\; m_6 \;\; m_5 \;\; m_3 = 0 \;\; 1 \;\; 1 \;\; 0$



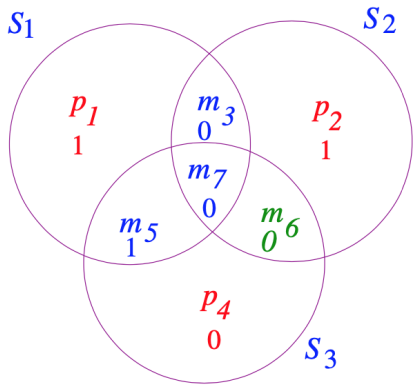
$$p_1 = m_3 \oplus m_5 \oplus m_7 = 1$$
$$p_2 = m_3 \oplus m_6 \oplus m_7 = 1$$
$$p_4 = m_5 \oplus m_6 \oplus m_7 = 0$$

$$\text{The codeword} = m_7 \;\; m_6 \;\; m_5 \;\; p_4 \;\; m_3 \;\; p_2 \;\; p_1 = 0110011$$

Hamming Code VII

Example 3.6 (Hamming Code: Decoding)

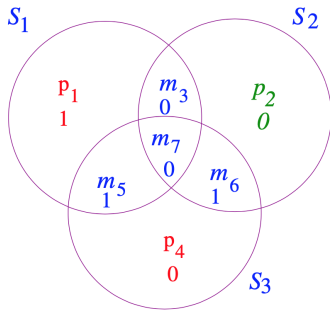
- Transmitted message: $m_7 m_6 m_5 p_4 m_3 p_2 p_1 = 0110011$
- Received message with one bit error: $m_7 m_6 m_5 p_4 m_3 p_2 p_1 = 0\hat{0}10011$
- By counting # of 1's in each circle, we find
  - There is no error in circle 1 $\Rightarrow S_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 = 0$
  - There is an error in circle 2 $\Rightarrow S_2 = m_3 \oplus p_2 \oplus m_6 \oplus m_7 = 1$
  - There is an error in circle 3 $\Rightarrow S_3 = p_4 \oplus m_5 \oplus m_6 \oplus m_7 = 1$
  - Therefor, the error is in the intersection of circle S_2 and S_3
    - $S_3 S_2 S_1 = 110 \Rightarrow m_6 = 1$ must be reverted to 1
- Note: if the bit error affects
  - two circle $\Rightarrow$ data bit
  - one circle $\Rightarrow$ parity bit



Hamming Code VIII

Example 3.7 (Hamming Code: Decoding)

- This time a parity bit is in error
- Transmitted message:
  $m_7 m_6 m_5 p_4 m_3 p_2 p_1 = 0110011$
- Received message with one bit error:
  $m_7 m_6 m_5 p_4 m_3 p_2 p_1 = 0110\hat{0}1$
- By counting 1's in each circle we find
  - There is no error in circle 1 $\Rightarrow S_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 = 0$
  - There is an error in circle 2 $\Rightarrow S_2 = m_3 \oplus \hat{p_2} \oplus m_6 \oplus m_7 = 1$
  - There is no error in circle 3 $\Rightarrow S_3 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 = 0$
  - Therefore the error is S_2
    - $S_3 S_2 S_1 = 010 \Rightarrow$second bit $\hat{p_2}$ must be reverted



Hamming Code IX

- Similarly, $H(11,7)$, $n = 11$, $r = 4$, $m = 7$
  - $p_1 = m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11}$ $\Rightarrow$ 1st bit = 1
  - $p_2 = m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11}$ $\Rightarrow$ 2nd bit = 1
  - $p_4 = m_5 \oplus m_6 \oplus m_7$ $\Rightarrow$ 3rd bit = 1
  - $p_8 = m_9 \oplus m_{10} \oplus m_{11}$ $\Rightarrow$ 4th bit = 1
- In general
  - Block length: $2^r - 1$
  - Message length: $2^r - r - 1$
  - Distance: 3
  - Notation: $H(m, m - r)$
  - Rate (efficiency): $1 - \frac{r}{2^r - 1}$

The Magic

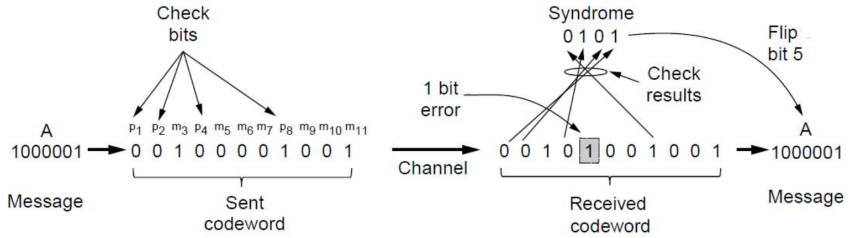
Any two different code words differ in at least 3 places in H(11,7)

1100011	1101000	1110100	1111111	1000101	1001110	1010010	1011001
0100110	0101101	0110001	0111010	0000000	0001011	0010111	0011100

- Why these specific bits? $\Rightarrow$ maximize code distance
- The knowledge of Hamming distance is used to:
  - Determine the capacity of a code $\Rightarrow$ how many bits for data and parity
  - Detect and correct errors

Hamming Code X

Example 3.8 (Hamming Code, H(11,7) $n = 11,\ r = 4,\ m = 7$



- m_5 is the only bit that can cause the error
  - $S_1 = p_1 \oplus m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} = 1$
  - $S_2 = p_2 \oplus m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} = 0$
  - $S_3 = p_4 \oplus m_5 \oplus m_6 \oplus m_7 = 1$
  - $S_4 = p_8 \oplus m_9 \oplus m_{10} \oplus m_{11} = 0$
- $S_4S_3S_2S_1 = 0101 \Rightarrow$ 5th bit, m_5 must be reverted

Hamming Code XI

Theorem 3.2

The number of errors that can be corrected is

$$\frac{d_{min}-1}{2}$$

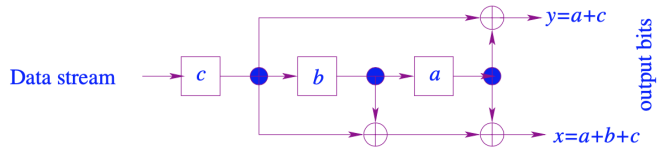
Outline I

Convolutional Coding I

- Generates redundant bits continuously
- More powerful than block codes
  - NASA Voyager 1977 and 802.11
  - Provides good performance at low cost

k bits $\longrightarrow$ (n, k, L) $\longrightarrow$ n bits, L lines

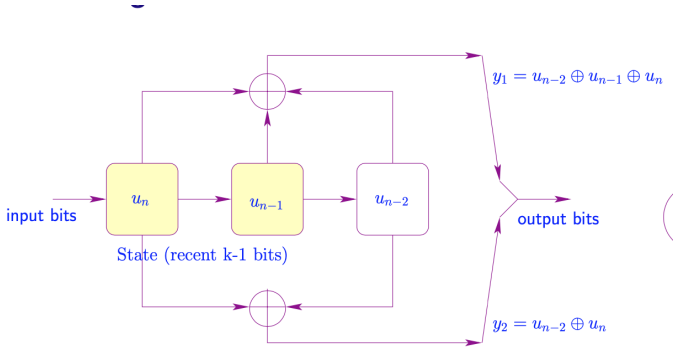
- (2,1,3) convolutional code

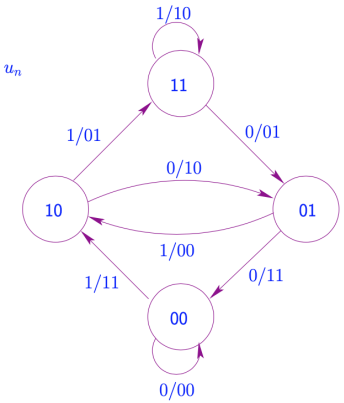


- Encodes information stream rather than information blocks
- Certain information bits affect the encoding on next information bits
- Easily implemented using shift register
  - input processes k bits at a time
  - Output produces n bits for every k input bits
  - k and n are very small
  - L (constraints) factor is the number of shifts over which an input bit can influence the encoder output
  - In general, the code rate $r = k/n$, 1/2 in this example

Convolutional Coding II

- Encoding



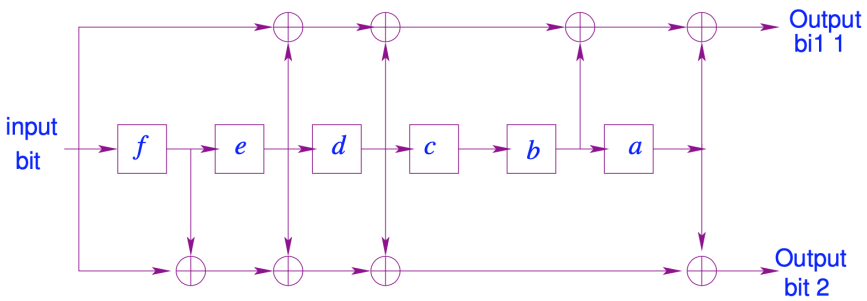


Input	Initial State	Final State	Output Codeword
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	10
1	10	11	01
0	11	01	01
1	11	11	10

- Decoding
  - Uses maximum likelihood for decoding (Tree of Trellis)
  - Viterbi algorithm is the most common

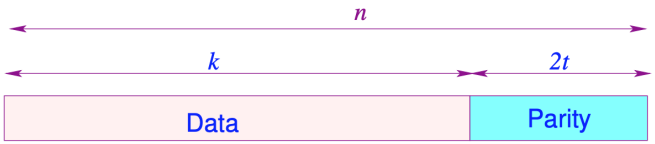
Convolutional Coding III

- (2,1,7) convolutional code



Reed Solomon (RS) Codes I

- Block-based error correcting codes with a wide range of applications
  - Storage devices (including tape, Compact Disk, DVD, Blu-ray, DSL, WiMax, RAID, bar-codes, etc.)
  - Wireless or mobile communications
    - Cellular telephones, microwave link, etc.
  - Satellite communications
  - Digital television, CD, DVD, RF, barcodes
  - High-speed modems such as ADSL, xDSL, etc.



- A popular Reed-Solomon code is RS(255,223) with 255 code word bytes, of which 223 bytes are data and 32 byte parity
  - $n = 2^r - 1 = 255 \Rightarrow s = 8$
  - $k = 223$
  - $2t = 32, t = 16$
- The decoder can correct any 16 symbol errors in the code word

Reed Solomon (RS) Codes II

- The decoder can correct any 16 symbol errors in the code word
  - Errors in up to 16 bytes anywhere in the codeword can be automatically corrected $\Rightarrow$ very strong

How Does it Work?

- RS codeword is generated using a special polynomial (generator)

$$g(x) = (x - a^i)(x - a^{i+1})(x - a^{i+2})...(x - a^{i+2d}) \quad (1)$$

- The codeword $c(x)$ is connected using $g(x)$ and the information block $i(x) \Rightarrow$ shift the bits to the left

$$c(x) = g(x).i(x) \quad (2)$$

Reed Solomon (RS) Codes III

- The $2t$ parity symbols in a systematic Reed-Solomon codeword are given by

$$p(x) = i(x) \cdot x^{n-k} \mod g(x) \tag{3}$$

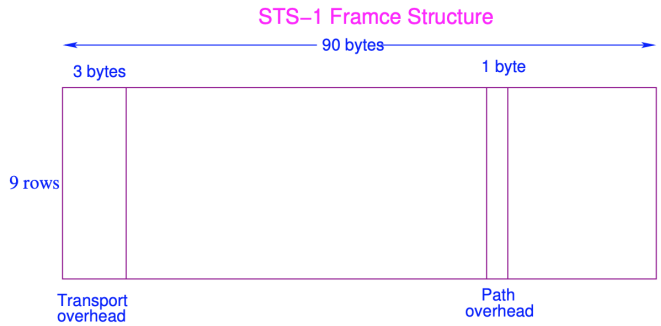
- The received codeword $r(x)$ is the original (transmitted) codeword $c(x)$ plus errors

$$r(x) = q(x) \cdot g(x) + e(x) \qquad q(x)g(x) = c(x) \tag{4}$$

- Finding the position of involves solving simultaneous equations with t unknowns
  - Several fast algorithms are available

Clock-Based Framing:SONET I

- SONET: Synchronous Optical Network
- ITU standard for transmission over fiber
- STS-1 (51.84 Mbps)
- Byte-interleaved multiplexing
- Each frame is $125\mu s$ long.



- Each frame is $125\mu s$ long $\Rightarrow$ 8000/s
- Frame length$= 90 \times 9 \times 8 = 6,400$ bits long

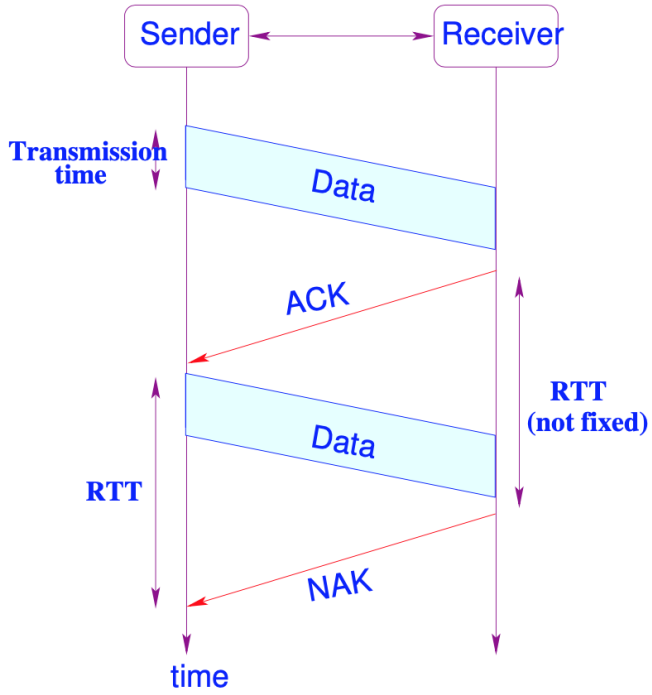
Outline I

Reliable Transmission

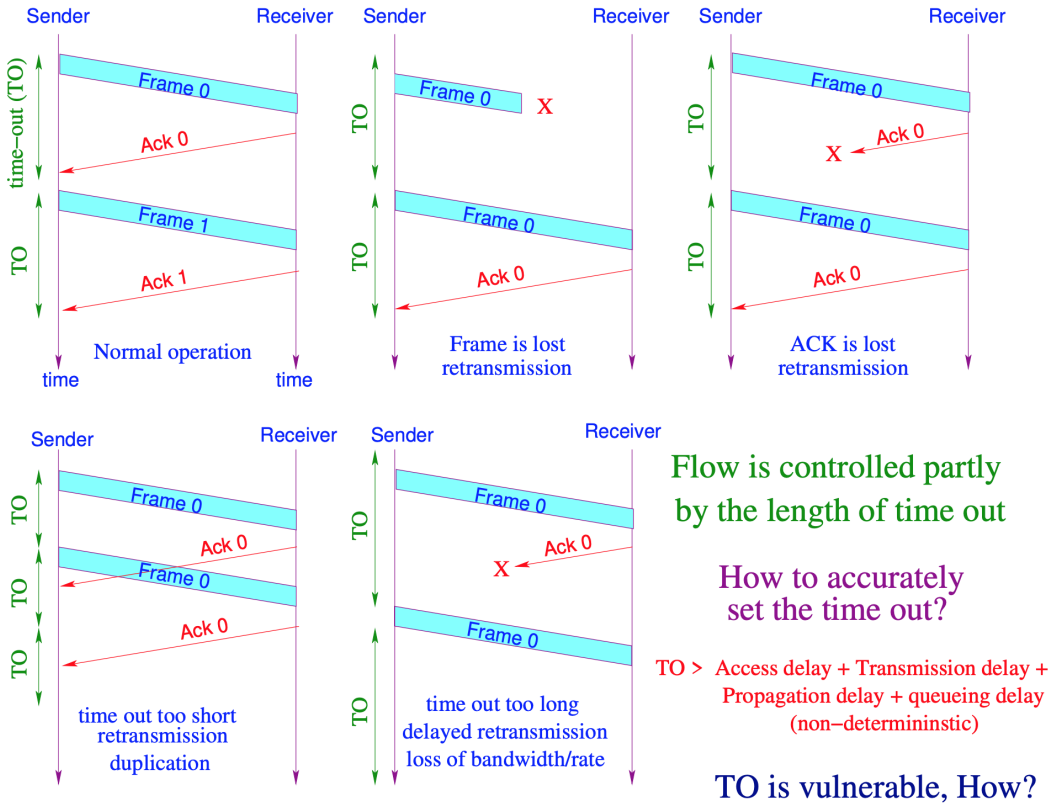
- Delivers frames
  - Without errors and
  - In proper order to the network layer
- How to recover from corrupt frames (errors)?
  1. Error Correction $\Rightarrow$ also called Forward Error Correction (FEC)
  2. Acknowledgments and Timeouts $\Rightarrow$ also called Automatic Repeat Request (ARQ)
- How to keep the proper oder of delivery?
  - The sender should not flood the receiver
  - But maximizes the throughput
- The sender is throttled until the receiver grants permission
  Flow Control $\Rightarrow$ ARQ

Automatic Repeat Request (ARQ)

- Performed with the combination of:
  1. Error Detection
  2. Acknowledgment
  3. Retransmission after timeout
  4. Negative acknowledgment



What Could Go Wrong

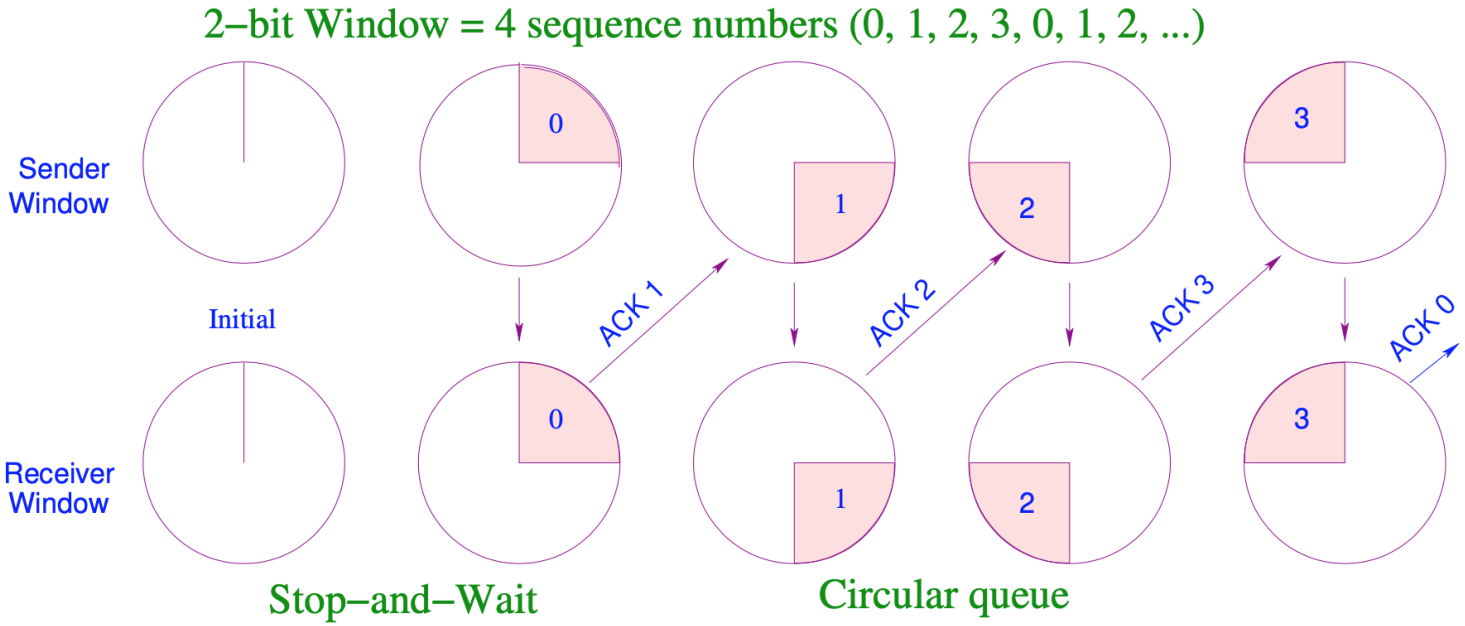


Frames in Error

- Possible errors
  - Damaged frames
    - Frames received with error
    - Frames lost
    - Last frame lost
  - Damaged ACL
    - One ACK lost, next one makes it
    - All ACK lost
- How to Handle frames in errors?
  1. Stop-and-Wait Protocol
  2. Go-Back-N Protocol
  3. Selective Reject Protocol

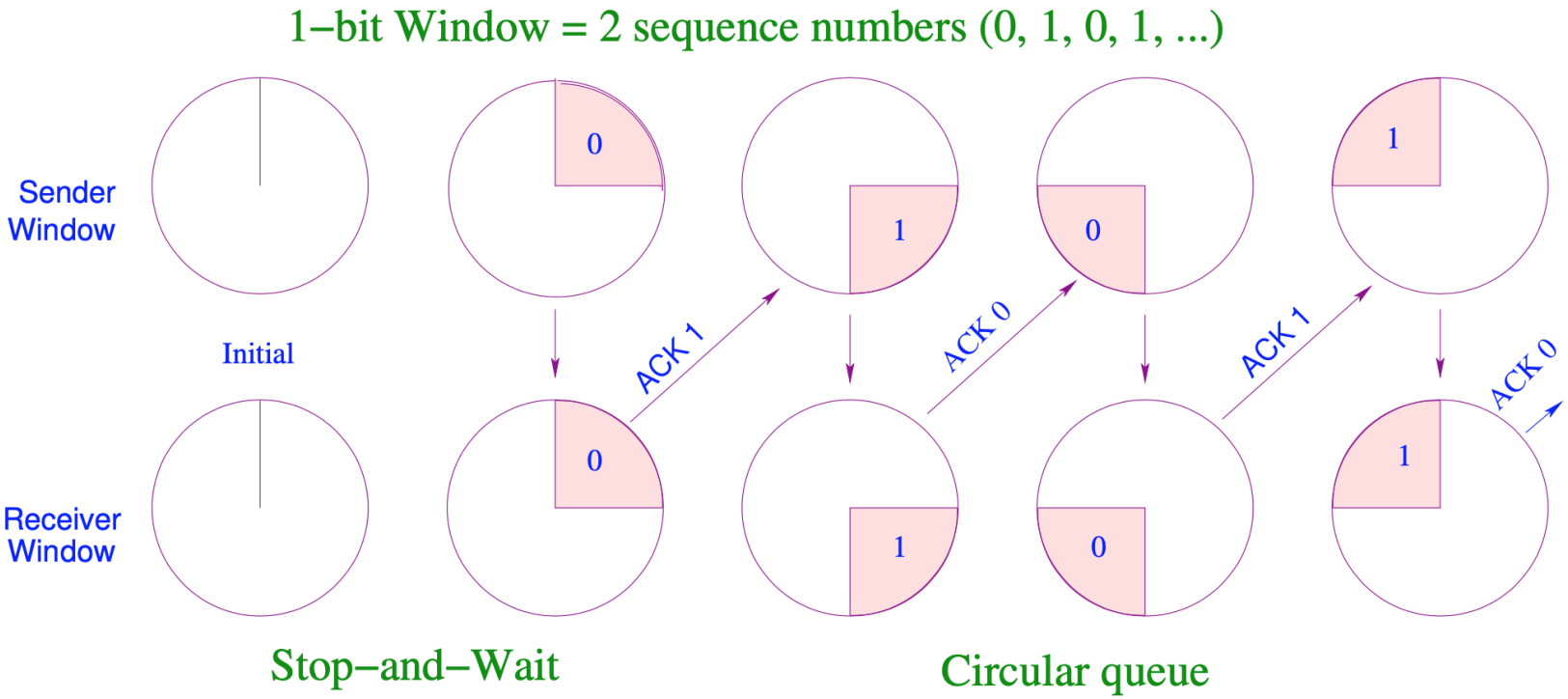
Stop-and-Wait Protocol I

- The sender/receiver each has two windows:
  - Sending windows
  - Receiving windows



- It is a duplex mechanism
  - One can send/receive at the same time
    - Piggybacking
- Save sequence numbers $\Rightarrow$ reuse once it is released

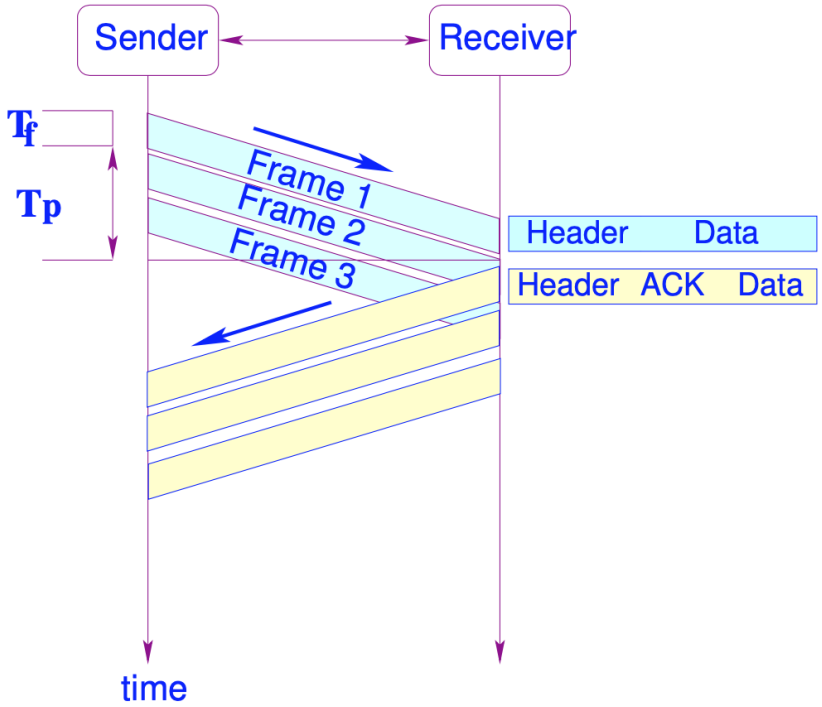
Stop-and-Wait Protocol II



- we always ACK the frame to be received
  - Receiver knows which frame has been received
  - Receiver knows which frame the source is waiting for

Stop-and-Wait Protocol III

Piggybacking



How good Stop-and-Wait is?

Example 3.9 (Utilization (or Efficiency) Problem)

- Assume 1.5 Mbps link and 45 ms RTT $\Rightarrow$ Pipe = 67.5 Kb (8KB)
- Frame size: 1KB
- SW uses about 1/8 of the bandwidth
- How do we fix this
  - Send up to 8 frames before having to wait for an ACK

Performance of SW Protocol I

- T_f: frame time and T_p: propagation delay

$$\alpha = \frac{T_p}{T_f} = \frac{\frac{Distance}{Speedofsignal}}{\frac{Framesize}{Bitrate}} = \frac{Distance \times BitRate}{Framesize \times Speedofsignal} \tag{5}$$

$$\text{SW Utilization} = E = \frac{1}{1+2\alpha} \tag{6}$$

Performance of SW Protocol II

Example 3.10

Long propagation delay

- Satellite link $\Rightarrow T_p = 278$ ms
- Suppose the frame size is 4k bits $= 4096$ bits
- Suppose data rate is DS0, 56 Kbps $\Rightarrow T_f = 4096/56000 = 0.073$ sec.

$$\alpha = \frac{T_p}{T_f} = \frac{0.278}{0.073} = 3.8$$

3.8 frames fits into the pipe on one direction

$$U = \frac{1}{1+2\alpha} = \frac{1}{1+7.6} = 0.12 \qquad U = 12\%$$

Performance of SW Protocol II

Example 3.11

Short propagation delay $\Rightarrow$ Ethernet

- $T_p = 10\mu s = 10^{-5}$ sec
- Frame size = 4k bits = 4096 bits
- Data rate = 10Mbps $\Rightarrow$ $T_f = 4096/10^7 = 0.0004096$ sec.

$$\alpha = \frac{T_p}{T_f} = \frac{10^{-5}}{0.0004096} = 0.0244$$

3.8 frames fits into the pipe on one direction

$$U = \frac{1}{1+2\alpha} = \frac{1}{1+0.0488} = 0.95 \qquad U = 95\%$$

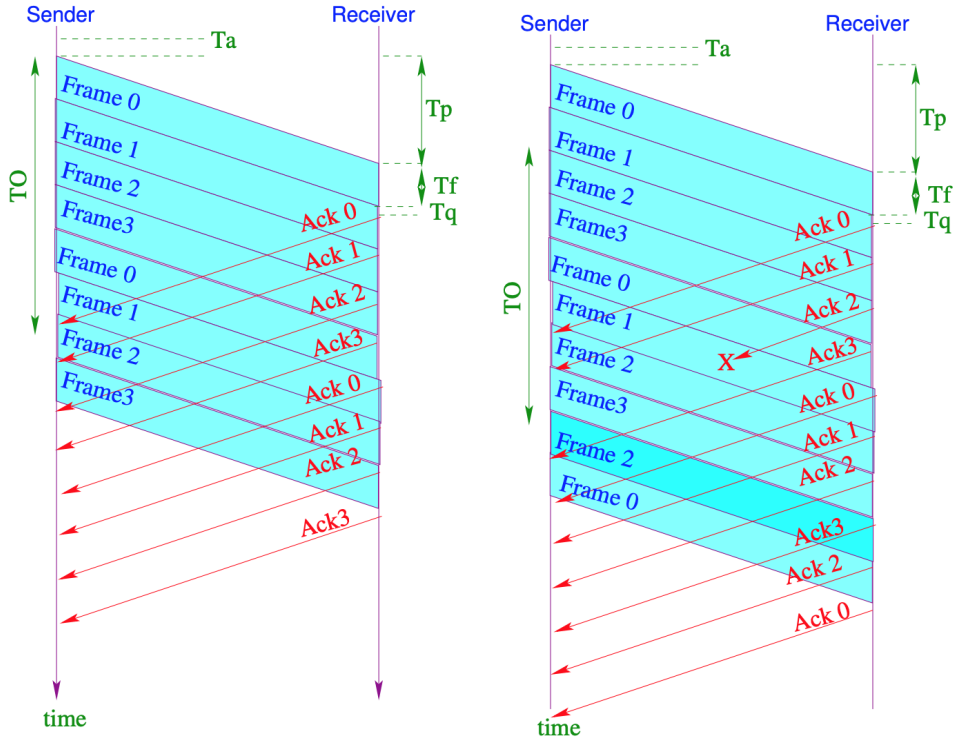
How to solve the Efficiency Problem

Keep the pipe/channel full to get better efficiency

Send more than one frame at a time

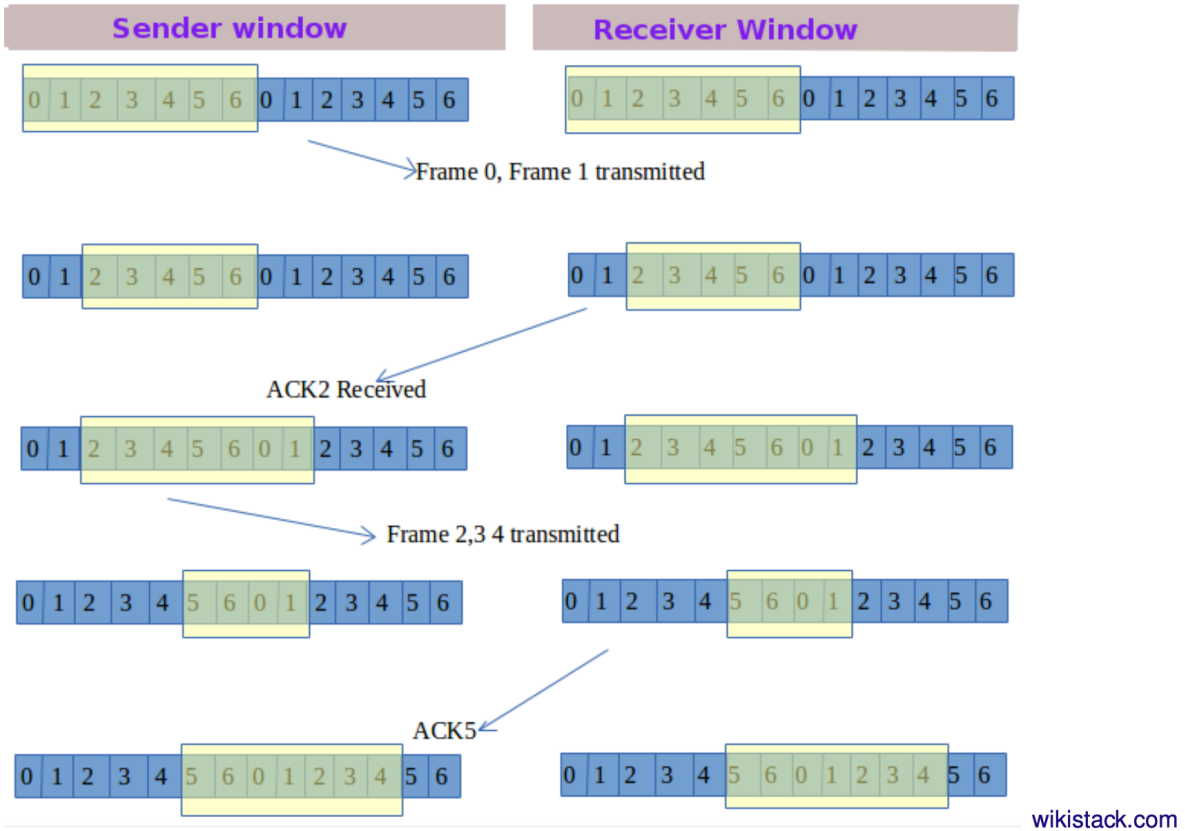
Sliding Window I

- Allow sender to transmit multiple frames before receiving an
  - keeping the pipe full



  - The receiver has two copies of frame 2
  - The 2nd incarnation discarded if both both have the same sequence numbers

Sliding Window II

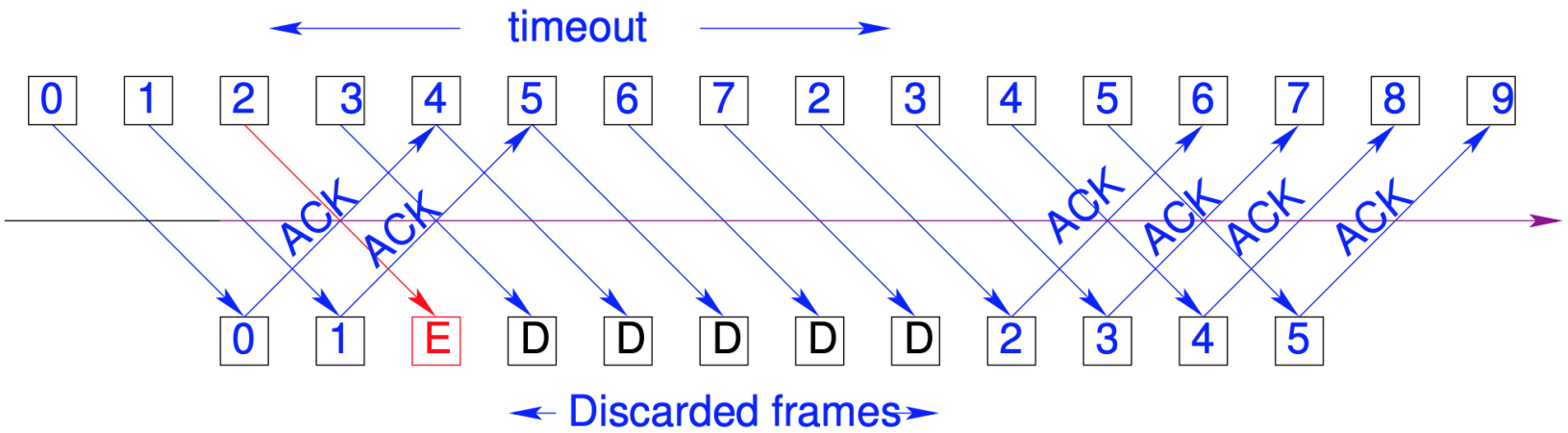


- Window size = 7
  - Potentially can send 7 frames if it does not overwhelm the receive

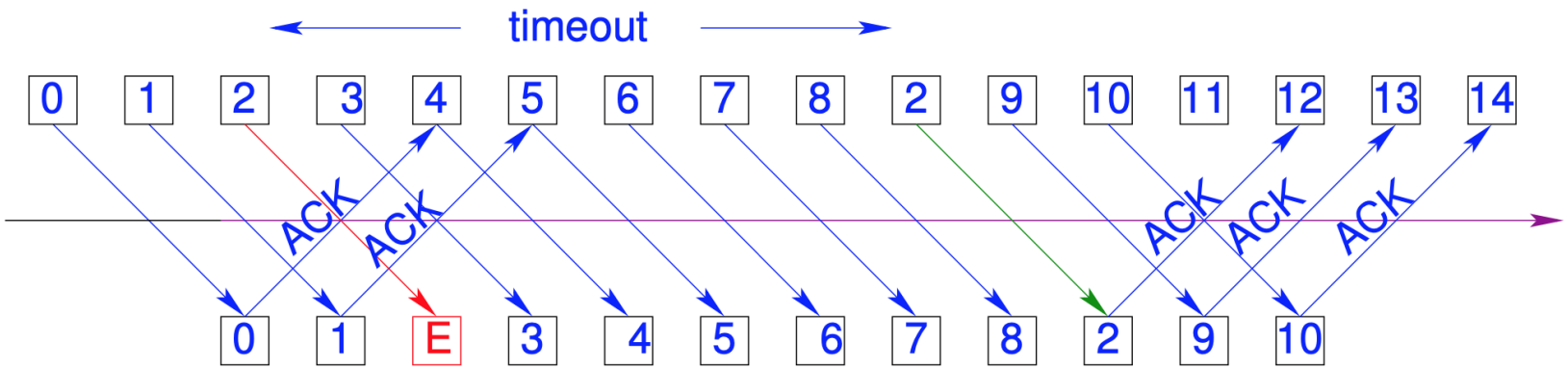
Sliding Window III

- Note that speed of light in:
  - vacuum is $300 \times 106 m/s$
  - fiber is $200 \times 106 m/s$
- The sender should not send frames at a higher rate than what the receiver can receive
  - Transmission rate = *min* sender rate, receiver rate
  - This is called Flow Control
- How?
  - Send only when an ACK arrives
  - Accurately calculate Time Out (TO)
- What could go wrong with sliding window?
  - Multiple frames can be lost
- How to handle that?
  - Use Go-Back-N Protocol or Selective-Reject Protocol
  - Selective-Reject Protocol performs well with devices with large buffers

Go-Back-N Protocol



Selective-Reject Protocol



Sequence Number Space

- *SeqNum* field is finite $\Rightarrow$ wraps around
- *SeqNum* space $>$ # of outstanding frames
- $SWS \leq MaxSeqNum - 1$ is not sufficient

Example 3.12 (suppose 3-bit SeqNum field (0..7))

- $SWS = RWS = 7$
- Sender transmit frames *0 ... 6*
- Arrive successfully, but ACKs lost
- Sender retransmits *0 ... 6*
- Receiver expecting *7,0..5* , but receives second incarnation of *0..5*
- *SWS ¡ (MaxSeqNum+1)/2* is correct rule
  - Intuitively, *SeqNum* "frames" between two halves of sequence number space
- What does it mean? $\Rightarrow$ Don't send more than half of your sequence number

Outline I

References

[Tanenbaum and Wetherall, 2011] Tanenbaum, A. S. and Wetherall, D. J. (2011). Computer Networks: 5th Edition. Prentice Hall PTR.