King Fahd University of Petroleum & Minerals Computer Engineering Dept

COE 540 – Computer Networks

Term 141

Dr. Ashraf S. Hasan Mahmoud

Rm 22-420

Ext. 1724

Email: ashraf@kfupm.edu.sa

Lecture Contents

1. Data link layer design issues
2. Error detection and correction
3. Data link protocols and performance
4. Example data link protocols

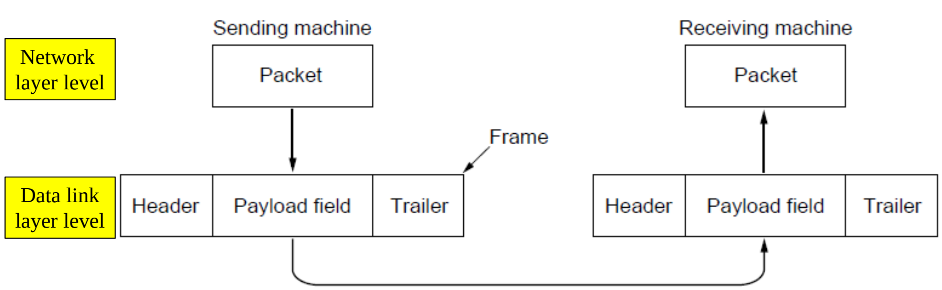
These slides are based on the Tanenbaum's textbook and original author slide

Data Link Design Issues

- Services provided to the Network layer
- Framing
- Error control
- Flow control

Data Link Services

- The main services include:
  - Providing well-defined service interface to the network layer
  - Dealing with transmission errors
  - Flow control
- Unit of transmission – FRAME



Services Provided to the Network Layer

- Three possibilities:
  1. Unacknowledged connectionless service
  2. Acknowledged connectionless service
  3. Acknowledged connection-oriented service

- Unacknowledged connectionless service:
  - No logical connection established/released
  - No attempt to detect or correct errors
  - Appropriate for real-time traffic
  - E.g. Ethernet
- Acknowledged connectionless service:
  - No logical connection
  - Each frame is sent individually and is acknowledged
  - E.g. WiFi
- Acknowledged connection-oriented service:
  - Source and destination establish a connection before data is transferred – 3 connection phases
  - Each sent frame is numbered and ack-ed
  - DL provides (to the network layer) a reliable bit stream

Services Provided to the Network Layer – cont'd

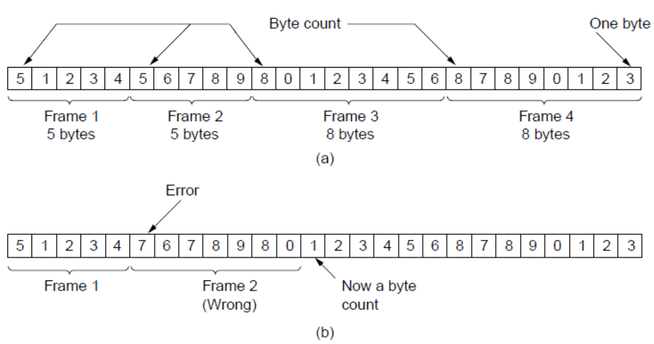
- Questions to contemplate about:
  1. Is the ACK function a requirement for the DL layer? Why or why not?
  2. Why not move the ACK function to the upper Network layer?

Framing

- Continuous bit stream (of the physical layer) is broken up to discrete frames
- Some overhead is added to the frame
  - Checksum/CRC – error detection;
  - Frame sequencing;
  - Etc.
- How to determine the frame boundary - Methods for framing:
  1. Bytes count
  2. Flag byte with byte stuffing
  3. Flag bit with bit stuffing
  4. Physical layer coding violation

Framing – Bytes Count

- A field in the header that determines the frame length
- What happens if the field is corrupted by noise?



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

Framing – Bytes Count - More

- Length of the length field should be at least $\lfloor \log_2 K_{max} \rfloor + 1$
  - Kmax is the maximum frame length
  - Similar to the overhead for bit-oriented framing
- Could any other method of encoding frame lengths require smaller expected number of bits?
  - Information theory
  - Given any probability assignment P(K) on frame lengths, then the minimum expected number of bits that can encode such lengths is at least the entropy of that distribution, given by

    $H = \Sigma P(K) \log_2 P(K)^{-1}$
  - Example – let P(K) = 1/Kmax (i.e. all lengths are equally probable) ➔ $H = \log_2$ Kmax
  - Example – let $P(K) = p(1-p)^{K-1}$ where $p = 1/E\{K\}$ ➔ $H = \log_2 E\{K\} + \log_2 e$ for large E{K}

Source Coding for Frame Lengths

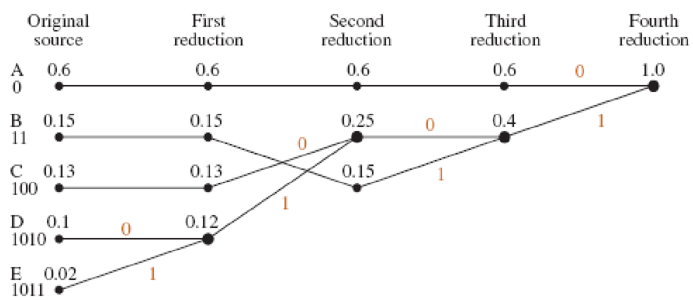
- The idea is to
  - map more likely values of K into short bit strings
  - Map less likely values of K into longer bit strings
- That is: map a given K into $\log_2 P(K)^{-1}$ bits
- For geometric distribution of K ➔
  - Maximum # of required bits
  - The resulting code is called: Unary-binary encoding

Huffman Source Coding

- **Example:** Consider the symbols (or frame lengths) occurring with the probabilities shown in figure. Find the coding that results in minimum average number of bits per symbol (or frame length)

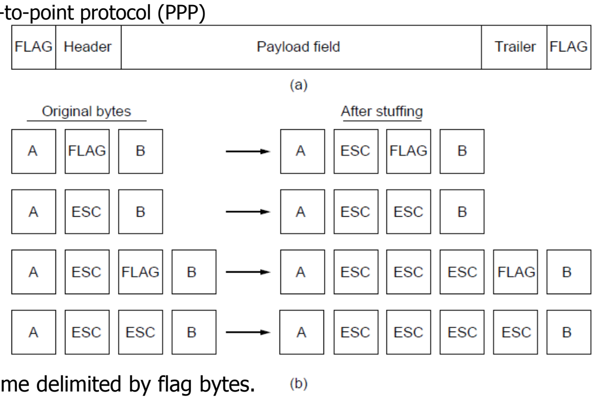
- **Solution:** See figure (A → 0, B → 11, C → 100, D → 1010, E → 1011)
  Avg # of bits, L_h = 1x0.6 + 2x0.15 + 3x0.13 + 4x0.1 + 4x0.02 = 1.77 bits
  Entropy, $H = \Sigma P(K) \log_2 P(K)^{-1} = 1.68$

  In general $H \leq L_h \leq H + 1$



Framing – Flag Byte

- Special bytes used to identify the boundaries of the frame
- Use of byte stuffing
  - What if the Flag byte appears in the payload?
  - What if the ESC byte itself appears in the payload?
- E.g. the point-to-point protocol (PPP)



(a) A frame delimited by flag bytes.
(b) Four examples of byte sequences before and after byte stuffing.

Framing – Flag Bits (Unique Pattern)

- Unique bit pattern
  - E.g. 01111110 or 0x7E – used by HDLC
- Use of bit stuffing
- Bit stuffing rules at the transmitter/receiver

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

Stuffed bits

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

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

Framing – Physical Layer Violations

- Example: 4B/5B encoding – only 16 valid patterns out of total of 32
- Use some reserved signals (code violations) to indicated the start and end of frame
  - No need to stuff the data!

Framing – General Comments

- Some data link protocols use a combination of the previous methods
  - 802.11 uses a 72-bit long preample + a length (count) field
- Compare the efficiency of flag byte based framing and the unique bit pattern framing? (For the worst case)

Error Control

- Essential function for reliable services
- Types of errors:
  - Erroneous frame
  - Lost frame
- Use of +ve/-ve ACKs
- Use of timers
  - May get a frame transmitted multiple times if the corresponding ACK is lost
- Need to manage sequence numbers and timers
- Error control = error detection + error correction

Flow Control

- Transmitter should not overwhelm receiver with data
- Feedback-based control flow
  - Well-defined rules about when a sender may transmit the next frame; receiver may grant permits allowing sending of frames
  - E.g. ARQ
- Rate-based flow control
  - TCP

Error Detection and Correction

- Added redundancy in the original data stream enables error detection and correction
- Error-detecting codes
- Error-correcting codes
- Error models:
  - Random - isolated erroneous bits; due to thermal noise
  - Burst – e.g. deep fades in wireless comm; or due to impulse noise
- Erasure channel – bit lost due invalid signal level
  - Easier to handle relative to channels that flip bits

Error-Correcting Codes

- Hamming Codes
- Binary convolutional codes
- Reed-Solomon codes
- Low-density parity check codes

Error-Correcting Codes - definitions

- m data bits + r redundant bits
- **Block code** – the r bits are computed based on m – i.e. data handled in blocks
  - Total length of block, n = m + r
- **Systematic code** – what is sent is the m data bits plus the r check bits
- **Linear code** – the r check bits are computed as a linear function of the m data bits
- Block of n = m + r bits ➔ codeword
- **Code rate** is m/n = m/(m+r)

Hamming Distance and Code Strength

- **Hamming distance**: number of bit positions in which two codewords differ

- There are 2^m possible data messages ➔ 2^m valid codewords

- There are 2^n possible patterns of n bit long
- Fraction of used patterns: 2^m/2^n = 2^(-r)

Error Detection/Correction Capability

- To detect d errors – code with distance d+1

- To correct d errors – code with distance 2d+1
  - Assumes a large number of errors is less likely than a low number of errors

Example: Consider the following code:
Valid code words:
0000000000, 0000011111, 1111100000, and 1111111111

- Code distance is 5 – why?
- Therefore, can correct up to (5-1)/2 = 2 bit errors (i.e. single and double bit errors)
- If 0000000111 is received ➔ Receiver will assume that 0000011111 is transmitted
- If 0000000000 is transmitted and 3 bit errors occur such that 0000000111 is received, then the receiver cannot correct properly.
- Can you detect double bit errors and detect quadruble bit errors at the same time?

Number of Check Bits (r)

- Consider a code: m data bits – r check bits that corrects SINGLE bit errors
  - $n = m + r \rightarrow 2^n$ total number of n bit patterns
- There are 2^m legal messages – For each there are n illegal codewords at distance 1 from it
  - Each of the 2^m messages require n+1 bit patterns
- Therefore, we must have

$$(n+1)2^m <= 2^n \text{ or } (m+r+1) <= 2^r$$

- Given m, this puts a lower limit on the number of check bits r that is needed to correct single bit errors

Example: m = 7, then r must be 4, 5, … for the above inequality to be satisfied

Hamming Codes achieves this theoretical lower limit on r

- i.e. There exists Hamming code with m = 11, r = 4 that can correct all single bit errors
- Code denoted by (n, m) = (11, 7)

How to Construct Hamming Codes - 1

- Bit positions are numbered from left to right – starting with bit 1, bit 2, etc.
- Check (parity bits) occupy bit positions that are powers of 2; i.e. 1, 2, 4, 8, 16, etc.
- Data bits occupy the rest of bit positions; i.e. 3, 5, 6, 7, 9, etc.

Example of Hamming Code (11, 7)

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
P1	P2	M3	P4	M5	M6	M7	P8	M9	M10	M11

(yellow)	Parity or check bit
(green)	Data bits

How to Construct Hamming Codes - 2

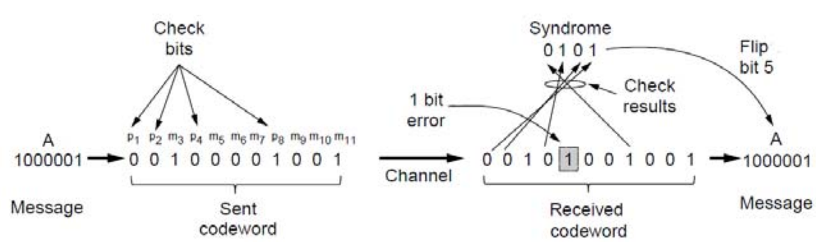
- How to compute parity bits (assuming EVEN parity):
  - Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...) ➔ B1 = B3 + B5 + B7 + B9 + …
  - Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...) ➔ B2 = B3 + B6 + B7 + B10 + B11 + …
  - Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...) ➔ B4 = B5+B6+B7 + B12+B13+B14+B15 + …
  - Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
  - Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
  - Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)
  - etc.
  - Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Hamming Code (11,7) Example

- Refer to the example in textbook page 226
- Computation of parity bits (refer to figure – assumes even parity):

P1 + B3(1)+B5(0)+B7(0)+B9(0)+B11(1) = 0 ➔ P1 = 0;
P2 + B3(1)+B6(0)+B7(0)+B10(0)+B11(1) = 0 ➔ P2 = 0;
P4 + B5(0)+B6(0)+B7(0) = 0 ➔ P4 = 0;
P8 + B9(0) + B10(0)+B11(1) = 0 ➔ P8 = 1;

- The check bits P1P2P4P8 = 0001.



Hamming Code (11,7) Example 2

- Refer to the example at: http://users.cs.fiu.edu/~downeyt/cop3402/hamming.html

Error Syndrome and Correction

- In the previous example, M5 is flipped
- The received codeword is

$$0_1 0_2 1_3 0_4 1_5 0_6 0_7 1_8 0_9 0_{10} 1_{11}$$

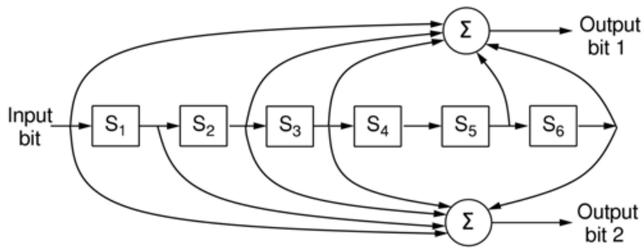
- Parity bits are re-computed at the receiver resulting in C4C3C2C1 = $(0101)_2$ = $(5)_{10}$ which is the index of the erroneous bit!

- Used mostly in error-correcting memory

P1 + B3(1)+B5(1)+B7(0)+B9(0)+B11(1) = 1 = C1
P2 + B3(1)+B6(0)+B7(0)+B10(0)+B11(1) = 0 = C2
P4 + B5(1)+B6(0)+B7(0) = 1 = C3
P8 + B9(0) + B10(0)+B11(1) = 0 = C4

Convolutional Codes

- Not a block code – no natural message size
- See example below:
  - Delay elements S1, S2, …, S6 are used
  - For every clock cycle one input bit is inserted, two output bits are read out
  - Output bits are function of the input bit as also the state of the coders (S1, S2, …, S6)
  - Coding rate, R_c = 1/2, Constraint length, k = 7



Convolutional Codes - 2

- Constraint length – defines the extent of the memory for the coder
- If the input is not included in the output bits, then the code is non-systematic
  - Case for previous example.
- Used widely in Telecomm (e.g. GSM, UMTS, WiMAX, etc.)

Convolutional Codes - Decoding

- Decoding – finding the sequence of input bits that is most likely (fewest number of errors) to have produced the observed sequence of output bit
- Viterbi decoding algorithm – Trellis diagram
- Hard-decision decoding versus soft-decision decoding

Convolutional Codes

- http://oscar.iitb.ac.in/onsiteDocumentsDirectory/674_ConvolutionalCoding/674_ConvolutionalCoding/index.html
- Excellent animation for encoder and decoder

Reed-Solomon Code

- Linear block code
  - Often systematic
- Operate on m-bit symbols
- Basic idea – n degree polynomial is completely specified by n+1 points.
  - E.g.: line ax + 1 is determined by two points – if given 4 (different) points, then any two would suffice to specify the line
  - The extra points are redundant – useful for error correction
- Uses polynomials that operate over finite field
  - For m-bit symbols, the codewords are 2^m-1 symbols long

Example: m = 8 (i.e. a byte) – RS code (255, 223) is widely used
  - Contains 255-223=32 redundant bits

- Excellent introduction on RS codes: http://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html

- Strong error-correcting properties especially for burst errors
- Used for telecom – DSL, satellite communications, etc.
- Used also for CDs, DVDs, and blu-ray discs.

Low-Density Parity Check (LDPC) Code

- Linear block code invented by R. Gallager
- The output bit if a function of a fraction of the input bits – matrix representation of the code has low density of 1s
- Decoding is done by an approximation algorithm which iteratively converges to the best fit of the received data to a legal codeword
- Practical for large block sizes and have excellent error-correction capabilities

- Used recently in digital video broadcasting, 10G Ethernet, IEEE 802.11n (option), etc.

Error Detection Codes

- Parity
- Checksums
- Cyclic redundancy Checks (CRCs)

Parity

- Parity bit is appended to data to form codeword
  - Even parity – number of 1s in the codeword is even
  - Odd – number of 1s in the codeword is odd
- With single parity bit – distance of code = 2
  - Can detect single bit errors
- Excellent of low bit error rate channels

Example: Pe = 10^{-6} (1 in megabit!), m = 1000 (data) bits

FEC - to correct single bit errors we need at least r = 10 bits
For a megabit of data, total overhead = 10,000 check bits

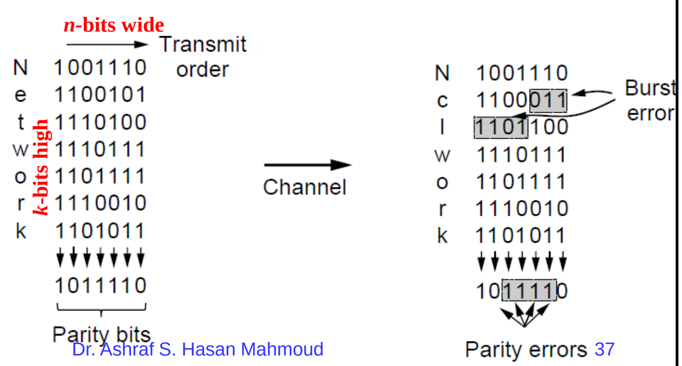
If parity is used to detect the single bit error ➔ 1000 parity bits are needed – With one expected error ➔
total overhead = 1000 + 1001 (retransmitted block with its parity bit) = 2001 bit

- More efficient than FEC

Parity – Working Against Error Bursts

- Form the data block as a matrix of n bits wide and k bits high
- Compute the parity bit for every column
- Send the data row wise (starting from the top and left to right) ➔ Interleaving
- Can handle a burst of up to n bits wide

- A burst of n+1 bits where only first and last bits are inverted will pass undetected
- For a frame that is badly garbled, the probability of a column having the correct parity is 0.5 ➔ the probability of accepting the bad block is 2^{-n}.

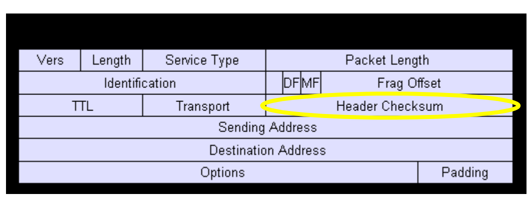


Checksums

- Typically – running sum of data words and appending the complement of the sum to the data block
- Rx-er – computing the running sum of the entire block including the appended checksum
  - If result is zero ➔ No error or undetected error

Example: 16-bit Internet checksum for the IP Protocols
Checksum is computed in 1's complement
Rx-er must get alls ones (i.e. 1111…111) when all words are added for a correct block
Checked on a hop by hop basis

- Does not detect deletion of insertion of zero words or swapping of parts of message
- No zero representation in 1's complement:
  - All zeros
  - All ones



Cyclic Redundancy Checks (CRCs)

- Most popular (very strong) error detection code at data link
  - AKA polynomial code
- Adding r bit CRC bit to m-bit message
- Generator polynomial of degree r, G(x)

  $G(x) = \mathbf{1}\, x^r + g_{r-1}x^{r-1} + \ldots + g_1 x + \mathbf{1}$
- m-bit data message represented as M(x) of degree m-1

CRC Calculation

- M(x) = m-bit data frame – M(x) has degree m-1
- G(x) = Generator polynomial of degree r

  $G(x) = \mathbf{1}\, x^r + g_{r-1}x^{r-1} + \ldots + g_1 x + \mathbf{1}$ - Note that $g_r = g_0 = 1$.
- C(X) is the remainder polynomial of degree r-1 (i.e. r bits long)
- T(X) is the transmitted frame of (m + r) bits

CRC Calculation (2)

- Design: frame T such that it divides the pattern G with no remainder?
- Solution: Since the first component of T, M, is the data part, it is required to find C (or the FCS) such that T divides G with no remainder

Using the polynomial equivalent:

$T(x) = x^r M(x) + C(x)$

One can show that $C(x)$ = remainder of $[x^r M(x)]/G(x)$

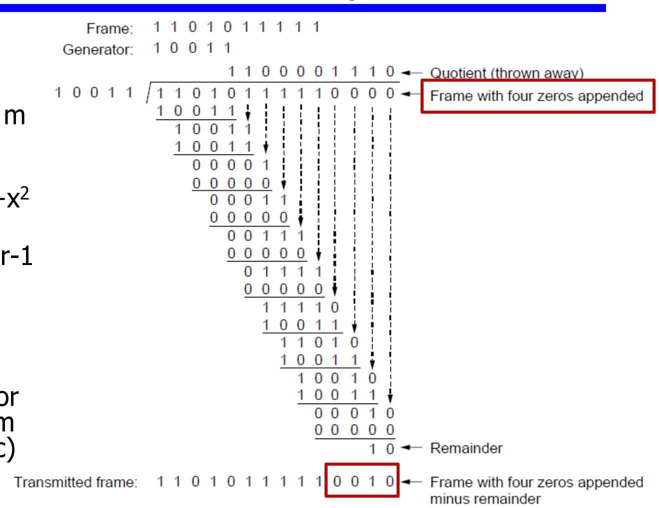
i.e if $x^r M(x) / G(x)$ is equal to $Q(x) + r(x)/G(x)$, then $C(x)$ is set to be equal to $r(x)$.

Note that:

Polynomial of degree m+r-1
------------------------------- = polynomial of degree m + remainder polynomial of degree r-1
Polynomial of degree r

CRC Calculation Example

- $G(X) = x^4+x+1$ – degree r = 4
- Message string = 1101011111 (i.e. m = 10)
- Note $M(x) = x^9+x^8+x^6+x^4+x^3+x^2+x+1$
- $C(x)$ is of degree r-1 at most (i.e. CRC has **r = 4 bits**
- Division can be carried in binary or in polynomial form (mod 2 arithmetic)



Error Detection with CRC

- Transmitted frame (data plus CRC) is T(x)
- Errors on the channel are represented by E(x)
- Received frame is T(x) + E(x)
  - i.e. bit is flipped in the position where error occurs
- The receiver divides {T(x)+E(x)} / G(x) and checks the remainder, R(x)
  - If R(X) is zero → no error or undetectable error
  - If R(X) is nonzero → error detected
- You can show that remainder of {T(x)+E(x)} / G(x) **is simply** remainder of E(x) / G(x)
  - Error patterns that contain G(x) as a factor will pass UNDETECTED!

Design of Generator Polynomial

- Single bit errors: $E(x) = x^i$, for i is the bit error location

If G(x) has at least two non-zero coefficients (coefficient of x^r and coefficient of x^0), then E(x) CANNOT divide G(x) without remainder ➔ All single bit errors will be detected

- Double bit errors: $E(x) = x^i+x^j$, for i>j

One can write $E(x) = x^i+x^j = x^j(x^{i-j}+1)$

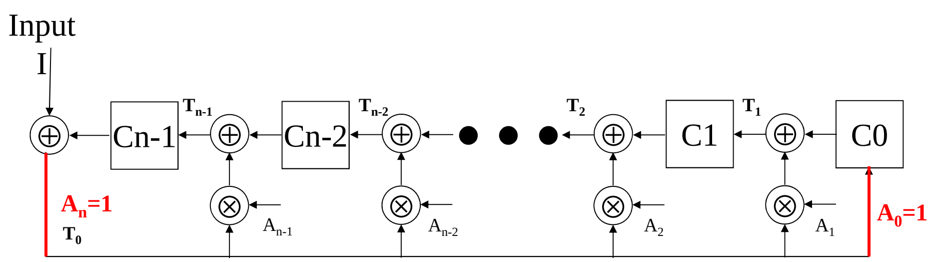
G(x) is not divisible by x – why?

If G(x) does not have a factor $(x^{i-j}+1)$ for any possible i and j in our frame. Meaning G(x) does not divide (x^k+1) for any k up to the maximum value of i-j (i.e. up to the maximum frame length)

- Other properties – check textbook and references

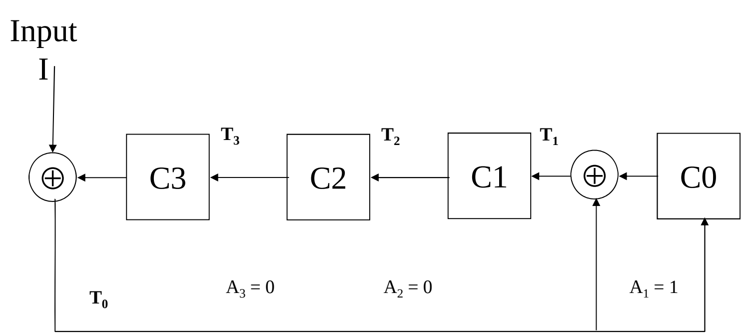
CRC – Shift Register Implementation – General Divisor

- **Divisor of degree n ➔ no of delay elements = n**
- **Note role of coefficients A0, A1, A2, …, An-1, and An.**



General CRC architecture to implement divisor
$G(X) = X^n + A_{n-1}X^{n-1} + \ldots + A_1X^1 + 1$

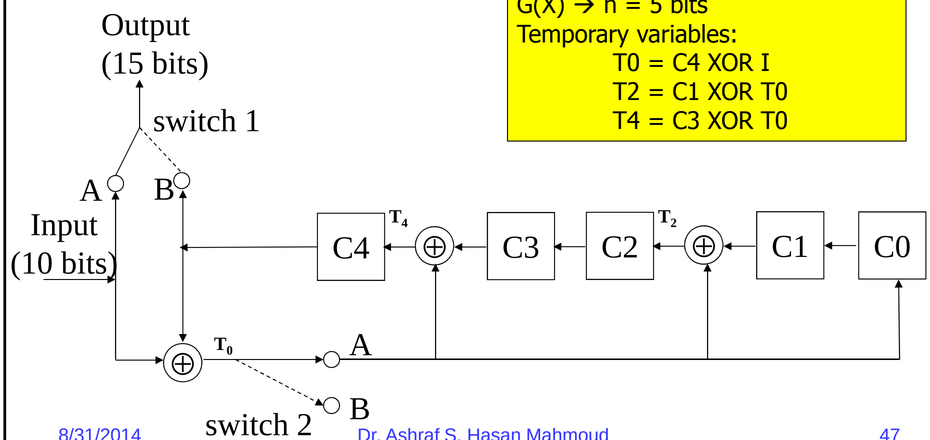
CRC – Shift Register Implementation – Example



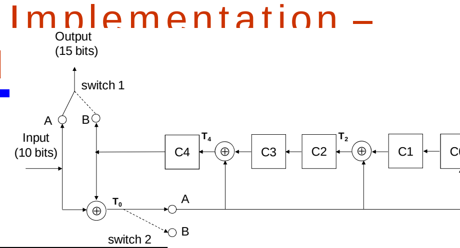
General CRC architecture to implement divisor
$G(X) = X^4 + X^1 + 1$ (Note that A1 = A4 = 1 and A2 = A3 = 0)

CRC – Shift Register Implementation – Example 6.6

Shift register circuit for dividing by
$G(X) = X^5+X^4+X^2+1$
I = [1010001101] → k = 10 bits
G(X) → n = 5 bits
Temporary variables:
T0 = C4 XOR I
T2 = C1 XOR T0
T4 = C3 XOR T0



CRC – Shift Register Implementation – Example 6.6 – cont'd



	C4	C3	C2	C1	C0	T4	T2	T0	I
Initial	0	0	0	0	0	1	1	1	1
step 1	1	0	1	0	1	1	1	1	0
step 2	1	1	1	1	1	1	1	0	1
step 3	1	1	1	1	0	0	0	1	0
step 4	0	1	0	0	1	1	0	0	0
step 5	1	0	0	1	0	1	0	1	0
step 6	1	0	0	0	1	0	0	0	1
step 7	0	0	0	1	0	1	0	1	1
step 8	1	0	0	0	1	1	1	1	0
step 9	1	0	1	1	1	0	1	0	1
step 10	0	1	1	1	0				

MSB

message to
be sent (k = 10 bits)

R

Some Popular CRC Polynomials

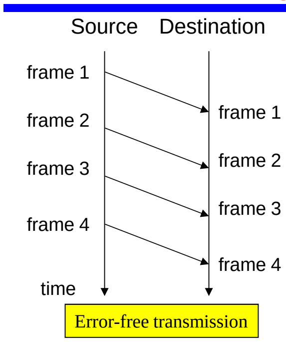
- CRC-12: $x^{12}+x^{11}+x^3+x^2+x+1$
- CRC-16: $x^{16}+x^{15}+x^2+1$
- CRC-CCITT: x16+x12+x5+1
- CRC-32: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

- CRC-12 – used for transmission of streams of 6-bit characters and generates a 12-bit FCS
- CEC-16 and CRC-CCITT – used for transmission of 8-bit characters in USA and Europe – result in 16-bit FCS
- CRC-32 – used in IEEE802 LAN standards – hamming distance of only 4!

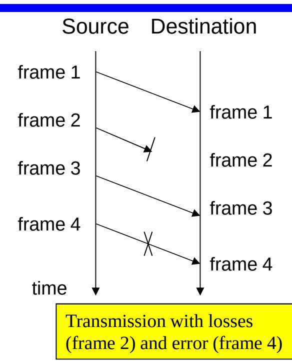
Elementary Data Link Protocols

- Flow control Using Automatic Repeat Request (ARQ) Protocol
  - Stop-and-wait
  - Sliding Window Protocols
- Performance Issues
  - Utilization
  - Link throughput
- Error Control Using ARQ Protocols
- Example Data Link Protocols

Elementary Data Link Protocols



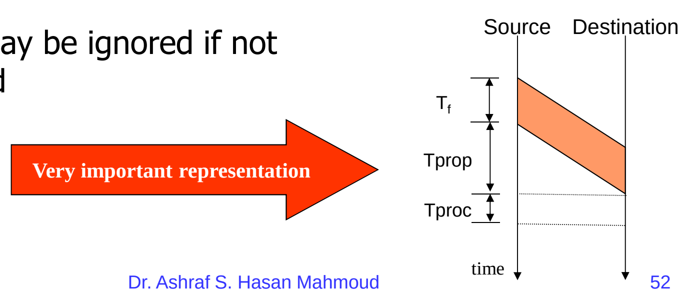
Models of Frame Transmission



- The destination has a limited buffer space. How will the source know that destination is ready to receive the next frame? Need for flow control
- Two types of damaged frames: erroneous frame or frame lost!
- In case of errors or lost frame, the source need to retransmit frames – i.e. a copy of transmitted frames must be kept. How will the source know when to discard copies of old frames?
- Etc.

Timing Diagram for Link Transmission

- Transmission of one frame:
  - T_f: time to transmit frame
  - Tprop: time for signal to propagate
  - Tproc: time for destination to process received frame – small delay (usually ignored if not specified)
- Tproc may be ignored if not specified



Send And Receive Numbers

- Typical frame structure:
  - SN – sequence number for the packet being transmitted
  - RN – sequence number for the NEXT packet in the opposite direction
  - Packet – payload
  - CRC – See previous set of notes
- Piggybacking

	SN	RN		Packet	CRC

What is ARQ?

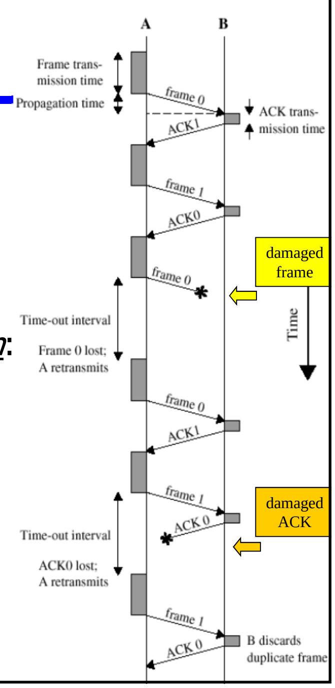
- Def: to detect frames in error and then request the transmitter to repeat the erroneous frames
  - Using ARQ, systems can automatically request the retransmission of missing packets or packets with errors.
- Error Control:
  - ARQ
  - Forward Error Correction – Def = ?
- ARQ Algorithms Figures of Merit
  - Correctness (i.e. only one packet released to upper layer)
  - Efficiency (i.e. throughput)
- Three common schemes
  - Stop & Wait
  - Go Back N
  - Selective

Stop-and-Wait Algorithm

- The simplest ARQ algorithm!
- Operation Rules:
  - Algorithm at node A for A-to-B transmission:
    1. Set the integer variable SN to 0
    2. Accept a packet from the next higher layer at A; if no packet is available, wait until it is; assign number SN to the new packet
    3. Transmit the SNth packet in a frame containing SN in the sequence number field
    4. If an error-free frame is received from B containing a request number RN greater than SN, increase SN to RN and go to step 2, if no such frame is received is received within some finite delay, go to step 3
  - Algorithm at node B for A-to-B transmission
    1. Set the integer variable RN to 0 and then repeat step 2 and 3 forever
    2. Whenever an error-free frame is received from A containing the sequence number SN equal to RN, release the received packet to the higher layer and increment RN
    3. At arbitrary times, but within bounded delay after receiving any error free data from A, transmit a frame to A containing RN in the request number field.
- The Gallager's textbook provides an *informal proof* for the correctness of the above algorithm:
  - Liveness: can continue for ever to accept new packets at A and release them to B
  - Safety: never produces an incorrect result (i.e. never releases a packet out of the correct order to the higher layer)

Modulo 2 Stop-and-Wait ARQ

- Uses Modulo 2 sequence numbers (SN and RN)
  - Both frames and ACKs are numbered
- Two types of errors:
  1. Frame lost or damaged – *Solution*: timeout timer
  2. Damaged or lost ACK – The timeout timer solves this problem



Stop-and-Wait Protocol: Efficiency

- After every frame, source must wait till acknowledgment ➔ Hence link propagation time is significant
- Total time to for one frame:

  T_total = Tf + 2Tprop + Tproc + Tack

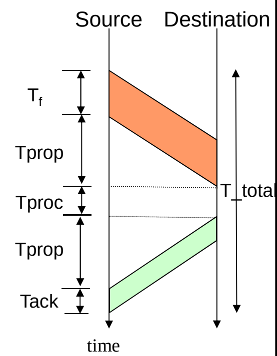
  if we ignore Tproc and Tack (usually very small)

  T_total = Tf + 2Tprop
- Link utilization, U is equal to

  U = Tf/ (T_total), or

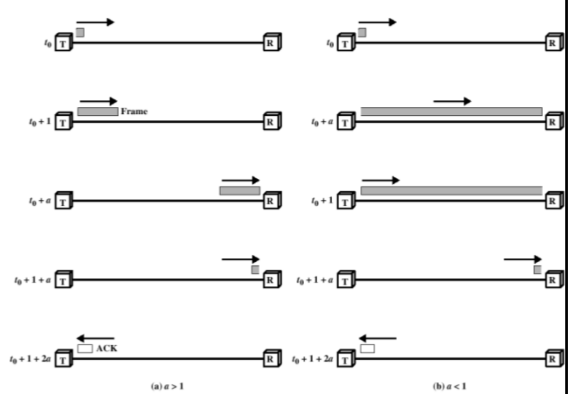
  = 1 / (1+2(Tprop/Tf)) = $\boxed{1 / (1 + 2\,a)}$

  where a = Tprop/Tf = length of link in bits
- If a < 1 (i.e. Tf > Tprop – when 1st transmitted bit reaches destination, source will still be transmitting ➔ U is close 100%
- If a > 1 (i.e. Tf < Tprop – frame transmission is completed before 1st bit reaches destination ➔U is low



Stop-and-Wait Protocol: Efficiency (2)

- Remember: a = Tprop/Tf = length of link in bits
- If a < 1 (i.e. Tf > Tprop – when 1st transmitted bit reaches destination, source will still be transmitting ➔ U is close 100%
- If a > 1 (i.e. Tf < Tprop – frame transmission is completed before 1st bit reaches destination ➔U is low
- Stop-and-Wait is efficient for links where a << 1 (long frames compared to propagation time)



Stop-and-Wait Protocol: Efficiency With Errors (3)

- Assume a frame is in error with probability P
- Therefore, average utilization can be written as

$$U = Tf / (Nr \times T\_total)$$

- Nr is the average number of transmissions of a frame, while T_total is equal to Tf + 2Tprop.
- For stop-and-wait, Nr is given by

$$\begin{aligned} Nr &= E[\text{no of transmissions}] = \Sigma\, i \times Prob[\ i \text{ transmissions}] \\ &= \Sigma\, i \times P^{i-1}(1-P) \\ &= 1/(1-P) \end{aligned}$$

- Therefore, utilization is given by

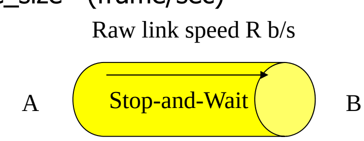
$$\boxed{U = (1-P)/(1+2a)}$$

Identities:
$\Sigma\,(X^{i-1}, i=1,\infty) = 1/(1-X)$ for $-1<X<1$
$\Sigma\,(iX^{i-1}, i=1,\infty) = 1/(1-X)^2$ for $-1<X<1$

- Note that for P = 0 (i.e. error free), the expression reduced to the previous result!

Important Performance Figures

- Utilization (U) – fraction of time the link is used for transmitting data
- Throughput (b/s) – effective b/s as experienced by user data
  - Throughput = R * U (b/s)
- Throughput (frame/s) – average data frames per second the link is supporting
  - Throughput = 1/T_total (frame/sec)
  - = R*U/data_frame_size (frame/sec)



Utilization = ?
Throughput = ?

Sliding Window Protocol

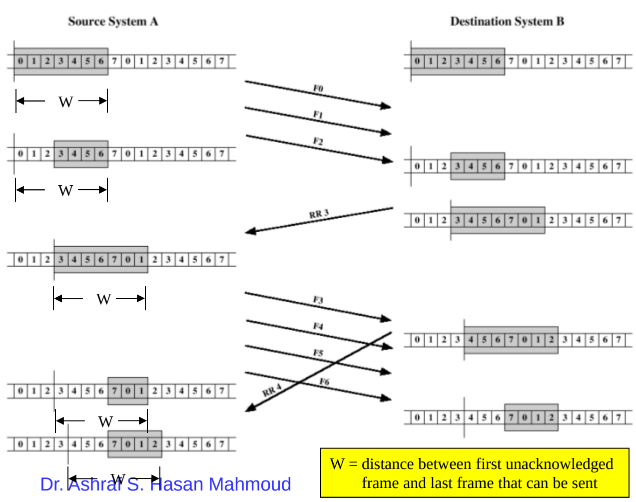
- Stop-and-Wait can be very inefficient when a > 1
- Protocol:
  - Assumes full duplex line
  - Source A and Destination B have buffers each of size W frames
  - For k-bit sequence numbers:
    - Frames are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
    - ACKs (RRs) are numbered: 0, 1, 2, …, 2^k-1, 0, 1, … (modulo 2^k)
  - A is allowed to transmit up to W frames without waiting for an ACK
  - B can receive up to W consecutive frames
  - ACK J (or RR J), where 0<=J<= 2^k-1, sent by B means B is have received frames up to frame J-1 and is ready to receive frame J
- Window size, W can be less or equal to 2^k-1

Sliding Window Protocol (2)

- Example of Sliding-Window-Protocol: k = 3 bits, W = 7

Observations:

- A may tx W = 7 frames (F0, F1, …, F6)
- After F0, F1, & F2 are tx-ed, window is shrunk (i.e. can not transmit except F3, F4, …, F6)
- When B sends RR3, A knows F0, F1 & F2 have been received and B is ready to receive F3
- Window is advanced to cover 7 frames (starting with F3 up to F1)
- A sends F3, F4, F5, & F6
- B responds with RR4 when F3 is received – A advances the window by one position to include F2



Sliding Window Protocol - Piggybacking

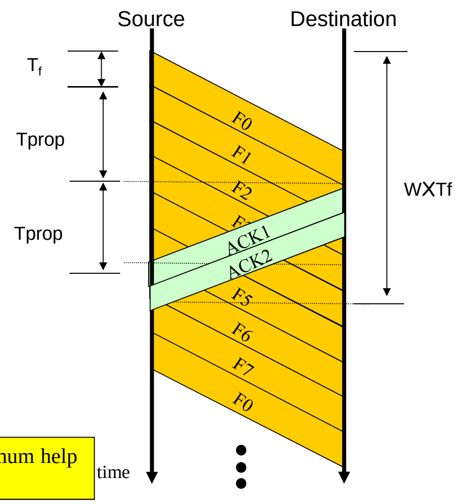
- When using sliding window protocol in full duplex connections:
  - Node A maintains its own transmit window
  - Node B maintains its own receive window
  - A frame contains: data field + ACK field
  - There is a sequence number for the data field, and a sequence number for the ACK field

Sliding Window Protocol - Efficiency

- Again we can distinguish two cases:
- Case 1: $W \geq 2a + 1$
- Case 2: $W < 2a + 1$

Sliding Window Protocol - Efficiency - Case 1

- Assume k=3, W = 7
  (ignoring Tack)
- Source can continuously keep transmitting!!
  - Because the ACK can arrive to source before the window is completed
- Utilization = 100%

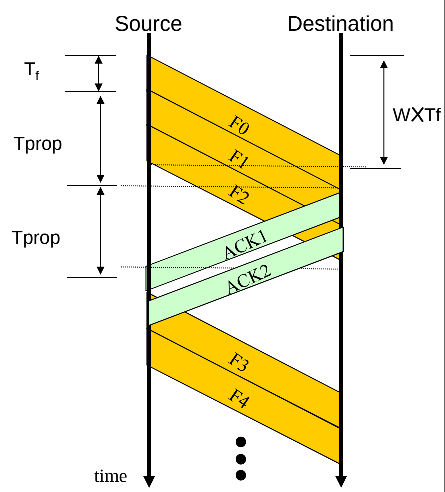


Sending ACK0 as soon as F0 is received is the maximum help the destination can do to increase utilization

Sliding Window Protocol - Efficiency - Case 2

- Assume k = 3, W = 3 (ignoring Tack)
- Source can NOT continuously keep transmitting!!
  - Because the ACK can NOT arrive to source before the window is completed
- Utilization = $\frac{W \times Tf}{Tf + 2 \times Tprop}$

  $= \frac{W}{1 + 2a}$



Sliding Window Protocol - Efficiency

- Refer to Appendix A
- When window size is W (for error free), link utilization, U, is given by

$$U = \begin{cases} 1 & W \geq (2a+1) \\ \dfrac{W}{2a+1} & W < (2a+1) \end{cases}$$

where a = Tprop/Tf or length of link in bits

- Sliding window protocol can achieve 100% utilization if W >= (2a + 1)

Sliding Window Protocol

- Animation for Sliding Window protocol
- Sliding Window Protocol Simulation (http://www.cs.stir.ac.uk/~kjt/software/comms/jasper/SWP3.html)

Go-Back-N ARQ

- Based on the sliding-window flow control procedure - Sliding Window Protocol slide
- Three types of errors:
  1. i^{th} frame damaged:
     a. If A send subsequent frames (i+1, i+2, …), B responds with REJ i ➔ A must retransmit i^{th} frame and ***all subsequent frames***
     b. If A does not send subsequent frames and B does not respond with RR or REJ (since frame was damaged) ➔ timeout timer at A expires – send a POLL signal to B; B sends an RR i, i.e. it expect the i^{th} frame – A sends the i^{th} frame again

Check for status of B before resending the frame

  2. Damaged RR (B receives i^{th} frame and sends RR i+1 which is lost or damaged):
     a. Since ACKs are cumulative – A may receive a subsequent RR j (j >i+1) before A times out
     b. If A times out, it sends a POLL signal to B – if B fails to respond (i.e. down) or its response is damaged subsequent POLLs are sent; procedure repeated certain number of time before link reset
  3. Damaged REJ – same as 1.b

Go-Back-N ARQ – Efficiency With Errors

- Remember that Go-back-N ARQ utilization for error-free channels is given by:

  U = 1 for W > 2a + 1
  = W/(2a+1) for W < 2a + 1

- Assume a data frame can be in error with probability P
- With Go-back-N if one frame in error, we may retransmit a number of frames, on average K, and NOT only one!
- The average number of transmitted frames to transmit one frame correctly, Nr, is given by

  Nr = E[number of transmitted frames to successfully transmit one frame]
  = Σ f(i) x $P^{i-1}(1-P)$

- If a frame is transmitted i times (i.e. first (i-1) times are erroneous while it was received correctly in the i^{th} time), then f(i) is the total number of frame transmissions if our original frame is in error.
- f(i) is given by

  f(i) = 1 + (i-1)K

- Substituting f(i) in the above relation, yields

  Nr = (1-P+KP)/(1-P)

Identities:
Σ (X^{i-1},i=1,∞) =1/(1-X) for -1<X<1
Σ (iX^{i-1},i=1,∞) =1/(1-X^2) for -1<X<1

- Examining the operation of Go-back-N, an approximate value for K is 2a+1
- Then utilization with errors is given by

  U = (1-P)/(1+2aP) for W > 2a+1
  (1-P)W/{(2a+1)(1-P+WP)} for W < 2a+1

Again, expression reduces to the previous result if you set P = 0

Selective-Reject ARQ

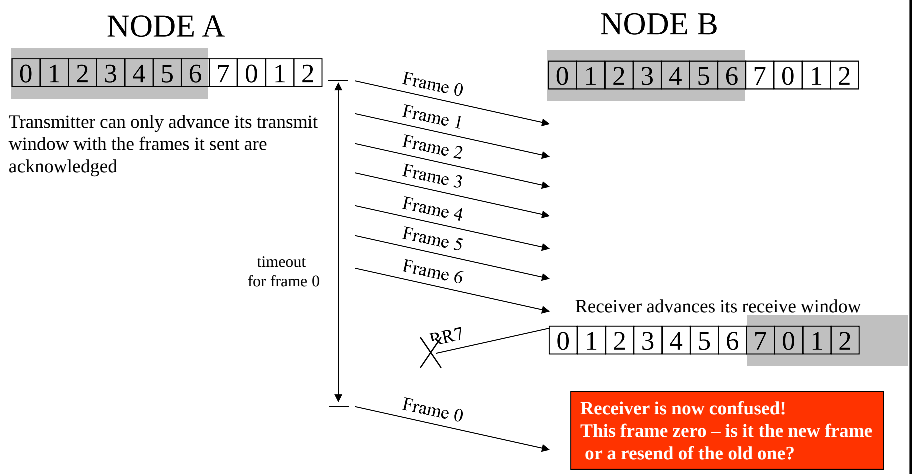
- In contrast to Go-Back-N, the only frames retransmitted are those that receive –ve ACK (called SREJ) or those that time out
- More efficient:
  - Rx-er must have large enough buffer to save *post*-SREJ frames
  - Buffer manipulation – re-insertion of out-of-order frames

Window Size for Selective-Reject ARQ – Why?

- Window size: should less or equal to half range of sequence numbers
  - For n-bit sequence numbers, Window size is $\leq 2^{n-1}$ (remember sequence numbers range from 0,1, …, 2^n-1)
- Why? See next example

Window Size for Selective-Reject ARQ – Why? (2)

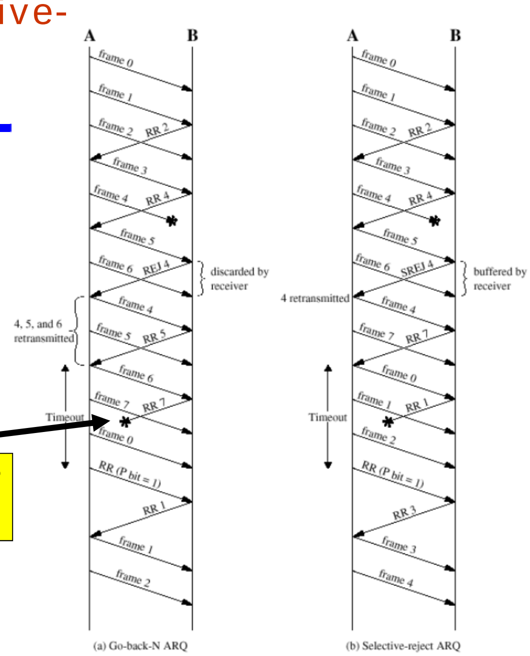
- Example: Consider 3-bit sequence number and window size of 7



Go-Back-N/Selective-Reject ARQ Examples

- With Go-back-N frames 4,5 and 6 are retransmitted
- With Selective-Reject only frame 4 is retransmitted

Did this lost RR7 affect flow?
How did the link recover?



Selective Reject ARQ – Efficiency With Errors

- Remember that Selective Reject utilization for error-free channels is given by:

$$U = 1 \quad \text{for } W > 2a + 1$$
$$\;\; = W/(2a+1) \quad \text{for } W < 2a + 1$$

- Assume a data frame can be in error with probability P
- With Selective Reject if one frame in error, we retransmit only the required frame
- The average number of transmitted frames to transmit one frame correctly, Nr, is given by

Nr = E[number of transmitted frames to successfully transmit one frame]
$= \Sigma\, i \times P^{i-1}(1-P) = 1/(1-P)$

- Then utilization with errors is given by

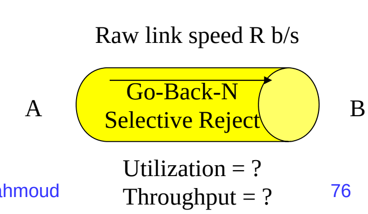
$$\boxed{\begin{array}{ll} U = 1-P & \text{for } W > 2a+1 \\ \quad (1-P)W/(2a+1) & \text{for } W < 2a+1 \end{array}}$$

Identities:
$\Sigma\,(X^{i-1}, i=1,\infty) = 1/(1-X)$ for $-1<X<1$
$\Sigma\,(iX^{i-1}, i=1,\infty) = 1/(1-X^2)$ for $-1<X<1$

Again, expression reduces to the previous result if you set P = 0

Important Performance Figures - Again

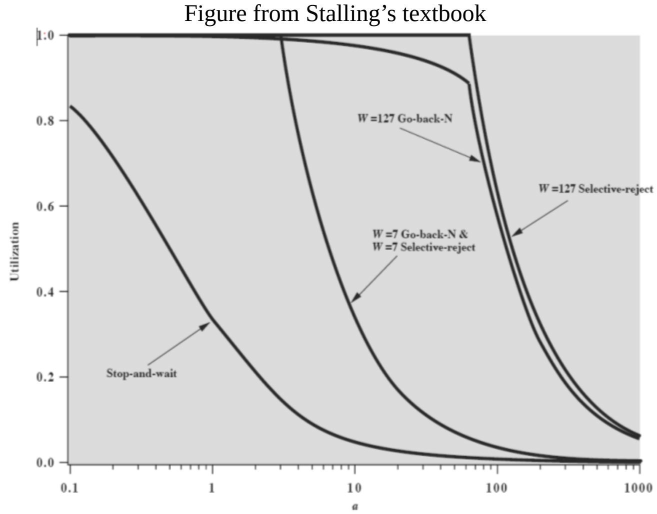
- Utilization (U) – fraction of time the link is used for transmitting data
- Throughput (b/s) – effective b/s as experienced by user data
  - Throughput = R * U (b/s)
- Throughput (frame/s) – average data frames per second the link is supporting
  - If U is equal to 100%
    ➔ Throughput = 1/Tf (frame/sec)
    = R*U/data_frame_size (frame/sec)
  - If U is LESS than 100%
    ➔ Throughput = W/T_totla (frame/sec)
    = R*U/data_frame_size (frame/sec)



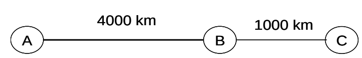
ARQ Utilization as a Function of a

- Remember a is given by Tprop/Tf – i.e. the length of the link in bits
- The curves are for P = 10^{-3}
- Note for W = 1, go-back-N and selective reject degenerate to the case of stop-and-wait
- Please note that the previous analyses are only approximate – errors in ACKs were ignored. Furthermore, in the case of go-back-N, errors in retransmitted frames other than the frame of interest were also ignored

Figure from Stalling's textbook



Example:



Problem: In the shown figure, frames are generated at node A and send to node C through node B. The following specifies the two communication links:

- **The data rate between node A and node B is 100 kb/s**
- **The propagation delay is 5 μsec/km for both links**
- **Both links are full-duplex**
- **All data frames are 1000 bits long; ACK frames are separate frames of negligible length**
- **Between A and B sliding window protocol with a window size of 3 is used**
- **Between B and C, stop-and-wait is used.**

There are no errors (lost or damaged frames)

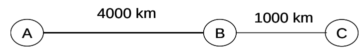
a) Calculate the utilization for link AB?

b) What is the throughput for link AB in bits per second? What is the throughput in frames per second?

c) Calculate the minimum rate required between nodes B and C so that the buffers of node B are not flooded.

d) What is the efficiency of the communication on link BC?

Example:



Solution:

Link AB: T_{f_AB} = frame length / R_{AB} = 1000/100 = 10 msec

T_{prop_AB} = 4000 km X 5 µsec = 20 msec

Link BC: T_{f_BC} = frame length / R_{BC} = 1000 / R_{BC}

T_{prop_BC} = 1000 km X 5 µsec = 5 msec

a) $a_{AB} = T_{prop\_AB} / T_{f\_AB}$ = 20 / 10 = 2

W = 3 is equal or less than (2 X a_{AB} + 1) = 5

➔ Utilization = W/(2 X a_{AB} + 1) = 3/5 = 60%

b) Throughput = 100 X 0.6 = 60 kb/s;

Throughput = 60 kb/s / (1000 bit) = 60 frame/second

c) Throughput for link BC in frames/second = 1 /(T_{f_BC} + 2XT_{prop_BC})

= 1/(1000/R_{BC} + 2X5X10^{-3})

= 1/(1000/R_{BC} + 10^{-2})

For not overflowing: frame throughput for link AB should be less or equal to frame throughput for link BC

➔ 60 <= 1/(1000/R_{BC} + 10^{-2})

1/60 >= 1000/ R_{BC} + 10^{-2}

1/60 - 10^{-2} >= 1000/ R_{BC}

R_{BC} >= 1000/(1/60 - 10^{-2}) = 150 kb/s

d) T_{f_BC} = 1000/150 kb/s = 6.667 msec

Efficiency (utilization) of link BC: $a_{BC} = T_{prop\_BC}/T_{f\_BC}$ = 5/6.666 = 0.75;

➔ Efficiency = 1/(2a+1) = 1/(2*0.75+1) = 40%

Example Data Link Protocols

- HDLC – basis for many of the currently operating data link protocols
- Point-to-Point Protocol (PPP) – considered in two scenarios:
  - Packets over SONET optical fiber links – Two ISP routers in a WAN configuration
  - ADSL link for a telephone local loop – edge of the internet

High-Level Data Link Control Protocol (HDLC)

- One of the most important data link control protocols
- Basic Characteristics:
  - Primary Station: issues *commands*
  - Secondary Station: issues *responses* – operates under the control of a primary station
  - Combined Station: issues commands and responses
- Two link configurations are defined:
  - Unbalanced: one primary plus one or more secondary
  - Balanced: two combined (functions as primary and/or secondary) stations

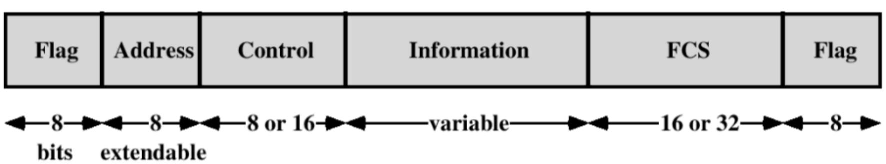
High-Level Data Link Control Protocol (HDLC) (2)

- Three transfer modes are defined:
  - Normal Response Mode (NRM) – used in unbalanced conf.; secondary may only tx data in response to a command from primary
  - Asynchronous Balanced Mode (ABM) – used in balanced conf.; either combined station may tx data without receiving permission from other station
  - Asynchronous Response Mode (ARM) – used in unbalanced conf.; Secondary may initiate data tx without explicit permission; primary still retains line control (initialization, error recovery, …)
- Animation for HDLC

HDLC - Applications

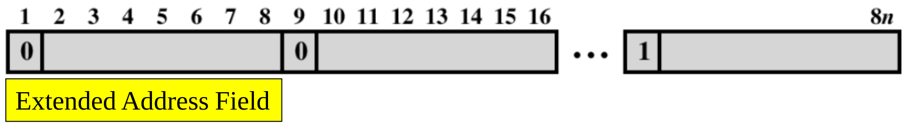
- NRM:
  - Point-multi-point (multi-drop line): one computer (primary) polls multiple terminals (secondary stations)
  - Point-to-point: computer and a peripheral
- ABM: most widely used (no polling involved)
  - Full duplex point-to-point
- ARM: rarely used

HDLC – Frame Structure – Flag Field



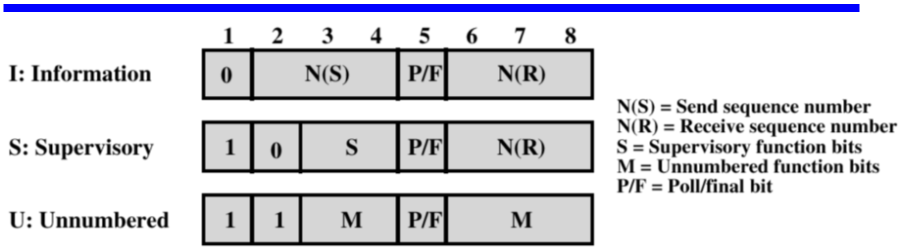
- Flag Field: unique pattern 01111110
  - Used for synchronization
  - To prevent this pattern form occurring in data ➔ *bit stuffing*
    - Tx-er inserts a 0 after each 5 1s
    - Rx-er, after detecting flag, monitors incoming bits – when a pattern of 5 1s appears; the 6th/7th bit are checked:
      - If 0, it is deleted
      - If 10, this is a flag
      - If 11, this is an ABORT
- Pitfalls of bit stuffing: one bit errors can split one frame into two or merge two frames into one

HDLC – Frame Structure - Address Field



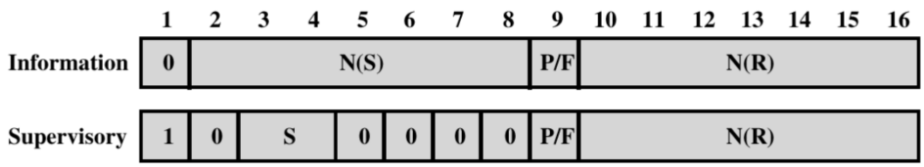
- Address field identifies the secondary station that transmitted or is to receive frame
- Not used (but included for uniformity) for point-to-point links
- Extendable – by prior arrangement
- Address = 11111111 (single octet) used for broadcasting; i.e. received by all secondary stations

HDLC – Frame Structure - Control Field



- First 2 bits of field determine the type of frame
  - Information frame (I): carry user data (upper layers) – flow and error control info is piggybacked on these frames as well
  - Supervisory frame (S): carry flow and error control info when there is no user data to tx
  - Unnumbered frame (U): provide supplementary link control
- Poll/Final (P/F) bit:
  - In command frames (P): used to solicit response from peer entity
  - In response frames (F): indicate response is the result of soliciting command

HDLC – Frame Structure - Control Field (2)



- "Set-mode" command ➔ extends control field to 16 bit for S and I frames
- Extension: 7-bit sequence numbers rather than 3-bit ones
- Unnumbered frames always use 3-bit sequence numbers

HDLC – Frame Structure – Information/FCS Fields

- Information field:
  - Present ONLY in I-frames and some U-frames
  - Contains integer number of octets
  - Length is variable – up to some system defined maximum
- FCS field:
  - Error detecting code
  - Calculated from *ALL* remaining bits in frame
  - Normally 16 bits (CRC-CCITT polynomial = $X^{16}+X^{12}+X^5+1$)
  - 32-bit optional FCS

HDLC Operation

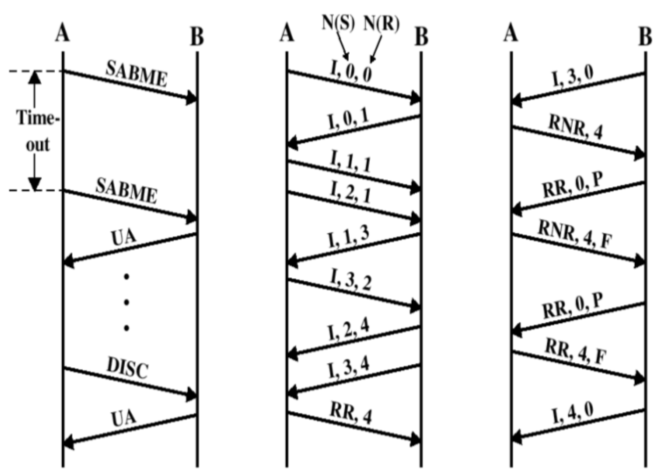
- Initialization
  - One side signals to the other the need for initialization
  - Specifies which of the three modes to use: NRM, ABM, or ARM
  - Specifies 3- or 7-bit sequence numbers
  - The other side can accept by sending unnumbered acknowledgment (UA)
  - The other side can reject by sending - A disconnected mode (DM) frame is sent
- Data Transfer
  - Exchange of I-frames: data and can perform flow/error control
  - S-frames can be used as well: RR, RNR, REJ, or SREJ
- Disconnect
  - DISC frame ➔ UA

HDLC – Operation

a) Link Setup & Disconnect:
- SABM command – starts timer
- B responds with UA (or DM if not interested)
- A receives UA and initializes its variables
- To disconnect: issue DISC command

b) Two-Way Data Exchange:
- Full-duplex exchange of I-frames

c) Busy Condition:
- Note the use of the P and F bits

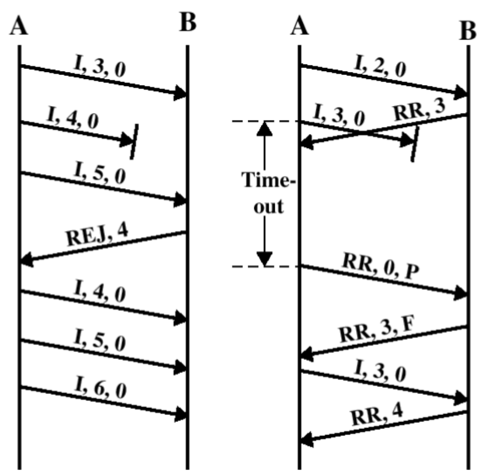


(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition

HDLC – Operation (2)

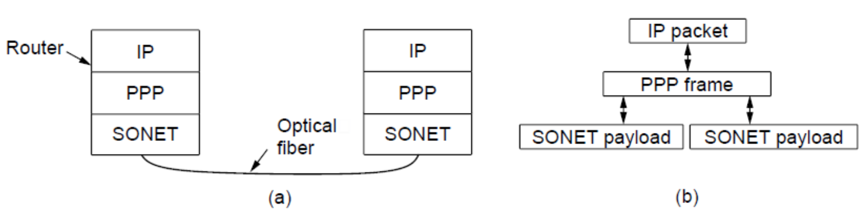
a) Reject Recovery:
- I-frame 4 was lost
- B receives I-frame 5 (out of order) – responds with REJ 4
- A resend I-frame 4 and all subsequent frames (Go-back-N)

b) Timeout Recovery:
- A sends I-frame 3 – but it is lost
- Timer expires before acknowledgement arrives
- A polls Node B
- B responds indicating it is still waiting for frame 3 – B set the F bit because this a response to A's solicitation



Packet Over SONET

- Runs on wide-area optical fiber links – backbone of communications networks
- Bit stream at a well defined rate – e.g. 2.4Gb/s for OC-48 link
  - Organized as fixed-size byte payloads every 125 microsec – where there is data or not
- PPP provides a mechanism of carry IP packets over these links



Packet over SONET. (a) A protocol stack. (b) Frame relationships

Point-to-Point Protocol (PPP)

- Defined in RFC 1661 and RFC 1662
- Extension to an earlier protocols called Serial Line Internet Protocol
- PPP handles link configuration, support of multiple protocols, authentication, etc.
- Main features:
  - Unambiguous framing with error detection capability
  - Link control for bringing lines up, testing them, negotiating options, etc. ➔ Link Control Protocol (LCP)
  - Allows to negotiate network-layer options ➔ Network Control Protocol (NCP)
- PPP is similar to HDLC
  - PPP is byte oriented
  - "Unnumbered mode" in PPP is commonly used to provide connectionless unacknowledged service.

PPP Frame Structure

- Flag byte 0x7E – similar to HDLC
  - Escape character 0x7D for byte stuffing
  - Escaped byte (i.e. the non legitimate flag is XORed with 0x20 resulting flipping the 5th bit) – therefore 0x7E will not appear in the payload
  - Facilitates scanning for start and end of frame
- If no activity on the link – send flags
- Address field
  - 0xFF – broadcast
- Control field
  - Default – 00000011 → unnumbered frame
- LCP may negotiate to eliminate the address and control fields (if they are constants)
- Protocol: Code specifying the nature of the payload
  - Codes starting with 0 – IPv4, IPv6, IPX, AppleTalk, etc.
  - Codes starting with 1 – PPP configuration protocols including LCP and different NCP for each network layer protocol supported
- Payload – variable length to some negotiated maximum sum
  - Default max is 1500 Bytes
- Checksum – 4 bytes CRC or 2 bytes CRC

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

The PPP full frame format for unnumbered mode operation

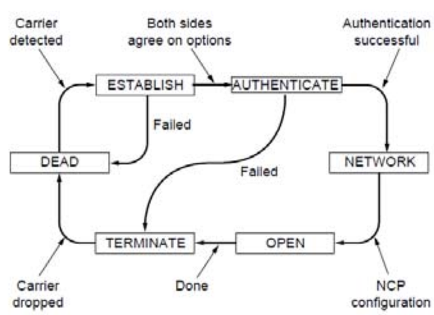
PPP Over SONET

- Specified in RFC 2615
- Recommendations
  - Use of 4-bytes checksums
  - Not compressing the address and control fields
- PPP payload is scrambled (refer to textbook section 2.5.1) before insertion to SONET
  - Long pseudorandom sequence to eliminate long runs of 0s

PPP States

- Link starts in DEAD state
- ESTABLISHED – physical layer connection is made
- Series of LCP packets to configure the PPP link
- Successful LCP negotiation leads to AUTHENTICATE

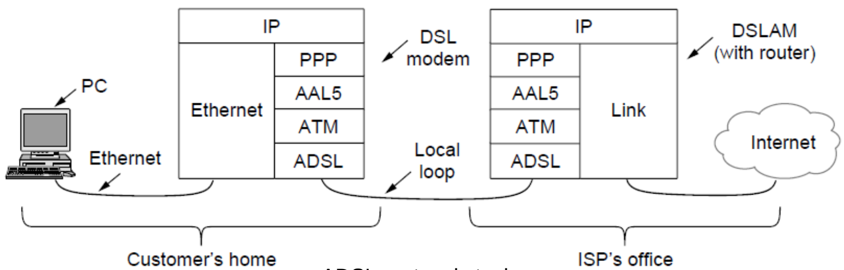
- Successful authentication leads to NETWORK
- Series of NCP packets are exchanged depending on the network-layers involved
  - For example – for IP protocol, assignment is IP addresses for the two link ends is done at this stage
- Data transport occurs in the OPEN state
- TERMINATE is used to bring down the link



State diagram for bringing a PPP link up and down

Asymmetric Digital Subscriber Loop (ADSL)

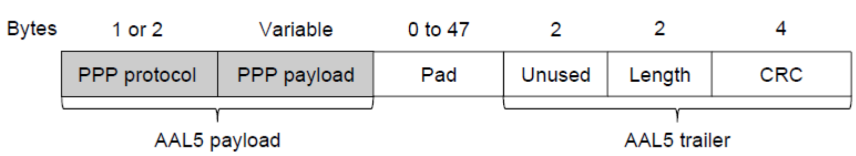
- Link between the ADSL modem (home side) and the DSLAM in the telephone local office
- Physical layer – OFDM (DMT signaling)
- PPP provides the link configuration and control



ADSL protocol stacks.

ATM and AAL5

- Asynchronous Transfer Mode (ATM) – connection-oriented (fixed payloads of 48 bytes + 5 bytes header)
  - Asynchronous
  - Virtual circuit
- AAL5 – ATM adaptation layer 5 used to map data to ATM cells
  - Multiple adaptation layers are defined for ATM – depending on the payload type (periodic, circuit, data, etc.)
- 4-byte CRC
- Padding – rounds overall length to multiple of 48
- No addresses – virtual circuit identifier in the cell header will suffice
- Only the PPP protocol and payload are transported!



AAL5 frame carrying PPP data