Data Link Control Layer, Error Detection, Error Correction, and Framing

EEE 538, WEEK 2

Dr. Nail Akar

Bilkent University

Electrical and Electronics Engineering Department

Error Detection Techniques

- Used by the receiver to determine if a packet contains errors
- If a packet is found to contain errors the receiver requests the transmitter to re-send the packet
- Error Detection Techniques
  - Parity Check
    - Single bit
    - Horizontal and vertical redundancy check
  - Cyclic Redundancy Check (CRC)
- Assume initially that the receiving DLC module knows where frames begin and end

Effectiveness of an Error Detection Technique

- **minimum distance of code (d)** (min # bit errors undetected) The minimum distance of a code is the smallest number of errors that can map one codeword onto another. If fewer than d errors occur they will always detected. Even more than d errors will often be detected (but not always!)
- **burst detecting ability (B)** (max burst length always detected)
- **probability of random bit pattern mistaken as error free** (good estimate if # errors in a frame >> d or B)

Parity Check Codes

K data bits	L parity check bits

- Each parity check is a modulo 2 sum of some of the data bits
- Example
  - C1 = x1 + x2 + x3
  - C2 = x2 + x3
- Internet checksum

Single Parity Check Code

- The check bit is 1 if frame contains odd number of 1's; otherwise it is 0
  - 1011011 -> 1011011 1
  - 1100110 -> 1100110 0
- Thus, encoded frame contains even number of 1's
- Receiver counts number of ones in frame
  - An even number of 1’s is interpreted as no errors
  - An odd number of 1’s means that an error must have occurred. A single error (or an odd number of errors) can be detected. An even number of errors cannot be detected. Nothing can be corrected
  - Probability of undetected error (independent errors)

$$P(undetected) = \sum_{i\ even} \binom{N}{i} p^i (1-p)^{N-i}$$

N = packet size
p = error prob.

Horizontal and Vertical Parity

1	0	0	1	0	1	0	1	Horizontal checks
0	1	1	1	0	1	0	0	
1	1	1	0	0	0	1	0	
1	0	0	0	1	1	1	0	
0	0	1	1	0	0	1	1	
1	0	1	1	1	1	1	0	

Vertical checks

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

- The data is viewed as a rectangular array (i.e., a sequence of words)
- Minimum distance=4, any 4 errors in a rectangular configuration is undetected

Cyclic Redundancy Check

M	C
K data bits	L parity check bits

$$T = M\ 2^L + C$$

T: transmitted data, codeword

M: message, information bits

C: parity bits

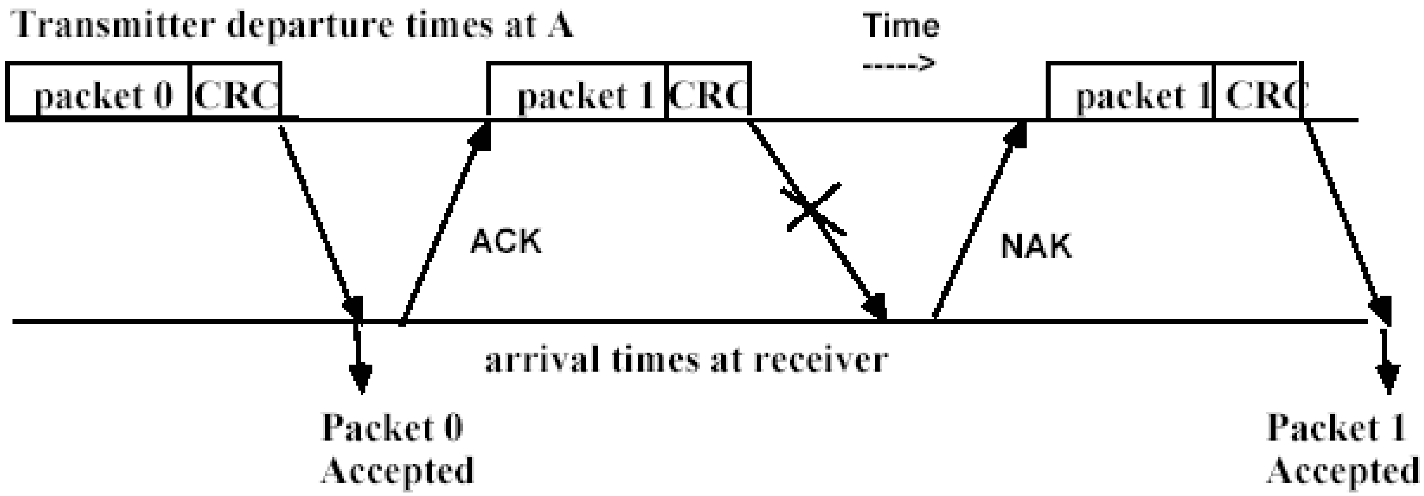
Physical Layer Error Characteristics

- Most Physical Layers (communications channels) are not well described by a simple BER parameter
- Most physical error processes tend to create a mix of random & bursts of errors
- A channel with a BER of 10^{-7} and a average burst size of1000 bits is very different from one with independent random errors
- Example: For an average frame length of 10^4 bits
  - random channel: E[Frame error rate] ~ 10^3
  - burst channel: E[Frame error rate] ~ 10^-6
- Best to characterize a channel by its Frame Error Rate
- This is a difficult problem for real systems

Automatic Repeat Request ARQ

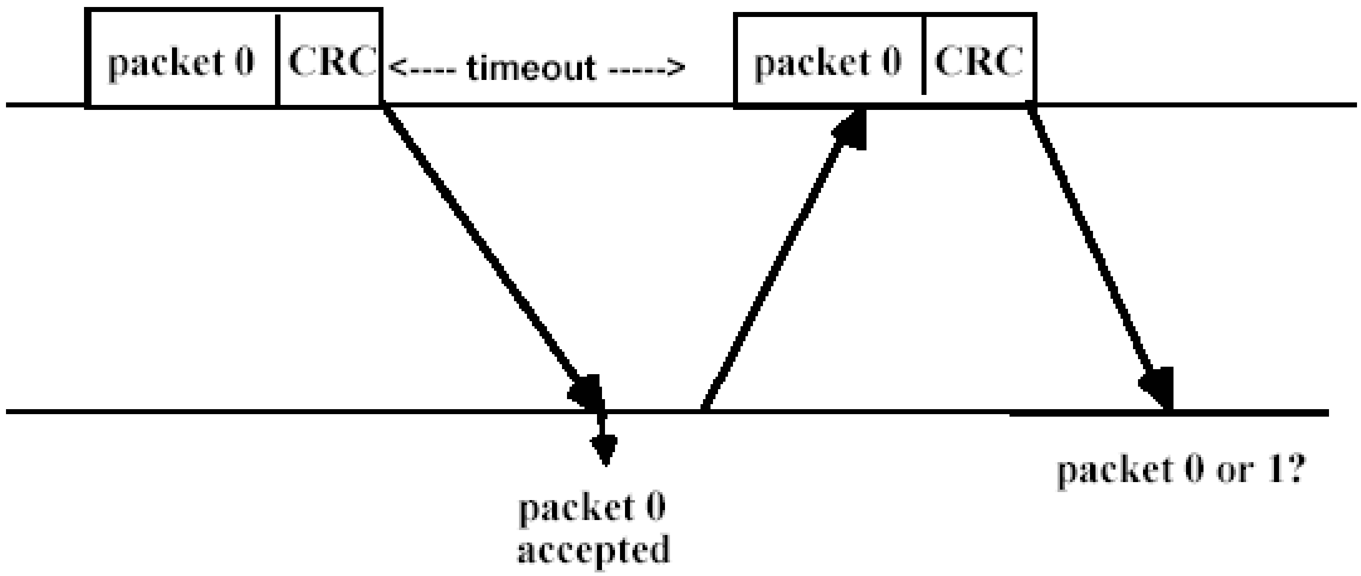
- When the receiver detects errors in a packet, how does it let the transmitter know to re-send the corresponding packet?
- Systems which automatically request the retransmission of missing packets or packets with errors are called ARQ systems.
- Three common schemes
  - Stop & Wait
  - Go Back N
  - Selective Repeat

Pure Stop and Wait



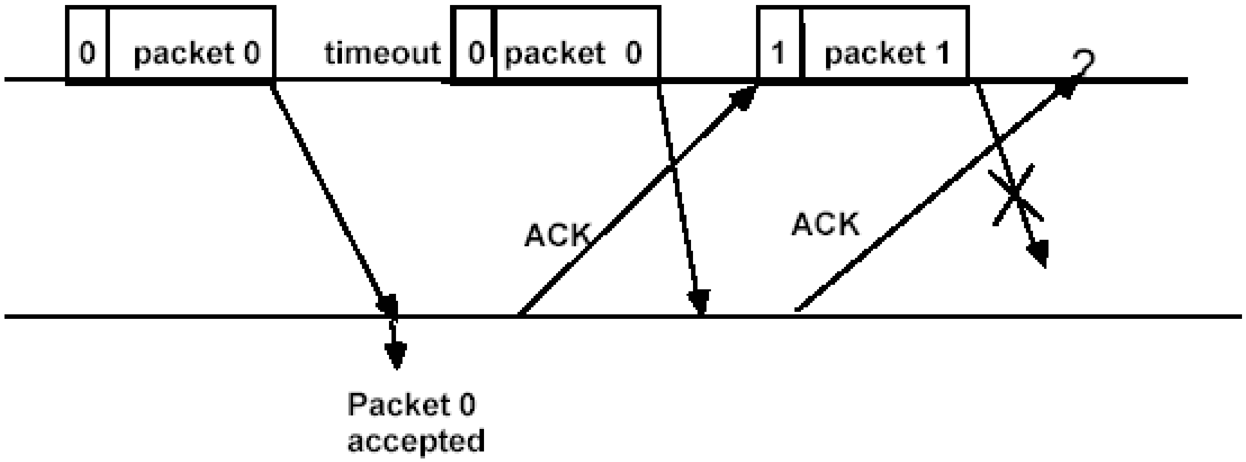
- Problem: Lost packets
- Sender will wait forever for an acknowledgement
- Packet may be lost due to framing errors
- Solution: Use time-out (TO)
- Sender retransmits the packet after a timeout

Sequence Numbers



- Problem: Unless packets are numbered the receiver cannot tell which packet it received
- Solution: Use packet numbers (sequence numbers)

Request Numbers are Required on ACKs



- As opposed to sending an ACK or a NAK, the receiver sends the number of the packet currently awaited
- Sequence numbers and request numbers can be sent modulo 2, avoiding the need to use large sequence numbers

Algorithm at Sender

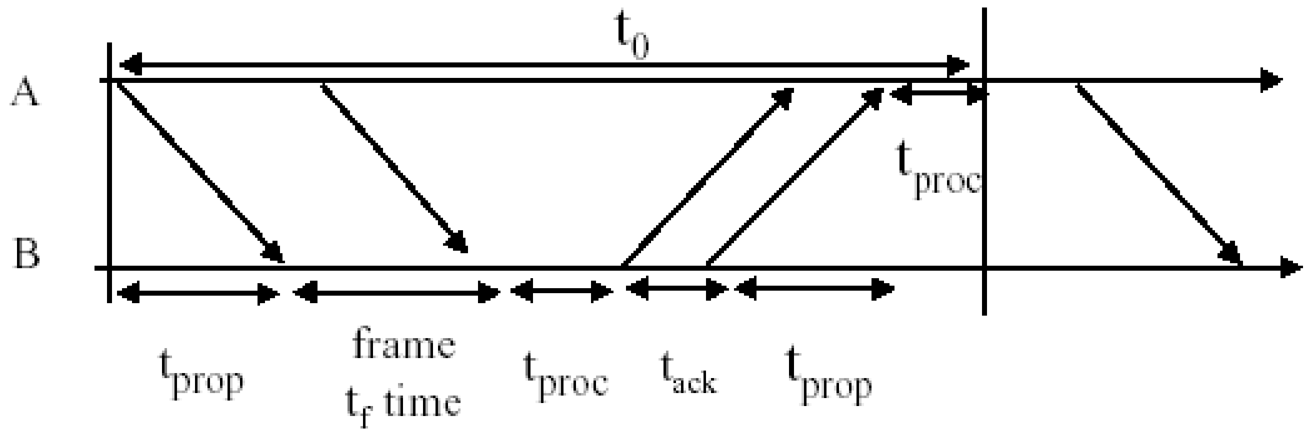
- Initial condition SN = 0
- 1) Accept packet from higher layer when available; assign number SN to it
- 2) Transmit packet SN in frame with sequence # SN
- 3) Wait for an error free frame from B
  - i. if received and it contains RN>SN in the request # field, set SN to RN and go to 1
  - ii. if not received within given time, go to 2

Algorithm at Receiver

- Initial condition RN = 0
- 1) Whenever an error-free frame is received from A with a sequence # equal to RN, release received packet to higher layer and increment RN

- At arbitrary times, but within bounded delay after receiving any error free frame from A, transmit a frame to A containing RN in the request # RN

Efficiency of Stop and Wait



t_{prop} = propagation delay

t_{proc} = processing delay

n_f = # bits in a frame (assume constant)

n_a = # bits in ACK/NAK frame

R = bit rate

Time to send frame (in absence of errors) and receive ACK:

$$t_0 = 2\,(t_{prop} + t_{proc}) + (n_f + n_a)/R$$

Stop and Wait

Effective transmission rate

$$R_{eff}^{0} = \frac{\text{number of information bits delivered to destination}}{\text{total time required to deliver the information bits}} = \frac{n_f - n_o}{t_0},$$

Transmission efficiency

$$\eta_0 = \frac{\dfrac{n_f - n_o}{t_0}}{R} = \frac{1 - \dfrac{n_o}{n_f}}{1 + \dfrac{n_a}{n_f} + \dfrac{2(t_{prop} + t_{proc})R}{n_f}}.$$

Delay-Bandwidth Product

Stop and Wait

nframe = 8192
noverhead = 64
nack = 64

R bps	3.00E+04	1.50E+06	4.50E+07	2.40E+09
tprop+tproc				
0.005	9.50E-01	3.50E-01	1.77E-02	3.39E-04
0.05	7.22E-01	5.14E-02	1.80E-03	3.39E-05
0.5	2.12E-01	5.39E-03	1.81E-04	3.39E-06

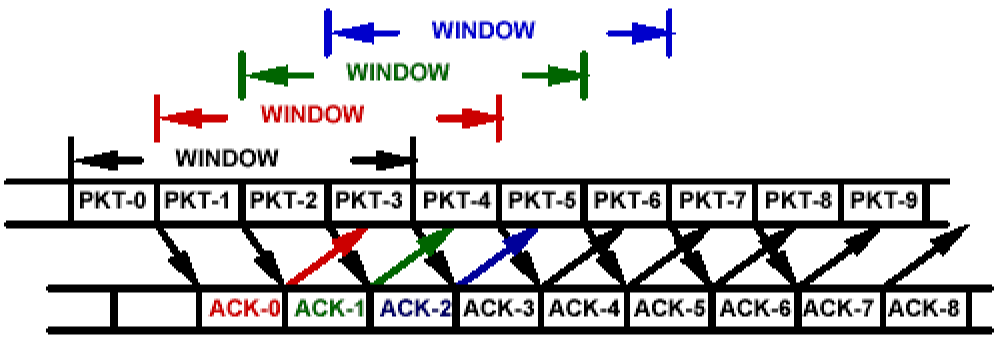
nframe= 524288

R bps	3.00E+04	1.50E+06	4.50E+07	2.40E+09
tprop+tproc				
0.005	9.99E-01	9.72E-01	5.38E-01	2.14E-02
0.05	9.94E-01	7.77E-01	1.04E-01	2.18E-03
0.5	9.46E-01	2.59E-01	1.15E-02	2.18E-04

Go-back N

- Stop and Wait is inefficient when propagation delay is larger than the packet transmission time
  - Can only send one packet per round-trip time
- Go Back N allows the transmission of new packets before earlier ones are acknowledged

- Go back N uses a window mechanism where the sender can send packets that are within a “window” (range) of packets
  - The window advances as acknowledgements for earlier packets are received



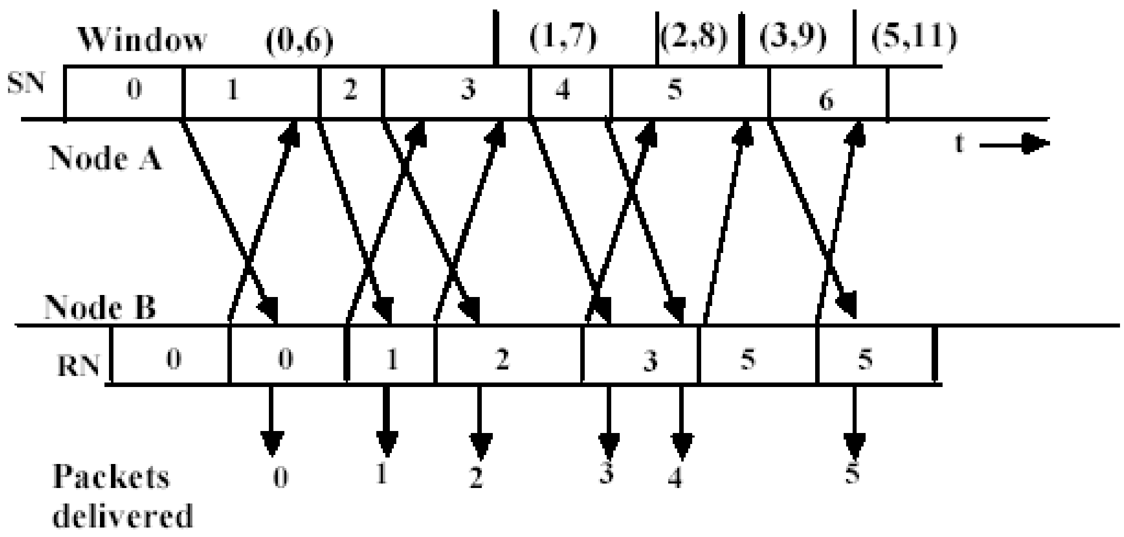
Features of Go-back N

- Window size = N
  - Sender cannot send packet i+N until it has received the ACK for packet i
- Receiver operates just like in Stop and Wait
  - Receive packets in order
  - Receiver cannot accept packet out of sequence
  - Send RN = i + 1 => ACK for all packets up to and including i
- Use of piggybacking
  - When traffic is bi-directional RN's are piggybacked on packets going in the other direction
    - Each packet contains a SN field indicating that packet's sequence number and a RN field acknowledging packets in the other direction

<--Frame Header --------->

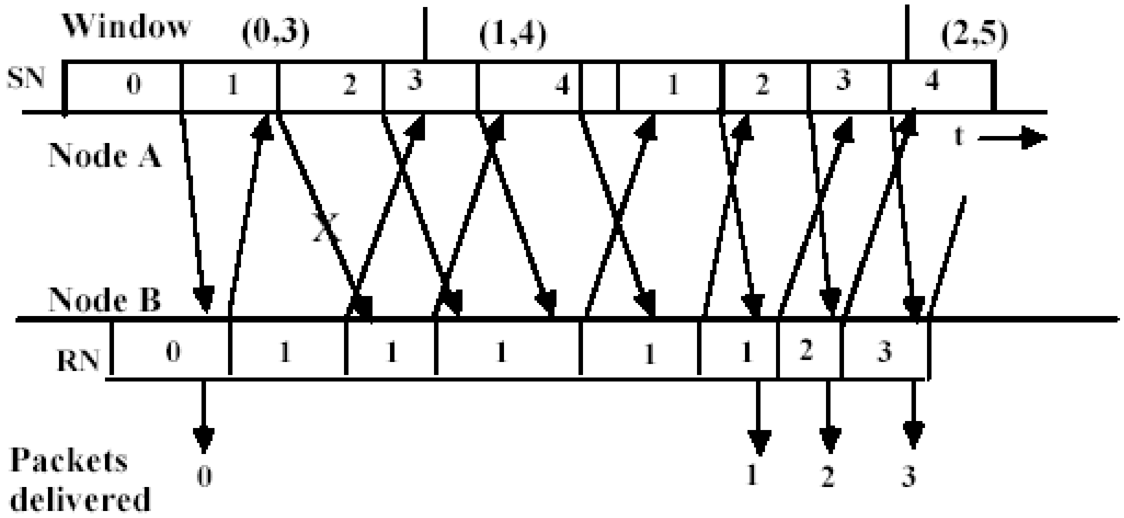
	SN	RN		Packet	CRC

Example Go-back 7



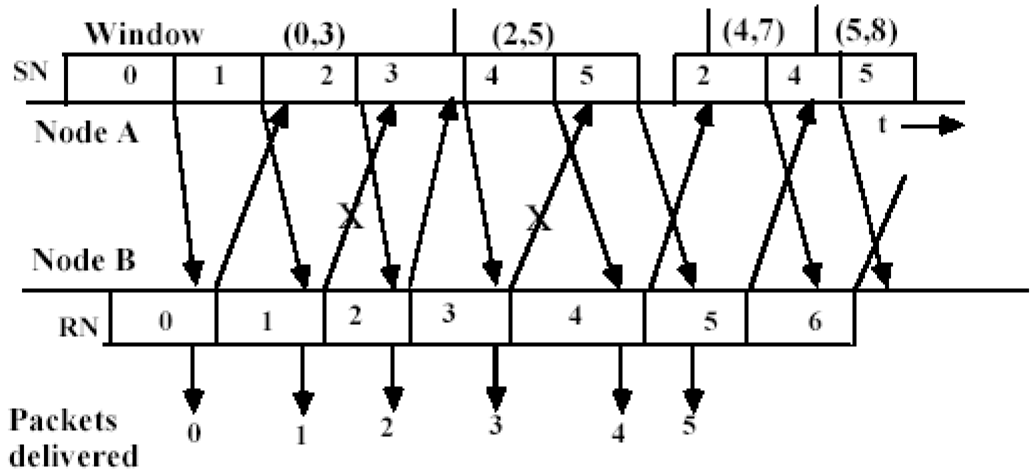
- Note that packet RN-1 must be accepted at B before a frame containing request RN can start transmission at B

Errors



- Note that the timeout value here is take to be the time to send a full window of packets
- Note that entire window has to be retransmitted after an error

Example



- When an error occurs in the reverse direction the ACK may still arrive in time. This is the case here where the packet from B to A with RN=2 arrives in time to prevent retransmission of packet 0
- Packet 2 is retransmitted because RN = 4 did not arrive in time, however it did arrive in time to prevent retransmission of packet 3
  - Was retransmission of packet 4 and 5 really necessary?

    Strictly no because the window allows transmission of packets 6 and 7 before further retransmissions. However, this is implementation dependent

Selective Repeat ARQ

- Selective Repeat attempts to retransmit only those packets that are actually lost (due to errors)
  - Receiver must be able to accept packets out of order
  - Since receiver must release packets to higher layer in order, the receiver must be able to buffer some packets

- Retransmission requests
  - Implicit
    - The receiver acknowledges every good packet, packets that are not ACKed before a time-out are assumed lost or in error
    - Notice that this approach must be used to be sure that every packet is eventually received
  - Explicit
    - An explicit NAK (selective reject) can request retransmission of just one packet
    - This approach can expedite the retransmission but is not strictly needed
  - One or both approaches are used in practice