Data Communication Networks
Data Link Layer

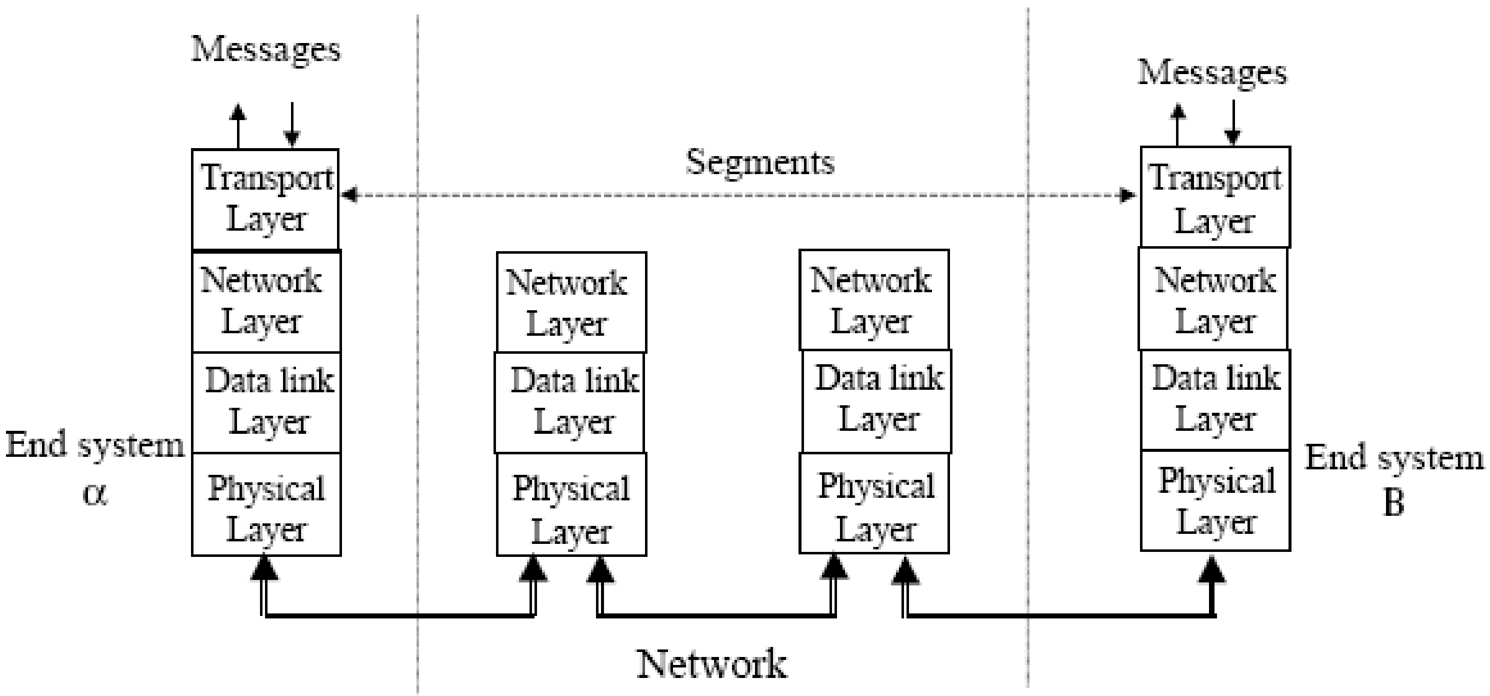
M. R. Pakravan
Department of Electrical Engineering
Sharif University of Technology

Data link layer

- Data Link Layer Purpose:
  - Ensure reliable, efficient communication between neighbor nodes
- Four specific functions:
  - Provide *services* to network layer
  - Framing
  - Error handling
  - Flow control
- Overview
  - Design issues
  - Error detection and correction
  - Elementary protocols
  - Sliding window protocols
  - Example protocols

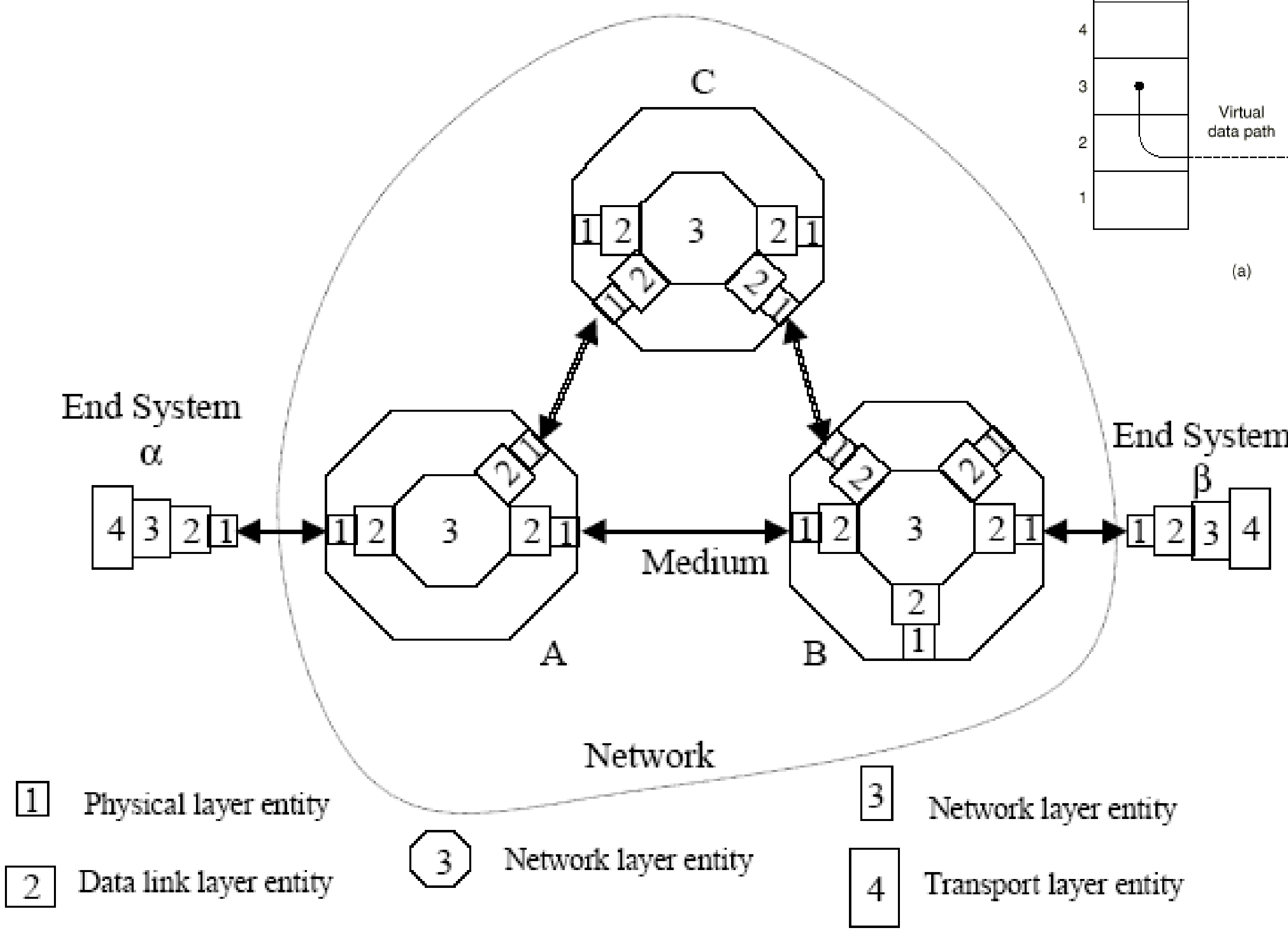
Data Link Layer

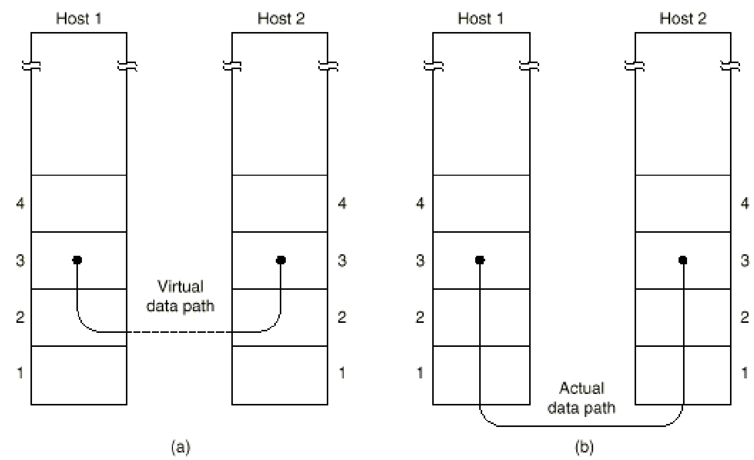
- Operation Perspective



Data Link Layer

- Data Link operation in a network





Services Provided to Network Layer

- Services provided to Network Layer:
  - Unacknowledged Connectionless
  - Acknowledged Connectionless
  - Acknowledged connection oriented

Services Provided to Network Layer

- Unacknowledged connectionless
  - Just send frames towards the destination
  - No connection is established, No connection released
  - No acknowledge of received frames
  - No attempt to recover lost frames
- Appropriate in case of:
  - Real-time traffic: speech, video: Short delay more important than 100% reliability
  - Low error channels: leave error correction to higher layers
  - Most LAN's are using this service class
- Example: Ethernet

Service to Network Layer

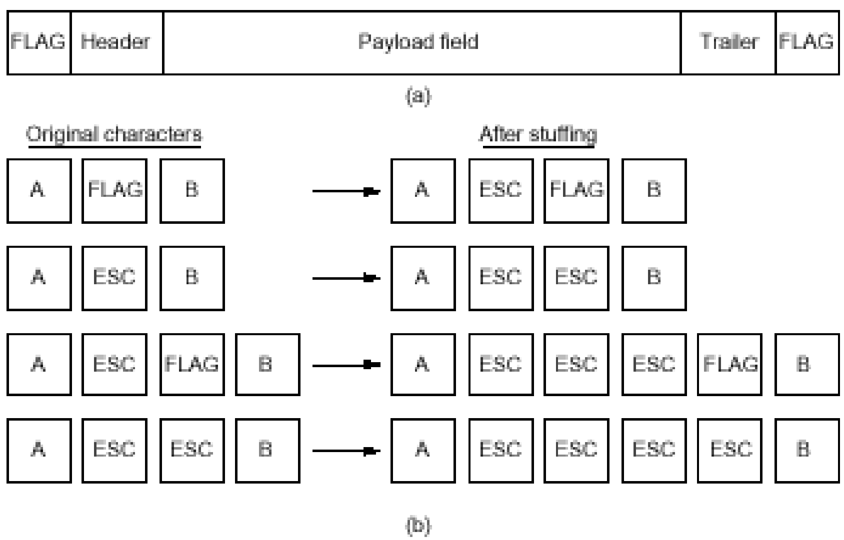
- Acknowledged connectionless
  - Each frame is acknowledged, but no connection is established
  - Acknowledgment is a service that can be performed by the transport layer as well
  - Data link layer provides this service to avoid long delays that could be caused if frames are not acknowledged
  - Specially important over un-reliable channels such as wireless
- Example: 802.11 (WiFi)

Service to Network Layer

- Acknowledged Connection Oriented
  - Establishes a connection before sending data
  - Each frame is numbered
  - Data link layer guarantees the delivery of a SINGLE copy of EVERY frame (With acknowledged connectionless, it is possible to receive multiple copies of a frame due to a lost ACK)
  - Releases the connection at the end of conversation (To release software and hardware resources tied up to the connection)
  - Provides a reliable bit stream to NL

Framing

- Character stuffing
  - Insert ASCII known characters to specify boundaries (FLAG)
  - Insert ESC before accidental FLAGs or ESCs of binary files. To be removed by the receiver's data link layer
  - Choice of marking characters (ESC, FLAG) Depends on 8 bit ASCII character set



Framing

- Bit Stuffing
  - Use a flag bit pattern (01111110)
  - Stuff a zero after 5 1's if it happened in user data
  - Easy to recover the frame if sync is lost

(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

Framing

- Coding Violation
  - Send physical layer signals that can not be associated with valid data sets to indicate frame boundaries
- Example: 4B/5B coding
  - Groups of 4 bits are mapped into group of 4 bits such that there is at least two transitions per block of bits
  - Special codes that are not part of the normal pattern (coding violations) can determine start and end of frames.
    - Start Delimeter: 11000 10001
    - End Delimeter: 01101 00111

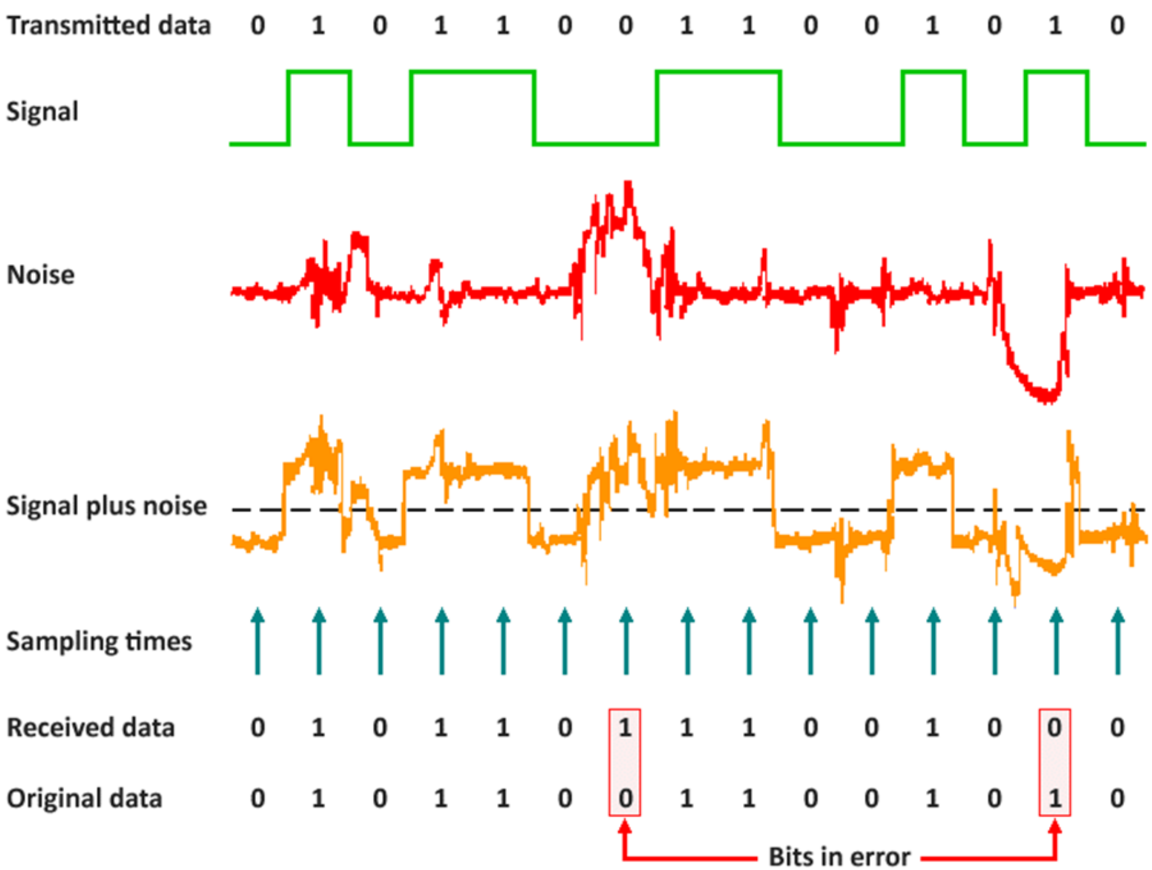
Data (Hex)	Data (Binary)	4B5B Code
0	0000	11110
1	0001	01001
2	0010	10100
3	0011	10101
4	0100	01010
5	0101	01011
6	0110	01110
7	0111	01111
8	1000	10010
9	1001	10011
A	1010	10110
B	1011	10111
C	1100	11010
D	1101	11011
E	1110	11100
F	1111	11101

Error Detection and Correction

- Error handling
  - Errors usually occur in bursts
  - Advantage: fewer frames are corrupted
  - Disadvantage: burst errors harder to detect and correct
- Two approaches:
  - Detection
  - Correction
- Both require extra bits per data word or message
- Idea: add r extra bits to frame data
- Not all 2^{m+r} codes are legal (only 2^m codes are legal)
- n = m + r bits code word, m bits for data and r bits added for error control

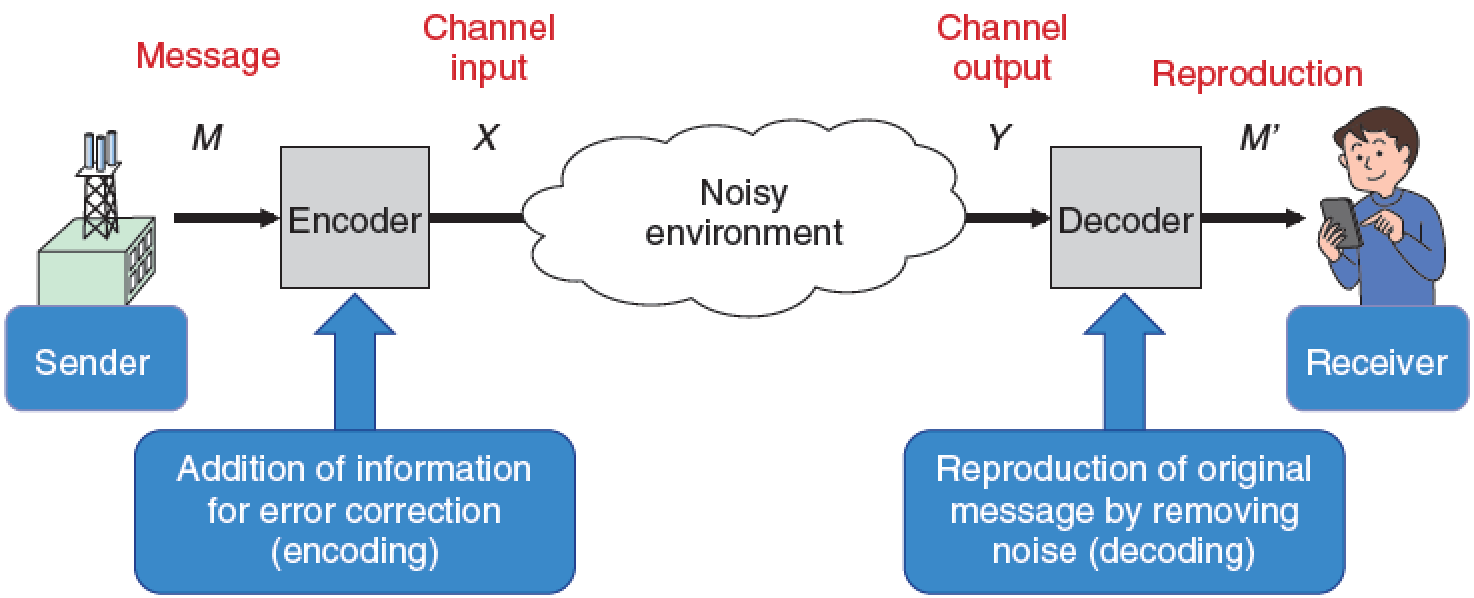
Error Control Coding

- Noise and other imperfections can cause errors in symbol detection



Error Control Coding

- Error control codes can be used to reduce the chance of error.



Decoding error probability = (probability of event $M \neq M$)

$$\text{Coding rate} = \frac{\text{(no. of message symbols)}}{\text{(no. of transmitted signals)}}$$

Error Control Coding

Error Detection

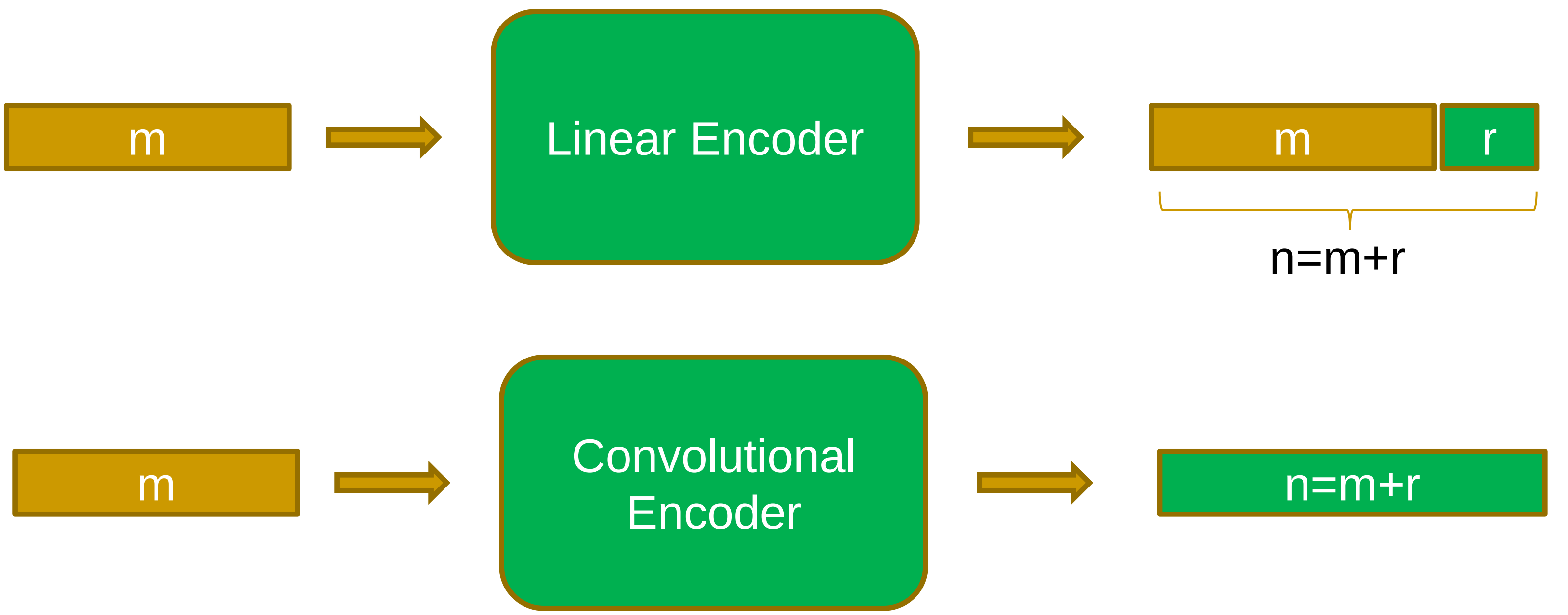
- Detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.

Error Correction

- Detection of errors and reconstruction of the original, error-free data

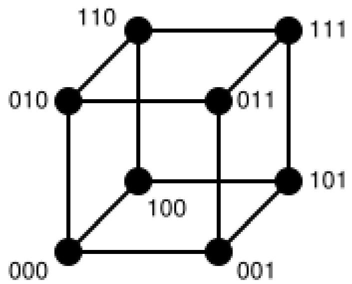
Error Control Coding

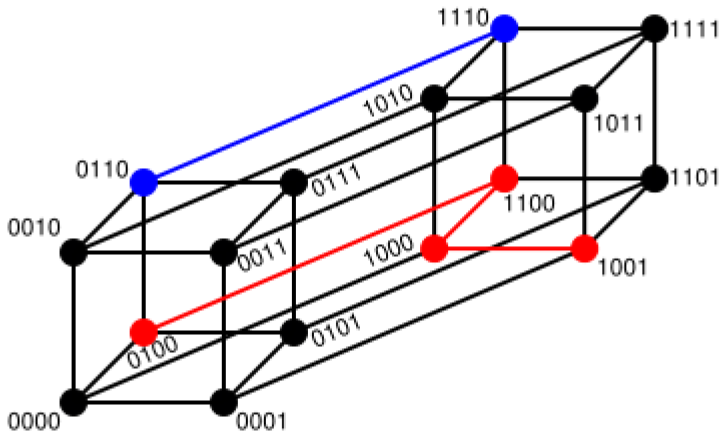
- Idea: Add controlled redundancy to the message



Hamming Distance

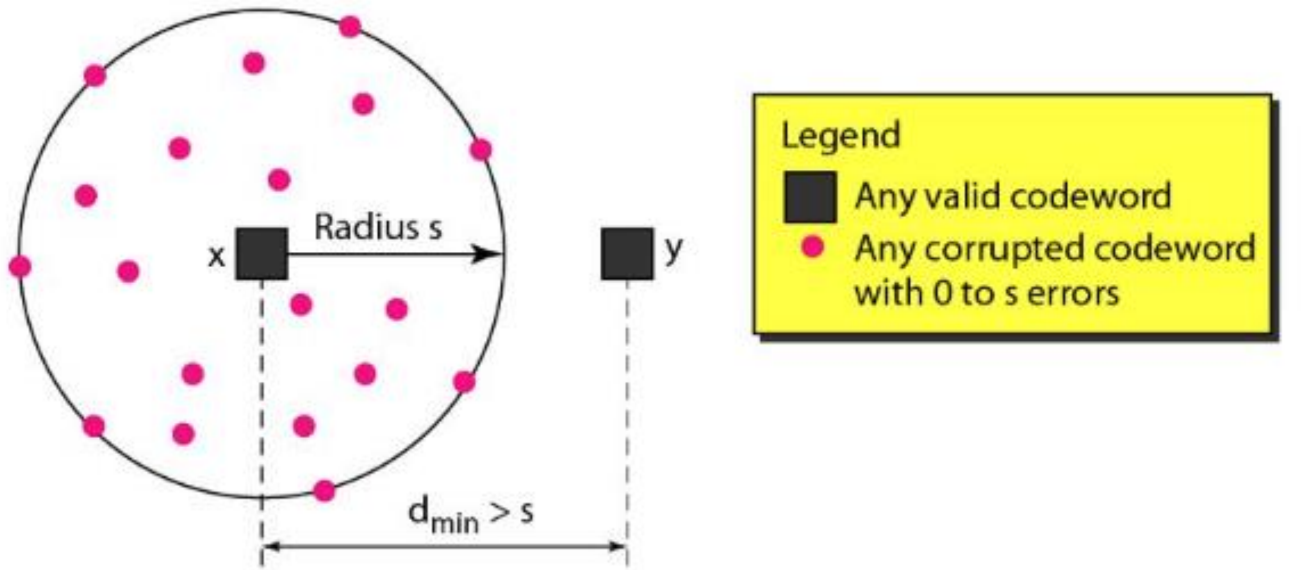
- The Hamming distance (HD) of 2 code words a and b is the number of bit positions in which a and b are different:
  HD (a,b)= Number of ones in (a XOR b)
- Note that HD (a,b) bit errors are required to convert a into b (or vice versa)





Error Detection Codes

- To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$



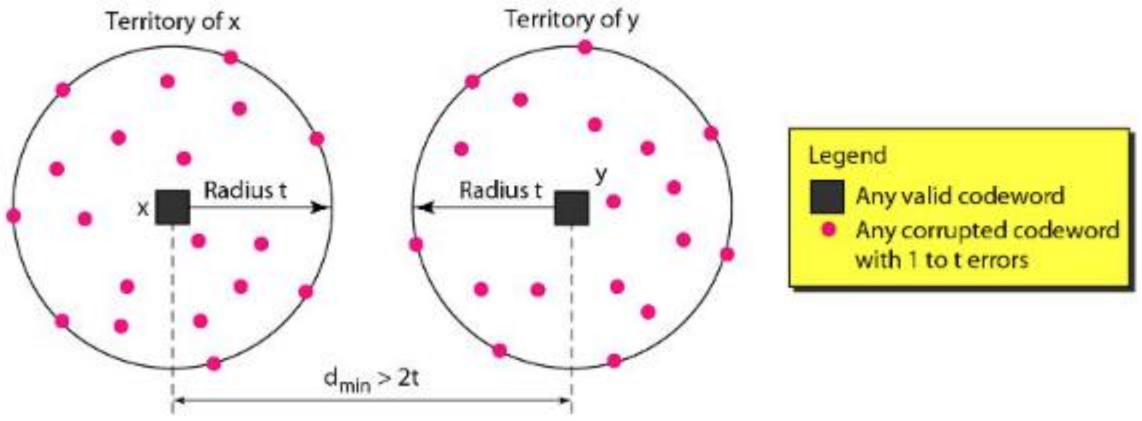
Error Control Coding

- Example 1: parity
  - Add (parity) bit to every data word such that total numbers of ones is even (or odd)
    - $HD_{min} = 2$
    - Single bit errors are detectable

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Error Correction Codes

- To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$



Error Control Coding

- Example 2:
  - 4 data words with $HD_{min} = 5$
  - If 00000.00111 arrives, what is the original?

Input	Encoded Output
00	00000.00000
01	00000.11111
10	11111.00000
11	11111.11111

Error Control Coding

- How many redundant (r) bits do we need for a single bit error correction?
  - Assume m bit message with r bits added for single error correction (n = m+r bits)
  - How many legal code-words? 2^m
  - How many illegal codes per code-words? $n*(2^m)$
  - $(n+1)*(2^m)<2^n$ => $(m+r+1)<2^r$
  - This is the minimum number of bits (r) we need to correct a single error in n bits with r parity bits
  - Example: M=8 => r>=4 ; M=16 => r>=5 ; M=240 => r>=8

Error Control Coding

Error Detection

- Encoder adds overhead for error detection
- Decoder detects error and requests re-transmission in case of error (Automatic Repeat request (ARQ))

Forward Error Correction (FEC)

- Encoder adds overhead to enable correction at the decoder
- Avoids delay of ARQ
- More overhead and increased complexity

Hybrid ARQ

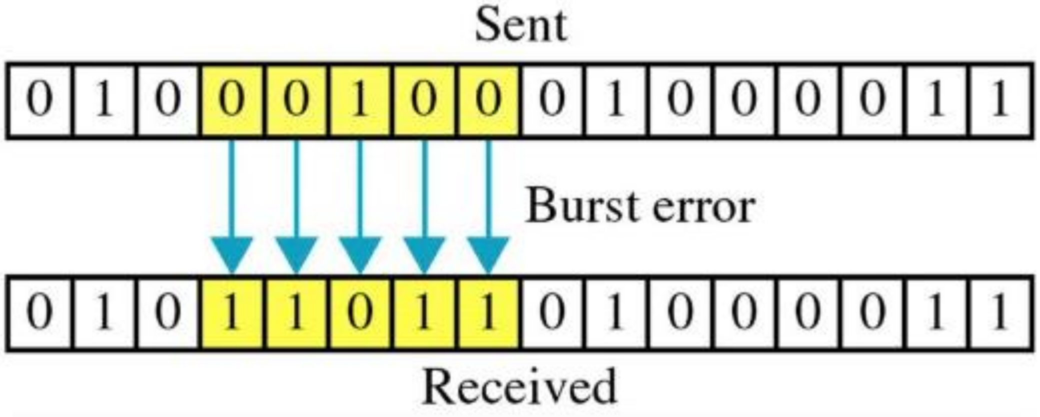
- Send messages with FEC and if decoding was not successful, request re-transmission
- Send messages with error detectioin codes and if there was an error, request FEC from source

Error Control Coding

- In some scenarios, error detection with retransmission is more efficient than error correction
- *Example:*
  - Error rate 10^{-6} ---- Block size 1000 bits
  - Single bit error correcting requires 10 bits / block
  - Single bit error detection requires 1 bit / block
  - Retransmission of 1 block (= 1001 bits) per 1000 blocks is more efficient

Burst Errors

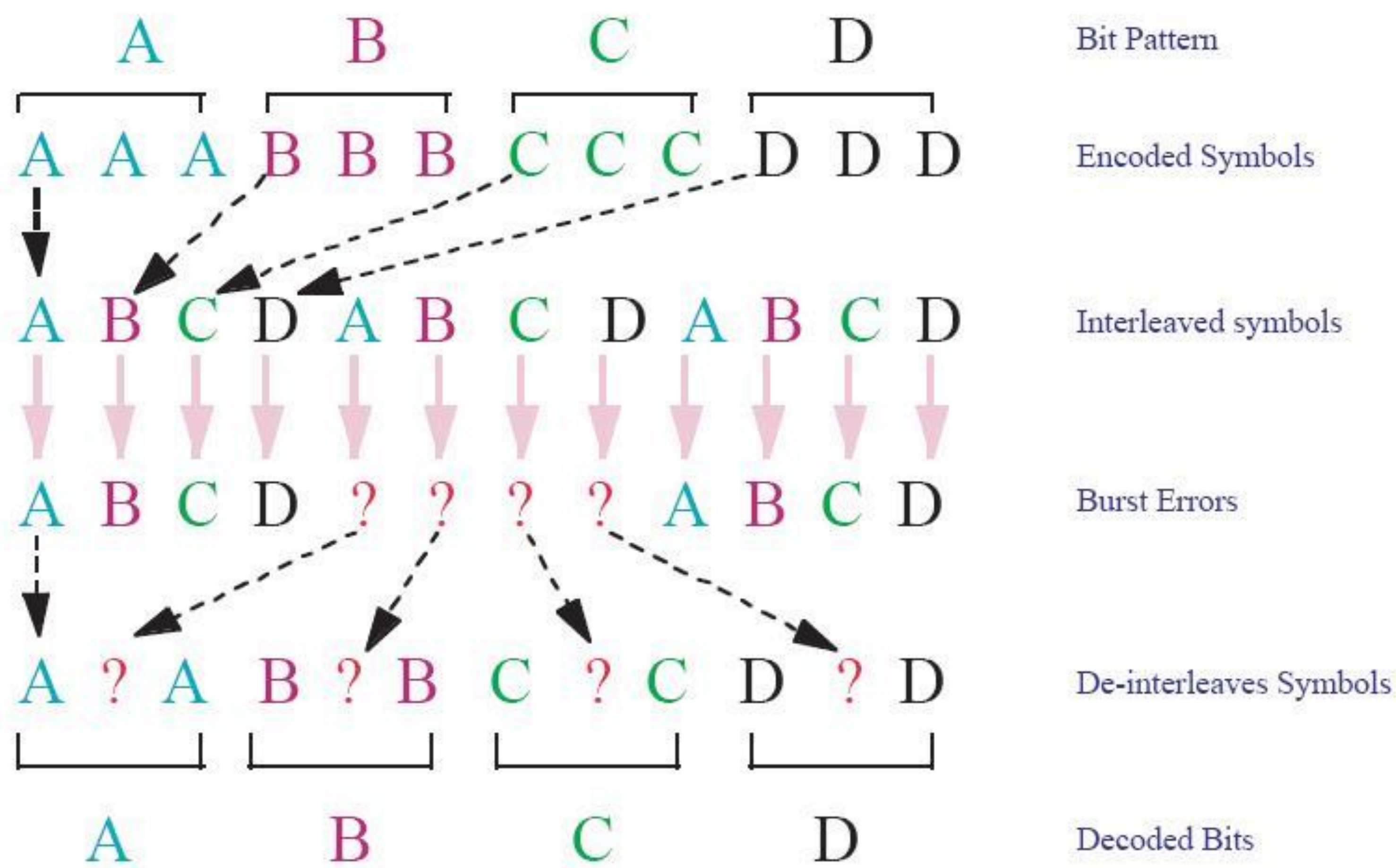
- In many scenarios, errors are seen as burst of errors
- FEC codes allow correction of a limited number of errors per block
- Burst errors may prevent decoders from recovering the correct message



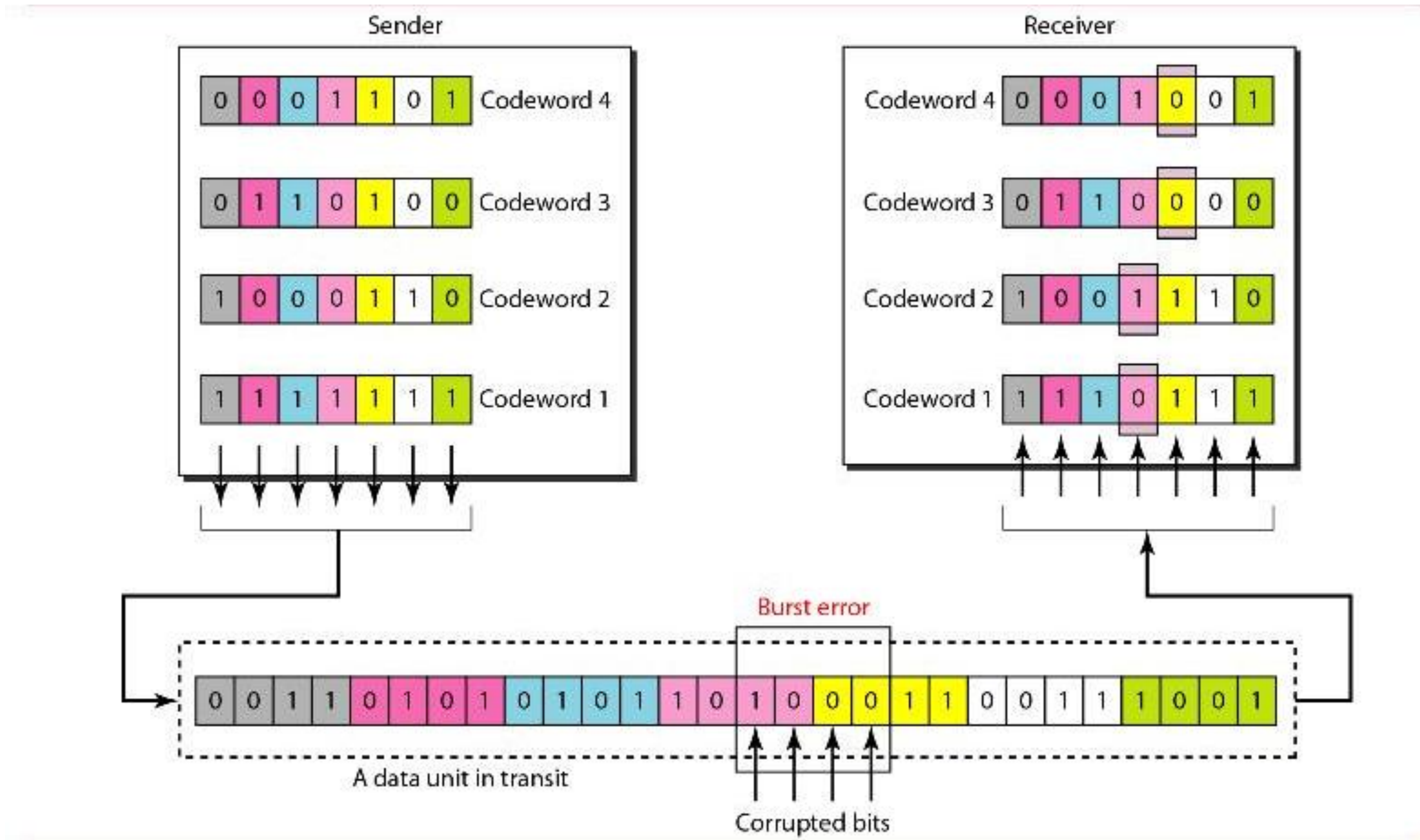
Interleaving

- We modify the order of symbol transmission at the source and re-order them at the receiver to allow proper operation of FEC

Interleaving

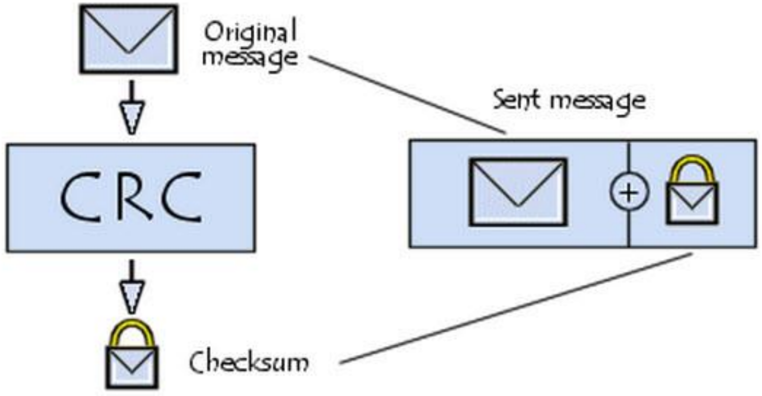


Interleaving



Cyclic Redundancy Check (CRC)

- Redundancy added to a block of bytes that uses cyclic codes to check and verify the block.

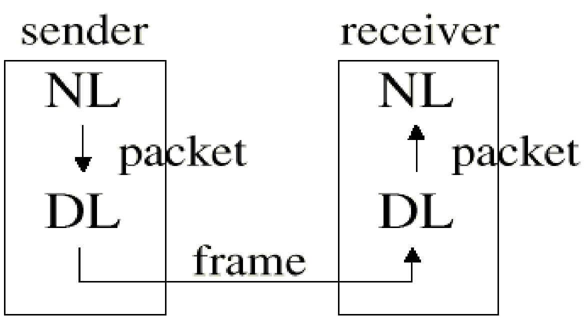


- Some standardized CRCs

Name	Length of CRC	Example Application
CRC-8	8	DVB-S2
CRC-16	16	Bluetooth
CRC-32	32	Ethernet

Elementary DL Protocols

- Assumptions:
  - Send a long data stream from sender to receiver
  - Checksum errors are signaled by the physical layer
  - NL, DL, PL are independent processes
  - Connection-oriented, reliable service needed at NL => one and only one copy of each frame and in the exact sequence should be received by NL



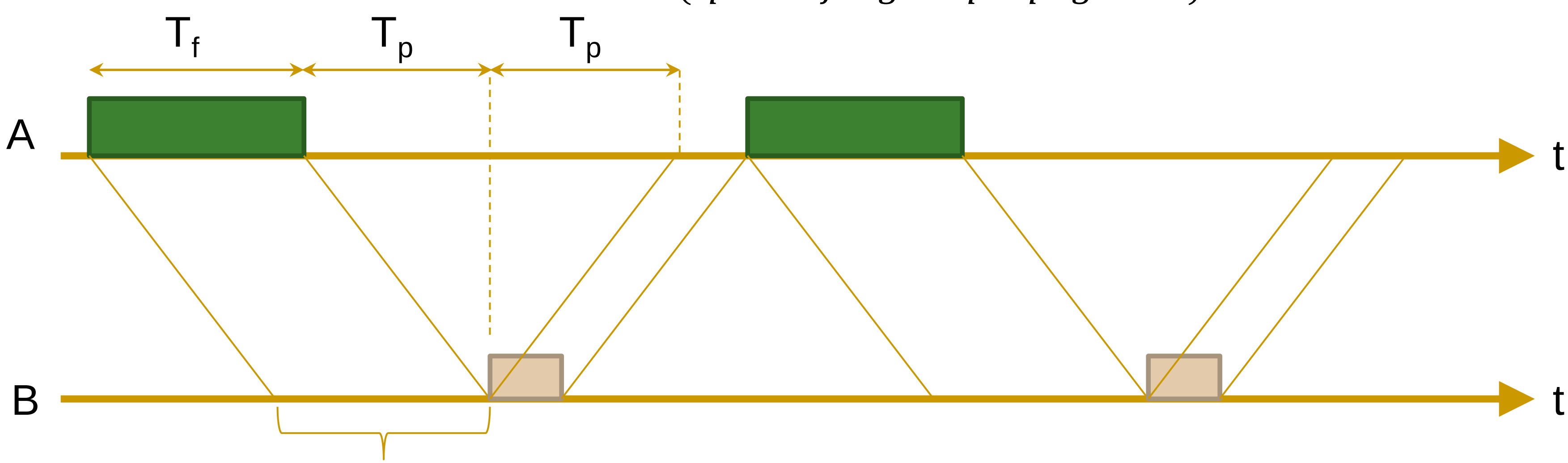
Frame = packet + header

Data Link: Flow Control

- Data Link layer protocol runs on two different systems with possibly different hardware/software characteristics.
- If the sender sends its frame at rate R_a and the receiver can process only at rate R_b where $R_a > R_b$, then the receiver will be overloaded and unable to receive all frames correctly.
- Therefore, we need to have a flow control mechanism in data link layer
- The solution: Use Acknowledgment

We Need "ACK"

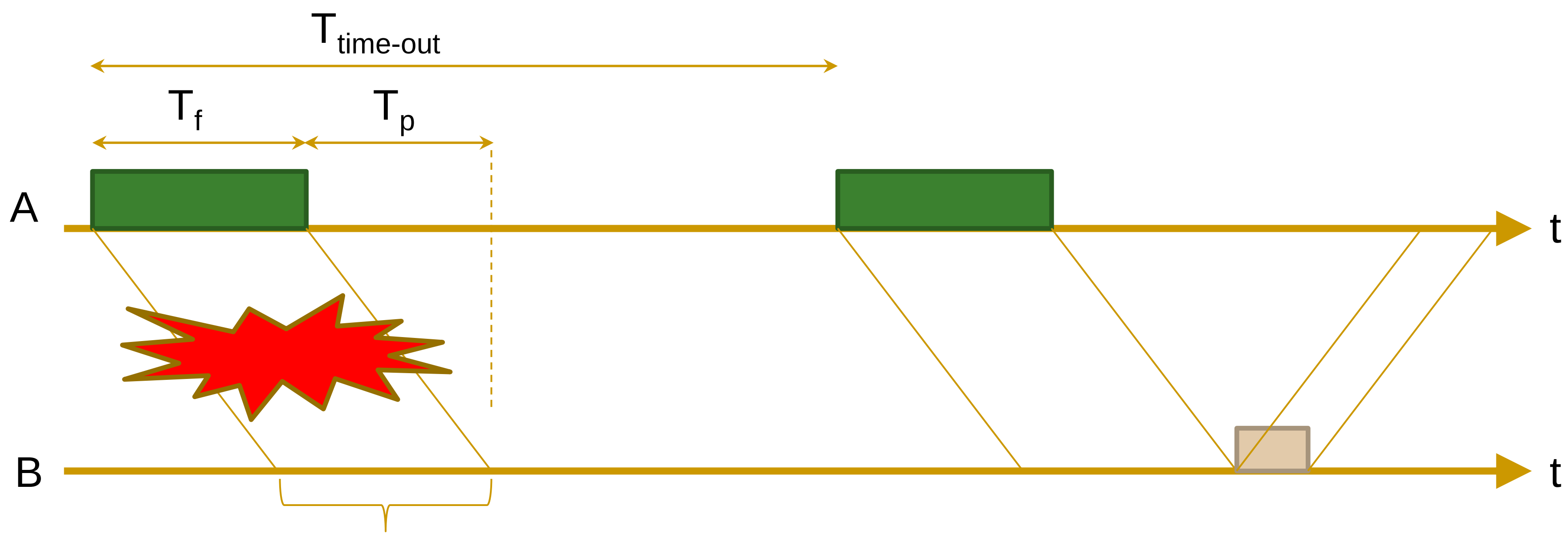
- Node A waits for an Ack frame from Node B before sending its next frame
- Frame Length in Time: $T_f = \dfrac{N_f\ (frame\ length\ in\ bits)}{R_A\ (Transmit\ Rate\ of\ A)}$
- Propagation Delay: $T_p = \dfrac{L\ (Distance\ between\ two\ nodes)}{V\ (speed\ of\ signal\ propagation)}$



B receives the frame and then generates an Ack

We Need "Timers"

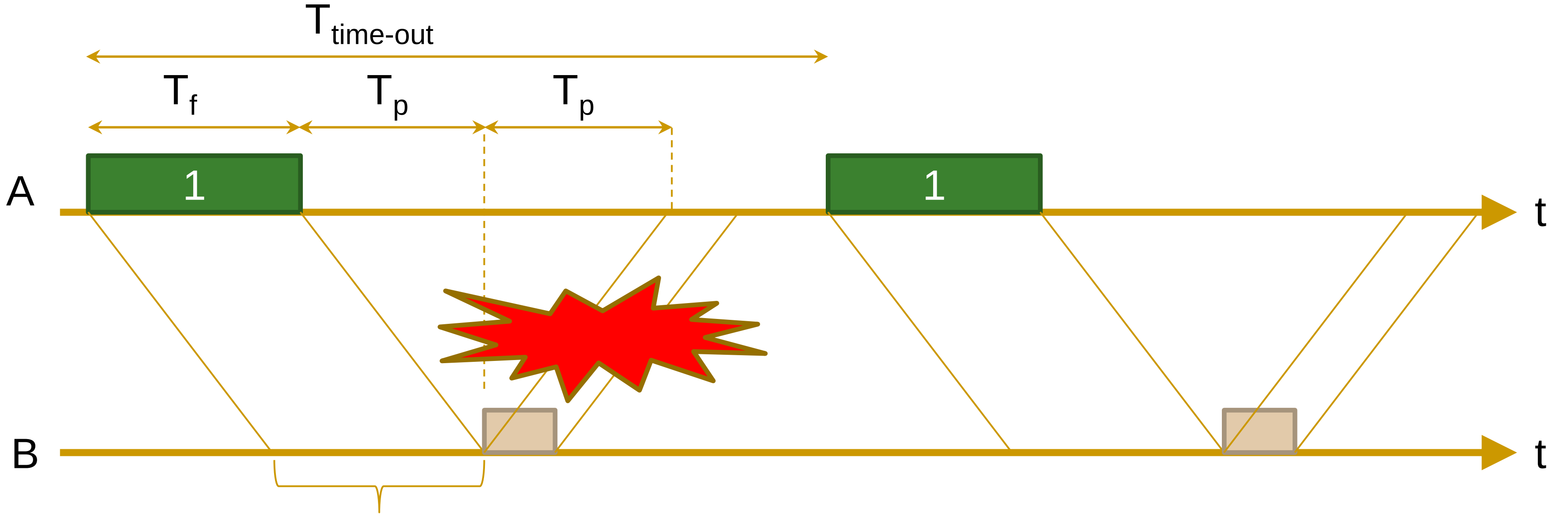
- What if a frame is not received correctly?
  - A could wait for the Ack that B has not transmitted (Mutual Locking)
  - Solution: A should have a timer to re-send to lost frame after some time



B: Frame Error or no Frames Received

We Need “Frame Number”

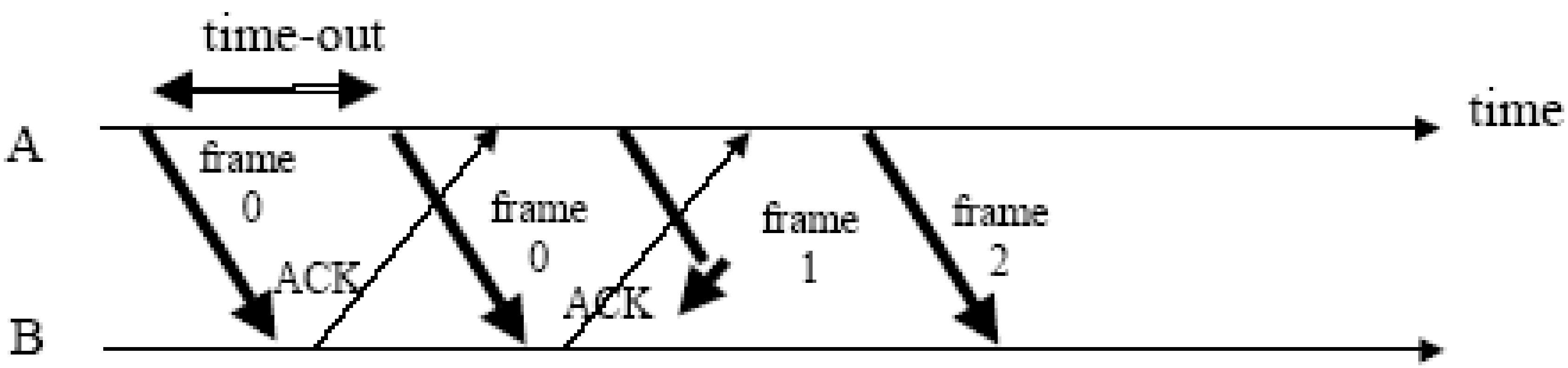
- What if the Ack is lost?
  - A sends the same frame again. Therefore, data link layer in B has two of the same frame. (Frame duplication)
  - Solution: We need to use frame numbers to avoid duplication



B receives the frame correctly, but the Ack is lost

We Need "Ack Number"

- What if the Timers are not set properly?
  - A receives an Ack after frame#1, so it thinks that frame #1 is delivered correctly to B, but this may not be true!
  - Solution: We should use Ack Numbers so A knows which frame was Acked.



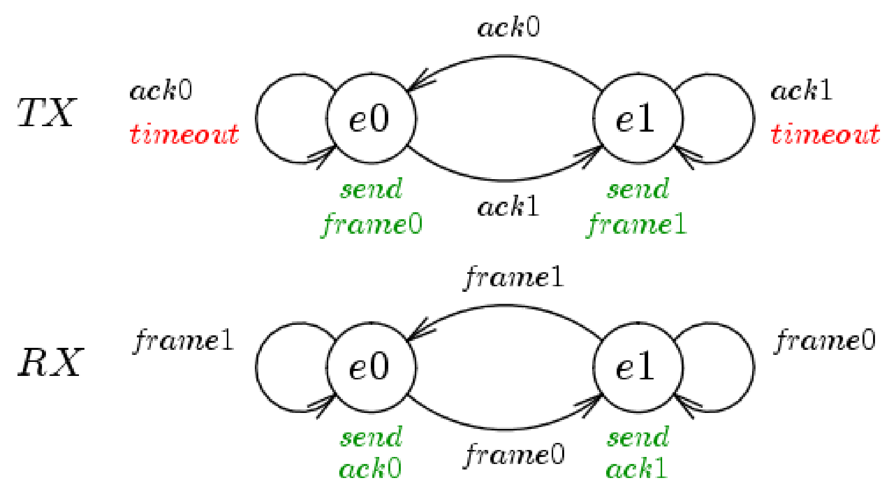
Transmitting station A misinterprets duplicate ACKs

Stop and Wait Protocol

- A (TX) sends its numbered frames, starts a timer for each frame and waits for an Ack for that exact same frame
- B (RX) checks the received frame and sends a Numbered Ack to confirm.
- A (TX) Stops and Waits for the correct Ack before it continues to the next frame

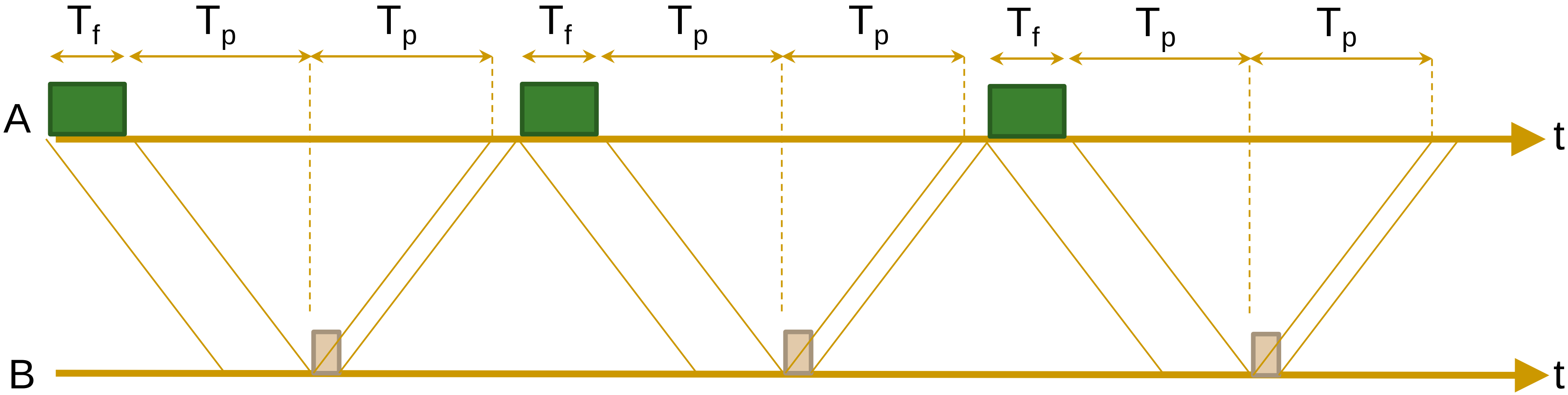
Seq#	Ack#	Packet	CRC

Frame



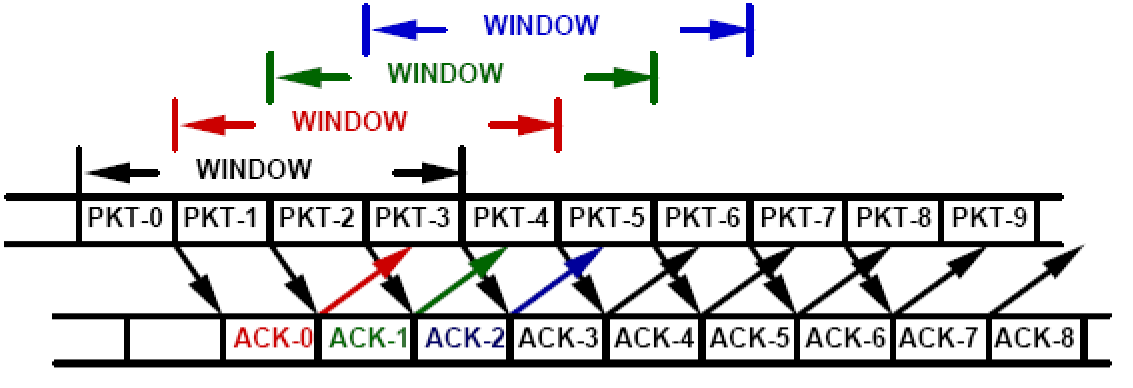
Stop and Wait

- If $a = T_p/T_f$ is not very small, then stop and wait is inefficient.



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 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



Go Back N

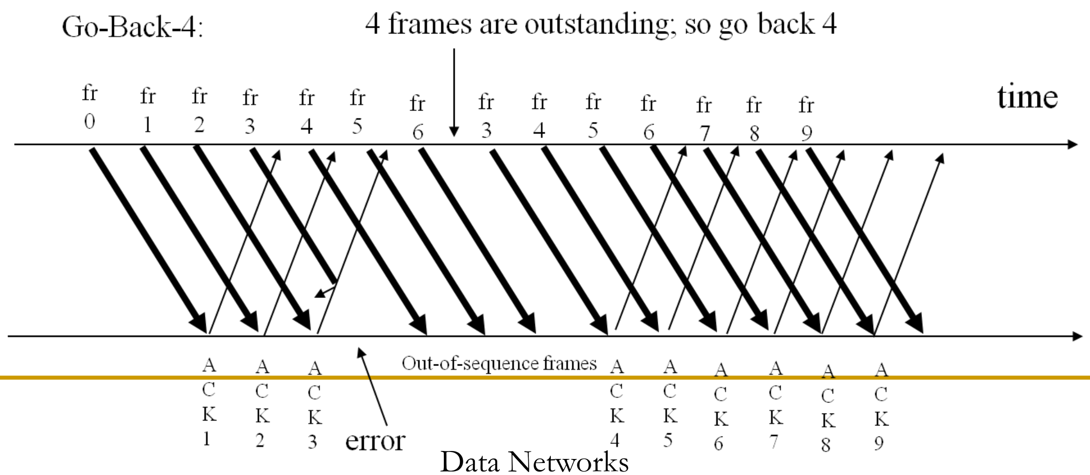
- Window size = N
- Sender cannot send packet i+N until it has received the ACK for packet i
- Receiver operates just like in Stop and Wait
    - Receive packets in order
    - Receiver cannot accept packet out of sequence
    - Send RN = i + 1 => ACK for all packets up to and including i
- Use of piggybacking
    - 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

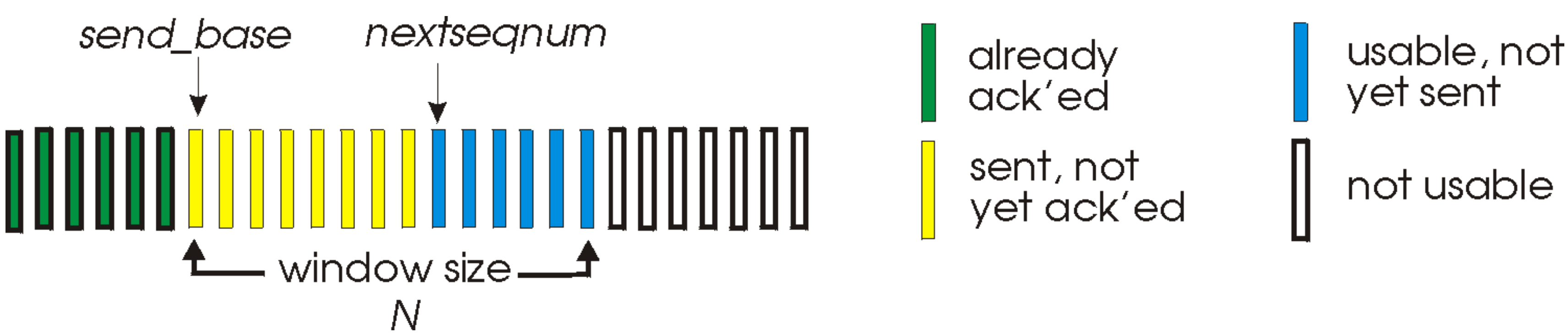
Go Back N

- The transmitter has a "window" of N packets that can be sent without acknowledgements
- This window ranges from the last value of RN obtained from the receiver (denoted SNmin) to SNmin+N-1
- When the transmitter reaches the end of its window, or times out, it goes back and retransmits packet SNmin
- Let SNmin be the smallest number packet not yet ACKed
- Let SNmax be the number of the next packet to be accepted from the higher layer (I.e., the next new packet to be transmitted)



Go Back N

- Sender:
  - k-bit seq # in frame header
  - “window” of up to N, consecutive unACK’ed frames allowed
  - ACK(n): ACKs all frames up to, including seq # n - “cumulative ACK”
  - may receive duplicate ACKs (see receiver)
  - timer for each in-flight frames
  - timeout(n): retransmit frames n and all higher seq # frames in window

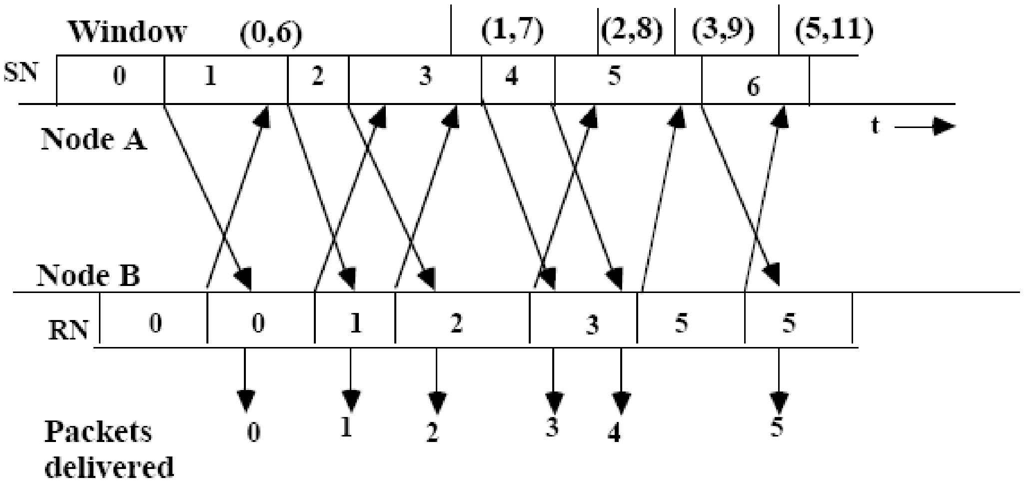


Go Back N – Sender Algorithm

- SNmin = 0; SNmax = 0
- Repeat
  - If SNmax < SNmin + N (entire window not yet sent)
    - Send packet SNmax ;
    - SNmax = SNmax + 1;
  - If packet arrives from receiver with RN > SNmin
    - SNmin = RN;
  - If SNmin < SNmax (there are still some unacknowledged packets) and sender cannot send any new packets
    - Choose some packet between SNmin and SNmax and re-send it or do not send anything
- The last rule says that when you cannot send any new packets you should re-send an old (not yet ACKed) packet
- There may be two reasons for not being able to send a new packet
  - Nothing new from higher layer
  - Window expired (SNmax = SNmin + N)
- No set rule on which packet to re-send
  - Least recently sent

Go Back N – Receiver Algorithm

- RN = 0;
- Repeat
  - When a good packet arrives,
    - if SN = RN
      - Accept packet
      - Increment RN = RN +1
- At regular intervals send an ACK packet with RN
  - Most implementations send an ACK whenever they receive a packet from the other direction
  - ACK can be delayed for piggybacking
- Receiver reject all packets with SN not equal RN
  - However, those packets may still contain useful RN numbers
- Note that packet RN-1 must be accepted at B before a frame containing request RN can start transmission at B

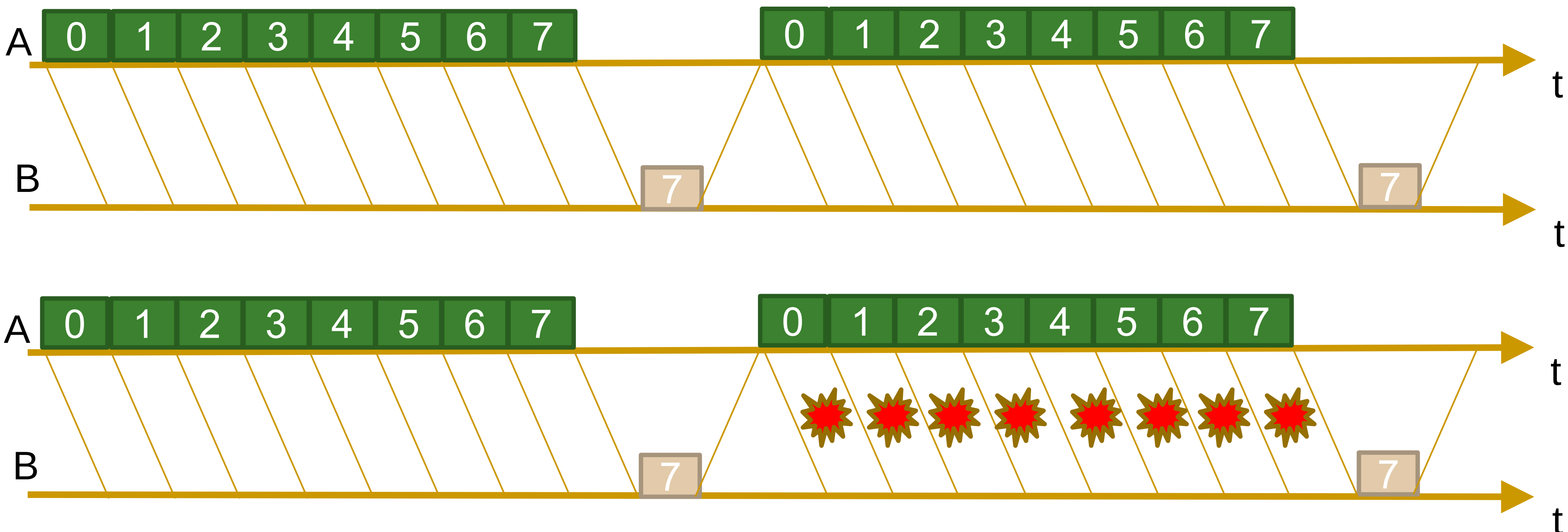


Notes of Go Back N

- Requires no buffering of packets at the receiver
- Sender must buffer up to N packets while waiting for their ACK
- Sender must re-send entire window in the event of an error
- Receiver can only accept packets in order
  - Receiver must deliver packets in order to higher layer
  - Cannot accept packet i+1 before packet i
  - This removes the need for buffering
  - This introduces the need to re-send the entire window upon error
- The major problem with Go Back N is this need to re-send the entire window when an error occurs.
- This is due to the fact that the receiver can only accept packets in order

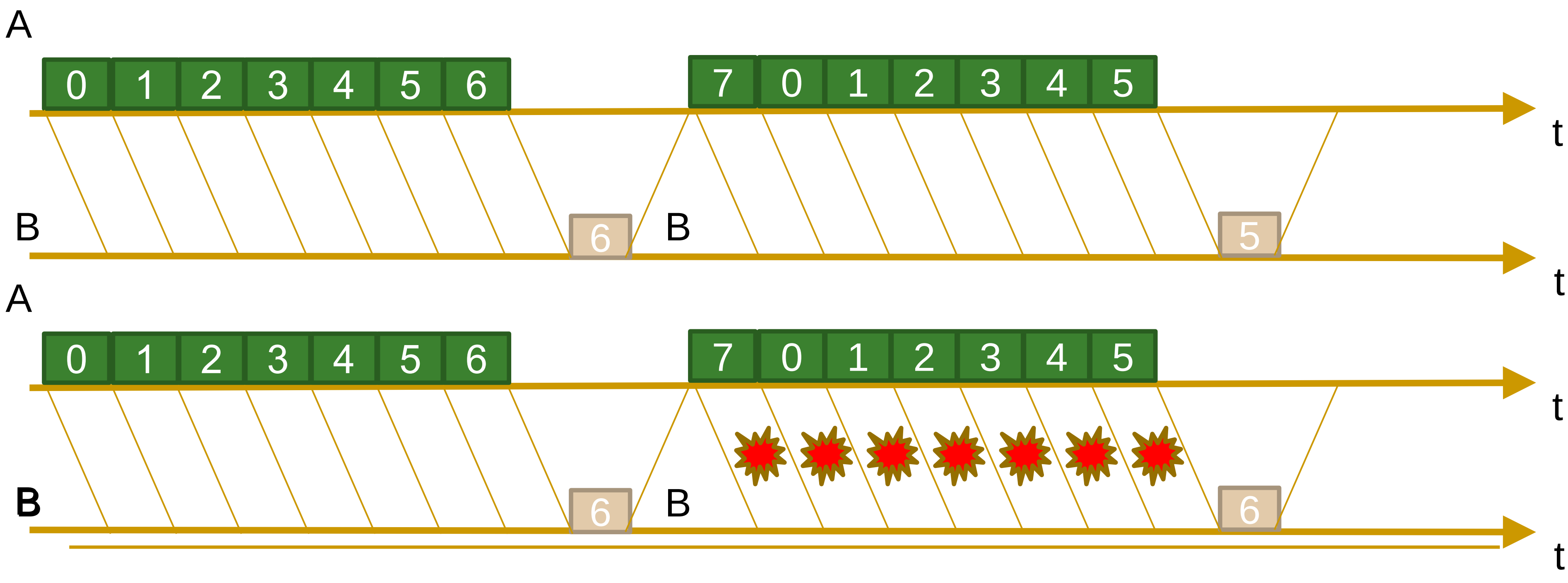
Go Back N Counters

- Window Size: N
- Sequence Counter: M different numbers (For example, If the counter has 3 bits, $M=2^3=8$)



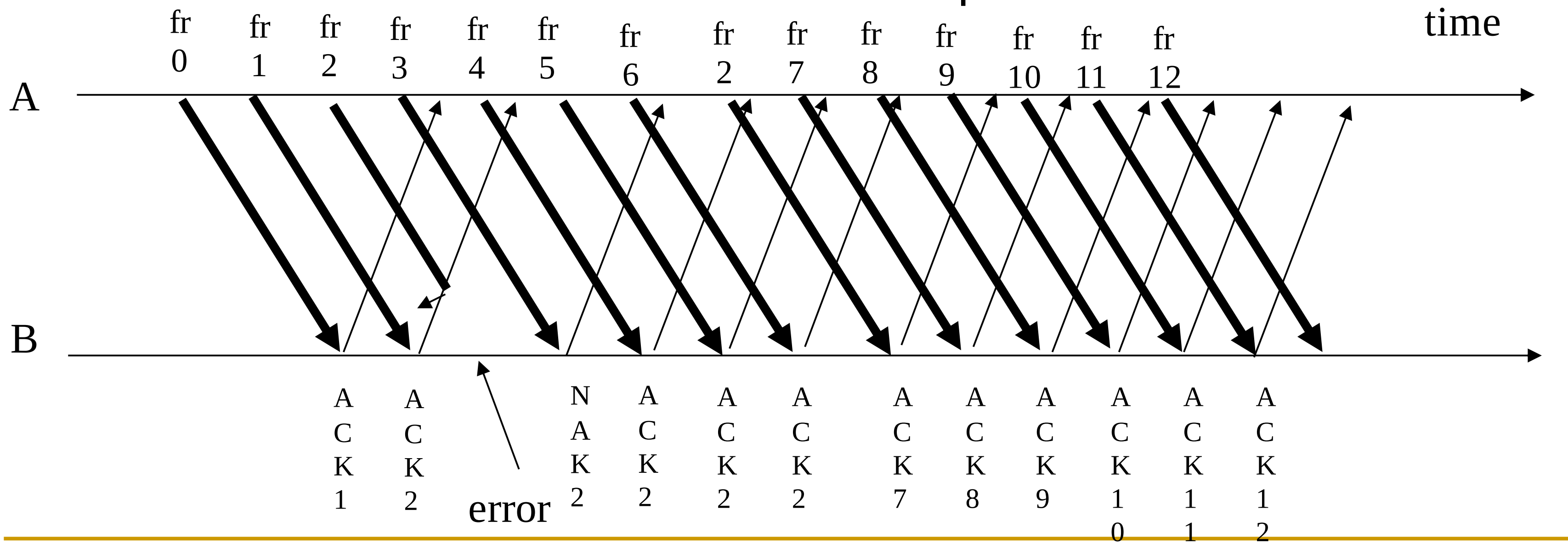
Go Back N Counters

- Window Size: N
- Sequence Counter: M different numbers (For example, If the counter has 3 bits, $M=2^3=8$)
- We should always choose N<M



Selective Repeat

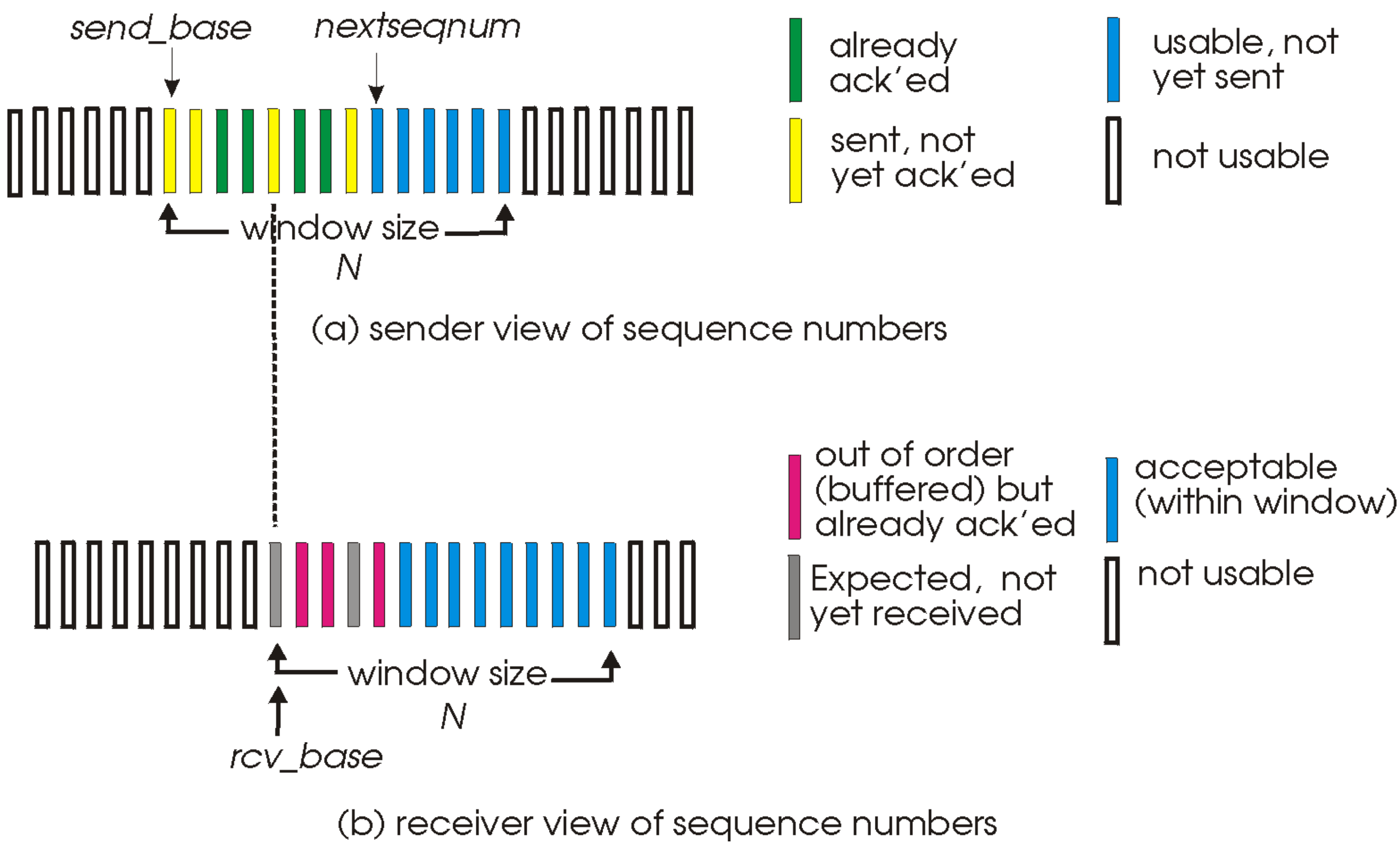
- Selective Repeat attempts to retransmit only those packets that are actually lost (due to errors)
  - Receiver must be able to accept packets out of order
  - Since receiver must release packets to higher layer in order, the receiver must be able to buffer some packets



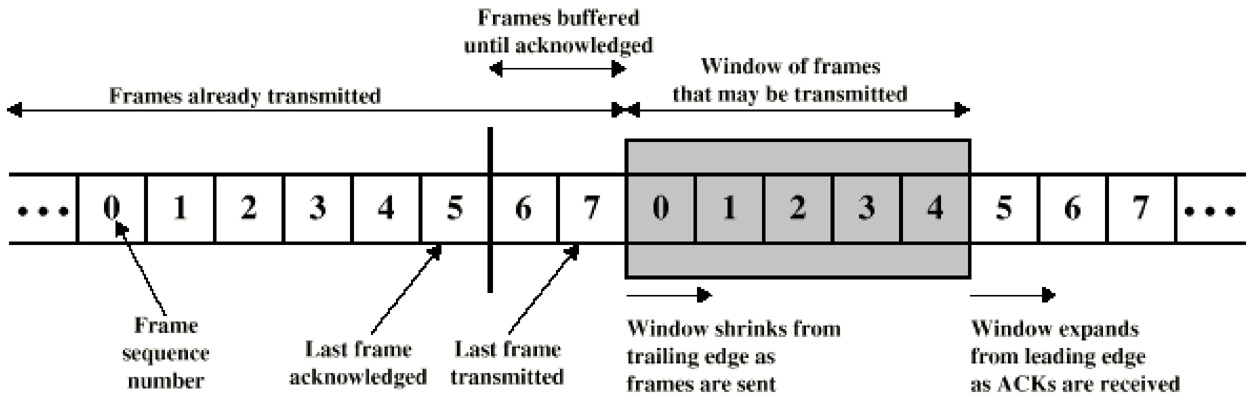
Selective Repeat

- 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

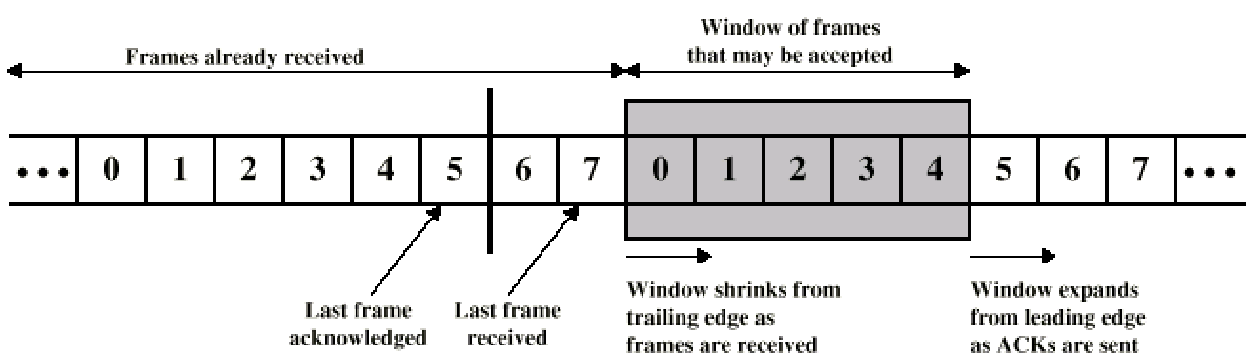
Selective Repeat



Selective Repeat

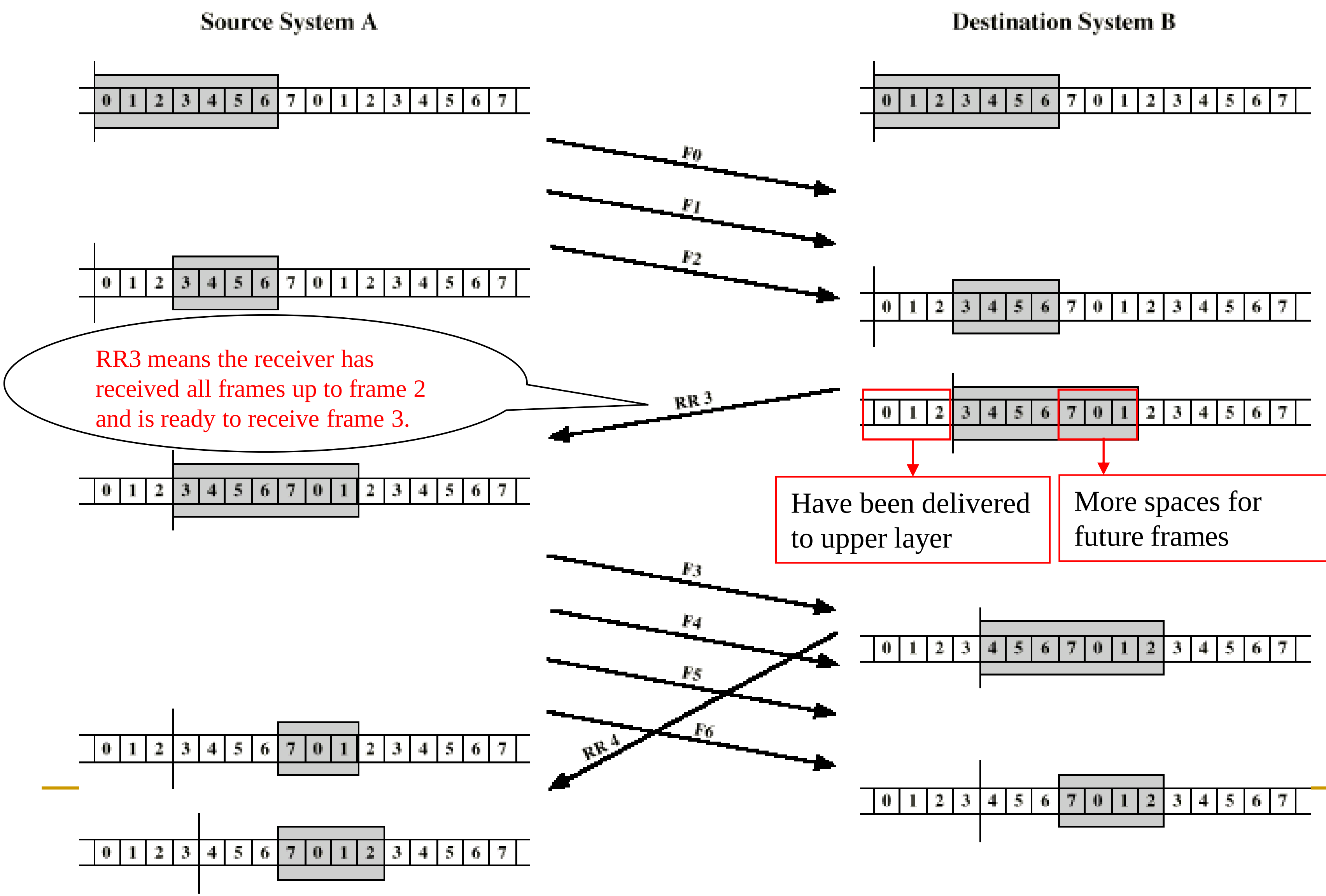


(a) Sender's perspective



(b) Receiver's perspective

Selective Repeat Example

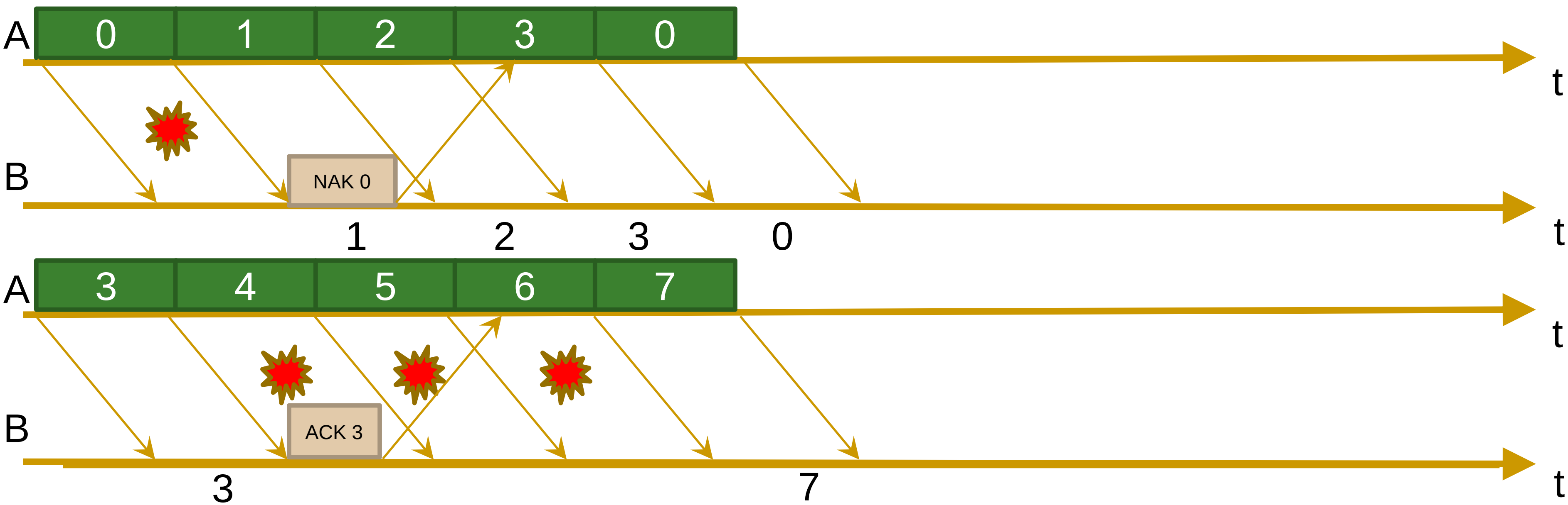


Selective Repeat Algorithm

- Sender:
  - Can transmit new packets as long as their number is within W (Window Size) of all un-ACKed packets
  - Retransmits un-ACKed packets after a timeout (Or upon a NAK if NAK is employed)
  - Must buffer all packets until they are ACKed (Up to W un-ACKed packets are possible)
- Receiver:
  - ACKs all correct packets
  - Stores correct packets until they can be delivered in order to the higher layer
  - Must buffer packets until they can be delivered in order i.e., until all lower numbered packets have been received. This is needed for orderly delivery of packets to the higher layer
  - Up to W packets may have to be buffered (in case the first packet of a window is lost)
- Implication of window size = N
  - Number of un-ACKed packets at sender =< N (window limit at sender)
  - Number of un-ACKed packets at sender cannot differ by more than N (window limit at the receiver required to deliver packets in order)
  - Packets must be numbered modulo M >= 2N (using $\log_2(M)$ bits)

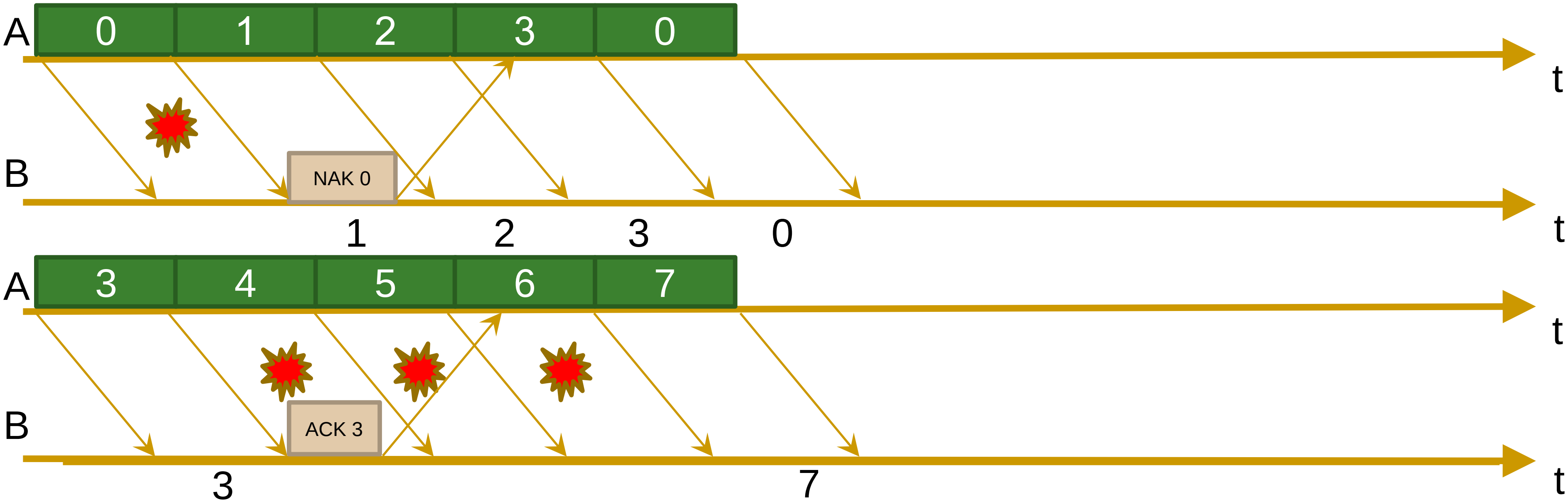
Selective Repeat Counters

- Example: Window Size N=4
- Case 1) Frame#0 has error, therefore B send NAK#0 and A sends frame#0 after frame#3
- Case 2) A sends frame#3, frame#4, frame#5, frame#6 and frame#7. All frames between frame#3 and frame#7 are lost. Therefore, B receives frame#3 followed by frame#7
- We should have a numbering system that allows differentiation between these two cases

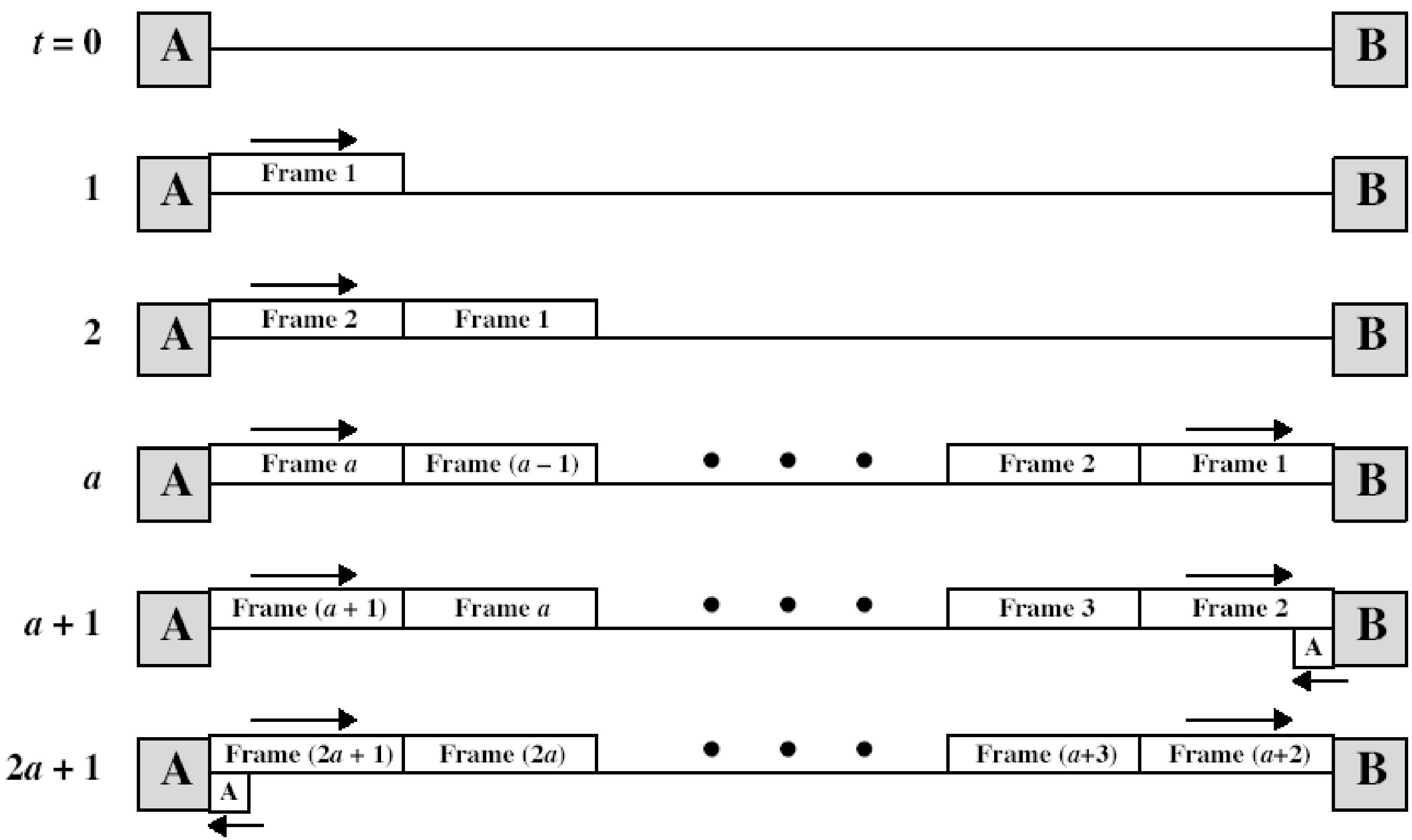


Selective Repeat Counters

- In general, the range of packets that may follow packet i at the receiver (Ex: N=4, Set i=3)
- Packet i may be followed by the first packet of the window (i -N+1) if it requires retransmission
- Packet i may be followed by the last packet of the window (i+N) if all Of the frames between i and i +N are lost
- Receiver must differentiate between packets i -N+1 ... i +N
- These 2N packets can be differentiated using Mod 2N numbering
- In Selective Repeat, we should choose M≥2N

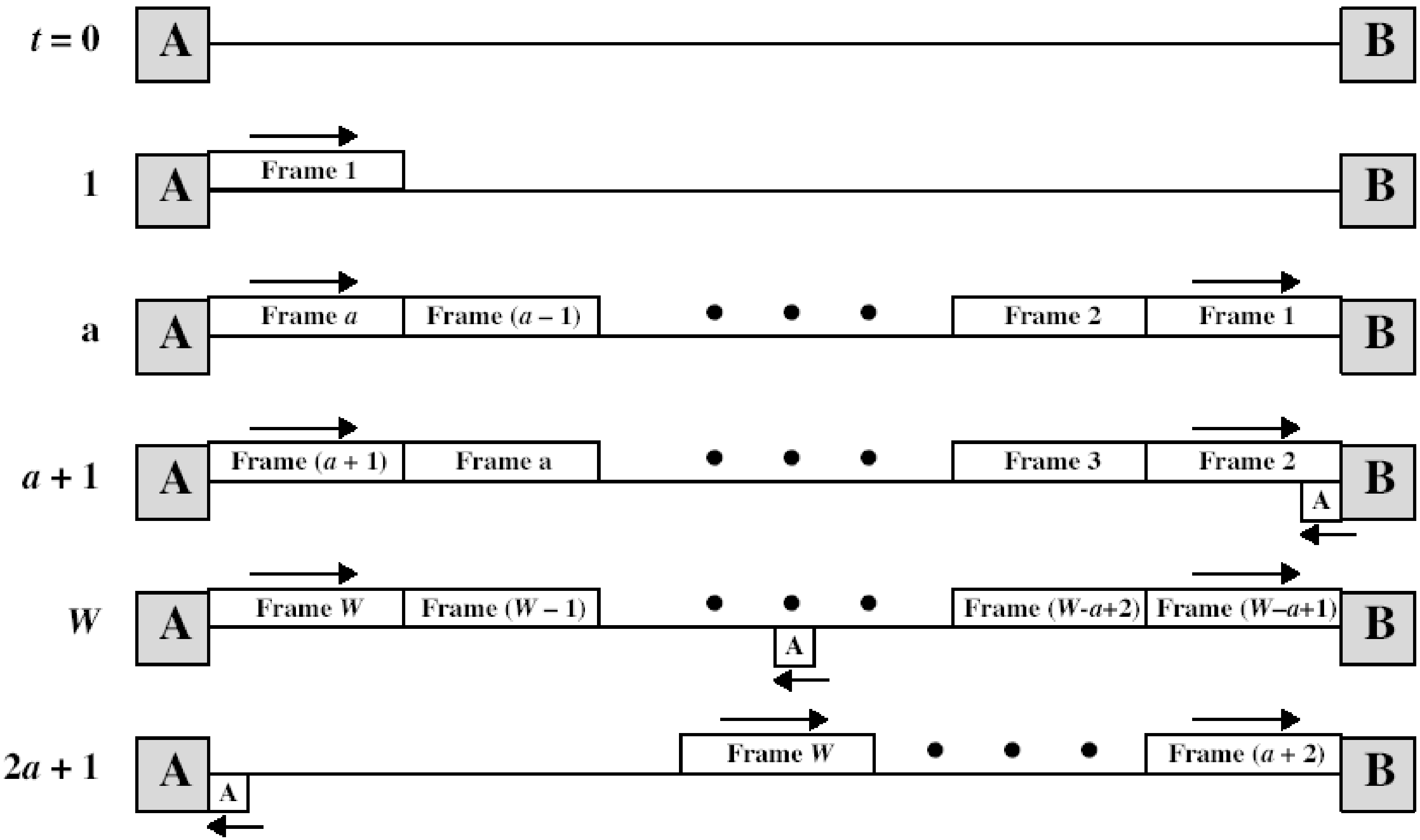


Performance Analysis of L2 Protocols



(a) W $2a + 1$

Performance Analysis of L2 Protocols

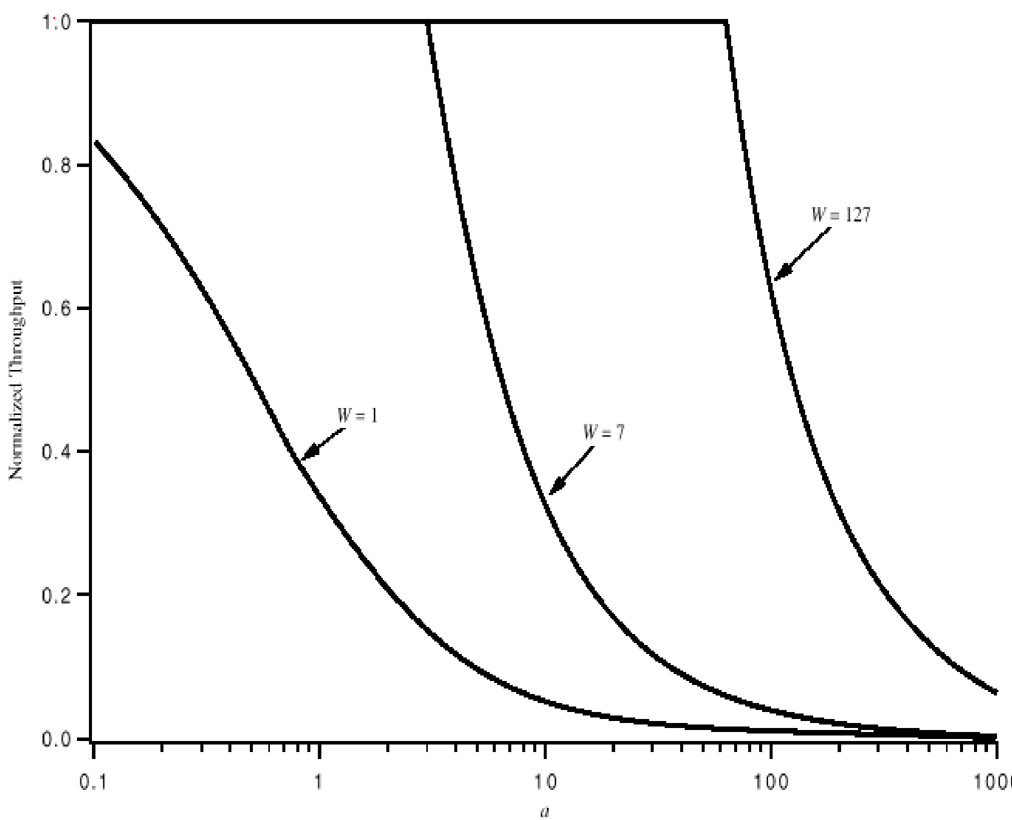


(b) $W < 2a + 1$

Link Utilization

- Error Free Operation
  - Stop and Wait: $U = \dfrac{1}{2a+1}$ $\quad a = \dfrac{T_{prop}}{T_{frame}}$
  - Sliding Window (W=window Size=2^n-1):

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



Link Utilization

- Probability of frame error = p
- Stop and Wait Utilization
  - Probability of a frame requiring exactly k transmission = $p^{k-1}(1-p)$
  - Expected Number of Transmission for a frame (N_r):

$$E[N] = N_r = \sum_{k=1}^{\infty} k \times \mathrm{Prob}(k\ Transmission) = \sum_{k=1}^{\infty} kp^{k-1}(1-p) = \frac{1}{1-p}$$

  - Utilization should be divided by N_r :

$$U = \frac{1-p}{1+2a}$$

- Selective Reject Utilization
  - Exact same idea as stop and wait

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

Link Utilization

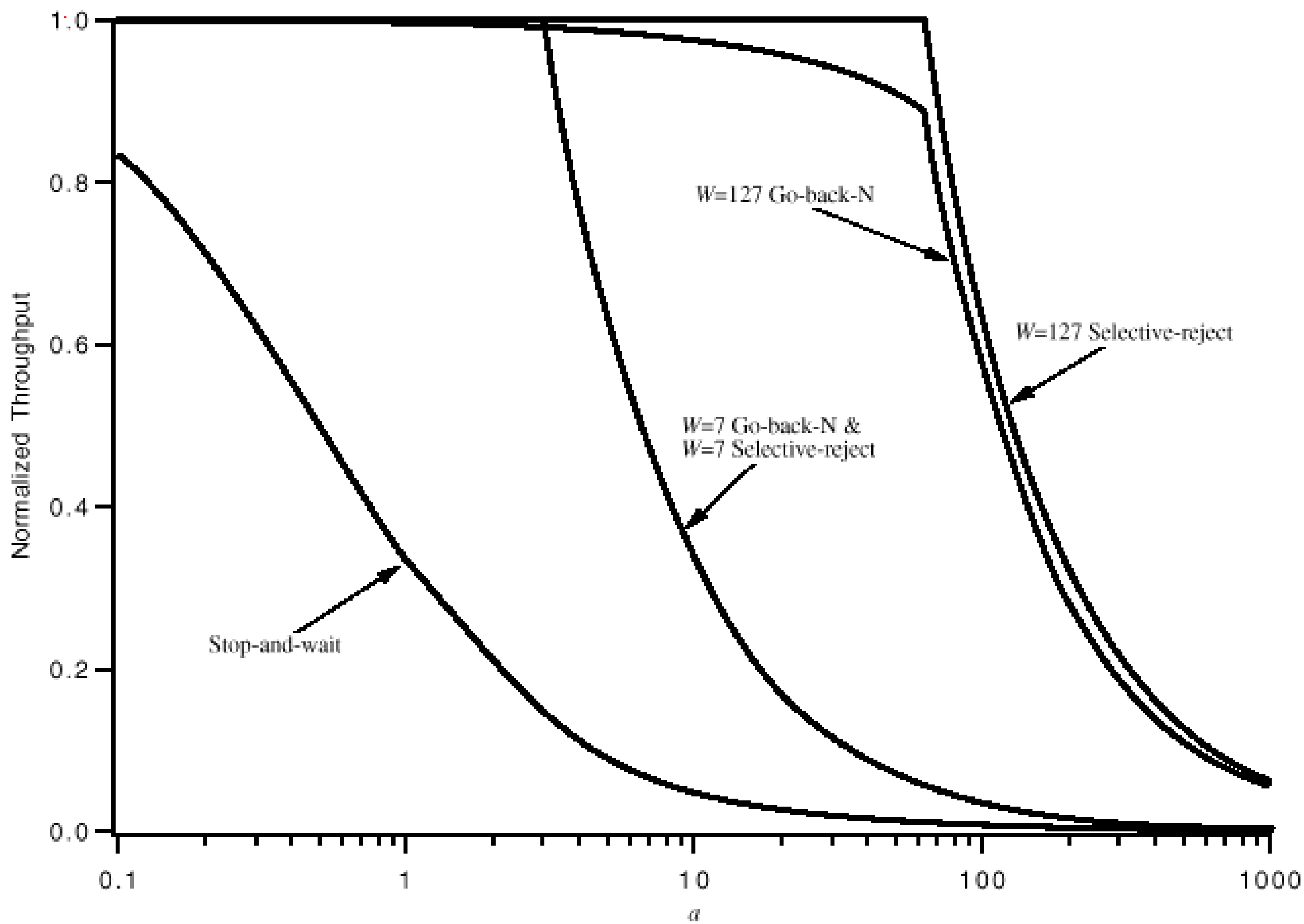
- Go back N Utilization
  - Each frame error requires re-transmission of *L* packets where *L*≥1
  - Total number of frames that should be transmitted if the original frame must be transmitted k times=*f*(*k*)=*1*+(*k*-*1*)*L*

$$E[N] = N_r = \sum_{k=1}^{\infty} f(k) \times \mathrm{Prob}(k\ Transmission) = \sum_{k=1}^{\infty} (1 - L + kL) p^{k-1} (1-p)$$

$$= (1-L)\sum_{k=1}^{\infty} p^{k-1}(1-p) + L\sum_{k=1}^{\infty} kp^{k-1}(1-p) = (1-L) + \frac{L}{1-p} = \frac{1-p+LP}{1-p}$$

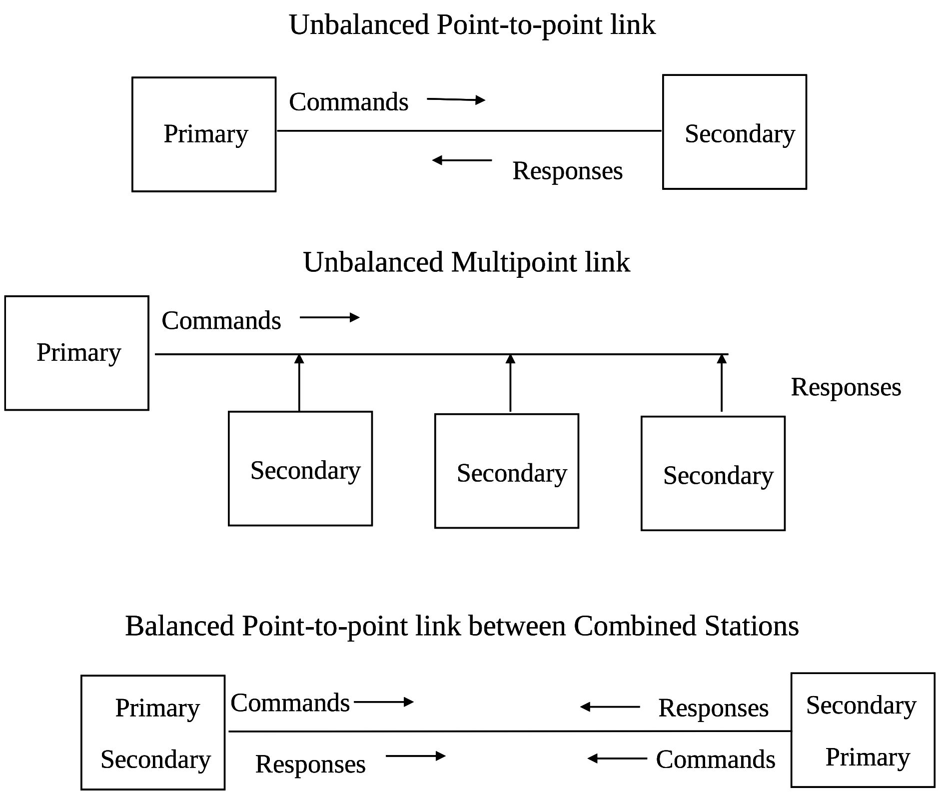
$$L \approx \begin{cases} 1+2a & W > 1+2a \\ W & W \le 1+2a \end{cases} \qquad U = \begin{cases} \dfrac{1-p}{1+2ap} & W > 2a+1 \\ \dfrac{W(1-p)}{(2a+1)(1-p+Wp)} & W < 2a+1 \end{cases}$$

Link Utilization



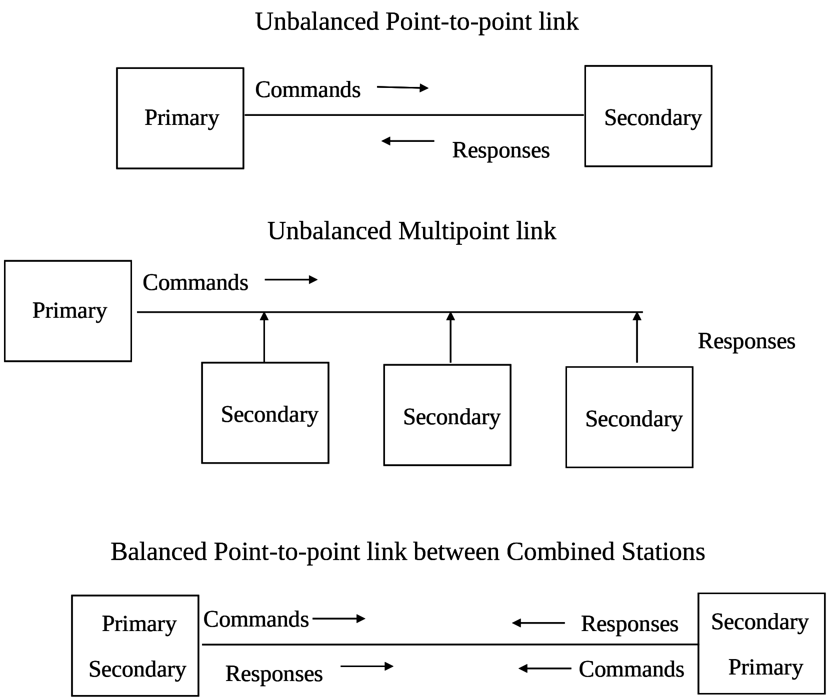
HDLC

- HDLC: high-level data link control
  - Common in X.25 networks
  - Derived from IBM’s SNA
  - Bit-oriented, bit-stuffing
- Station Types:
  - Primary station
    - Controls operation of link
    - Frames issued are called commands
    - Maintains separate logical link to each secondary station
  - Secondary station
    - Under control of primary station
    - Frames issued called responses
  - Combined station
    - May issue commands and responses



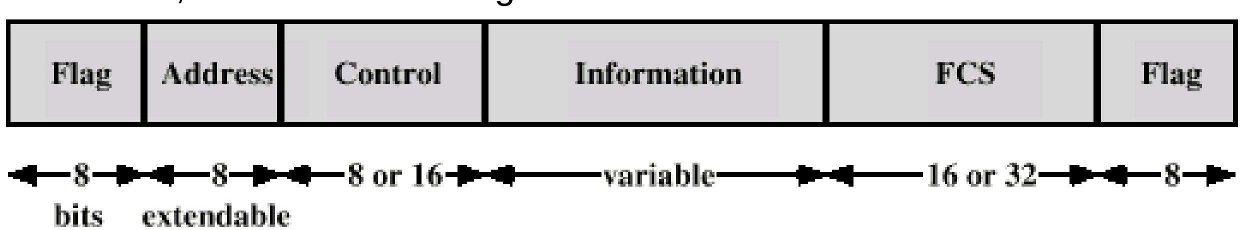
HDLC

- HDLC Link Configurations (Full duplex & Half duplex)
  - Unbalanced: One primary and one or more secondary stations
  - Balanced: Two combined stations
- Normal Response Mode (NRM)
  - Unbalanced configuration
  - Primary initiates transfer to secondary. Secondary may only transmit data in response to command from primary
  - Used on multi-drop lines
  - Host computer as primary.
  - Terminals as secondary
- Asynchronous Balanced Mode (ABM)
  - Balanced configuration (Most widely used)
  - Either station may initiate transmission without receiving permission. No polling overhead



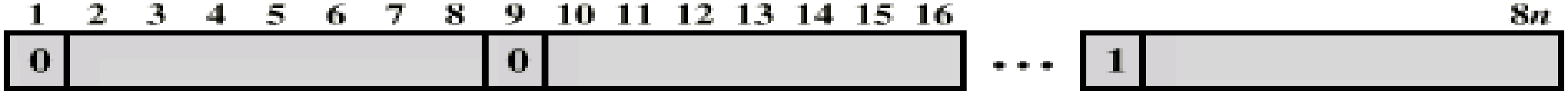
HDLC

- Frame Structure
  - All transmissions in frames
  - Single frame format for all data and control exchanges
  - On idle point to point links, flag sequences are transmitted continuously
- Flag Fields
  - Delimit frame at both ends with 01111110
  - May close one frame and open another
  - Receiver hunts for flag sequence to synchronize
  - Bit stuffing used to avoid confusion with data containing 01111110
    - 0 inserted after every sequence of five 1s
    - If receiver detects five 1s it checks next bit
    - If 0, it is deleted
    - If 1 and seventh bit is 0, accept as flag
    - If sixth and seventh bits 1, sender is indicating abort



HDLC Frame Fields

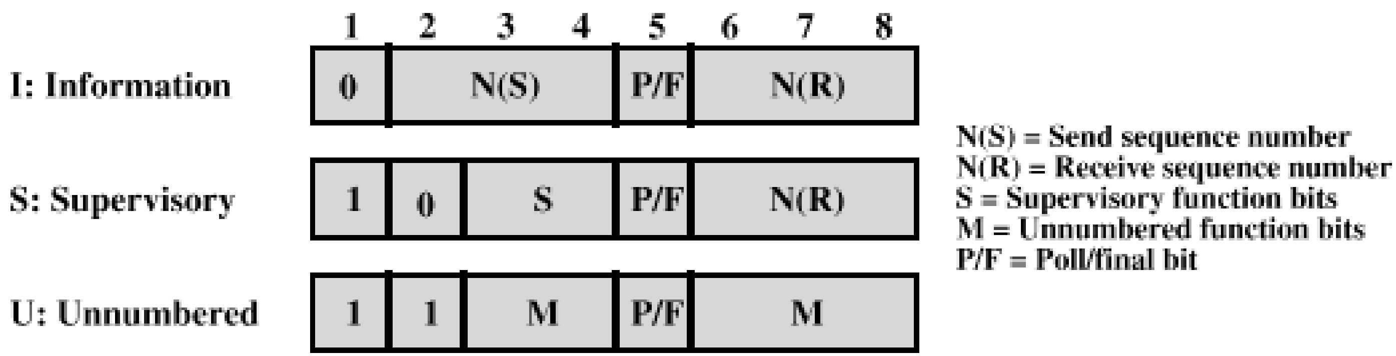
- Address Field
  - Identifies secondary station that sent or will receive frame
  - Usually 8 bits long
  - May be extended to multiples of 7 bits
    - LSB of each octet indicates that it is the last octet (1) or not (0)
  - All ones (11111111) is for broadcast



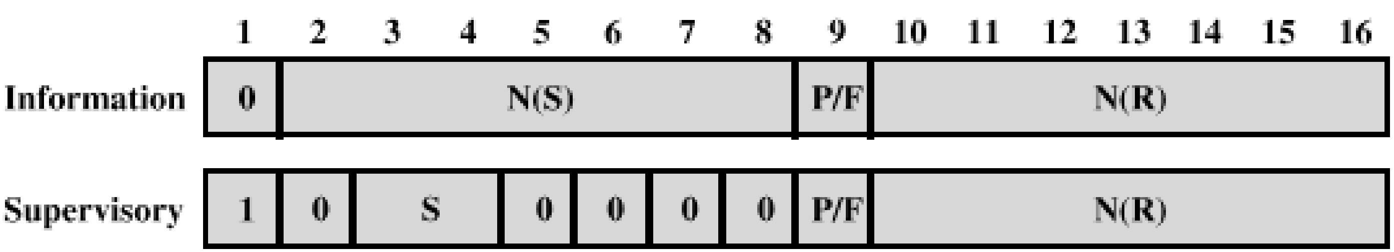
(b) Extended Address Field

HDLC Control Field

- Control frame varies for different frame types
  - Information - data to be transmitted to user (next layer up)
    - Flow and error control piggybacked on information frames
  - Supervisory - ARQ when piggyback not used
  - Unnumbered - supplementary link control
- First one or two bits of control field: frame type identification



(c) 8-bit control field format



(d) 16-bit control field format

HDLC frame types

- Supervisory frames
  - Receive Ready Frame (SS=00)
    - Used when there are no information frames available to piggyback the ACK
  - Reject Frame (SS=01)
    - Receiver tells the sender to go back and start sending from the frame number N(R)
  - Receive Not Ready (SS=10)
    - All frames up to frame number N(R) have been received properly. But the receiver can not accept any frames at this time. (e.g. due to a full buffer)
  - Selective Reject (SS=11)
    - Receiver asks the sender to re-send frame number N(R)

- Unnumbered frames
- Used to start up the link or tear it down
- Some examples:
  - SABM: Set ABM Mode
  - SNRM: Set NRM Mode
  - SABME: Set ABM Extended Mode
  - DISC: Disconnect
  - UA: Unnumbered Acknowledgment
  - FRMR: Frame Reject

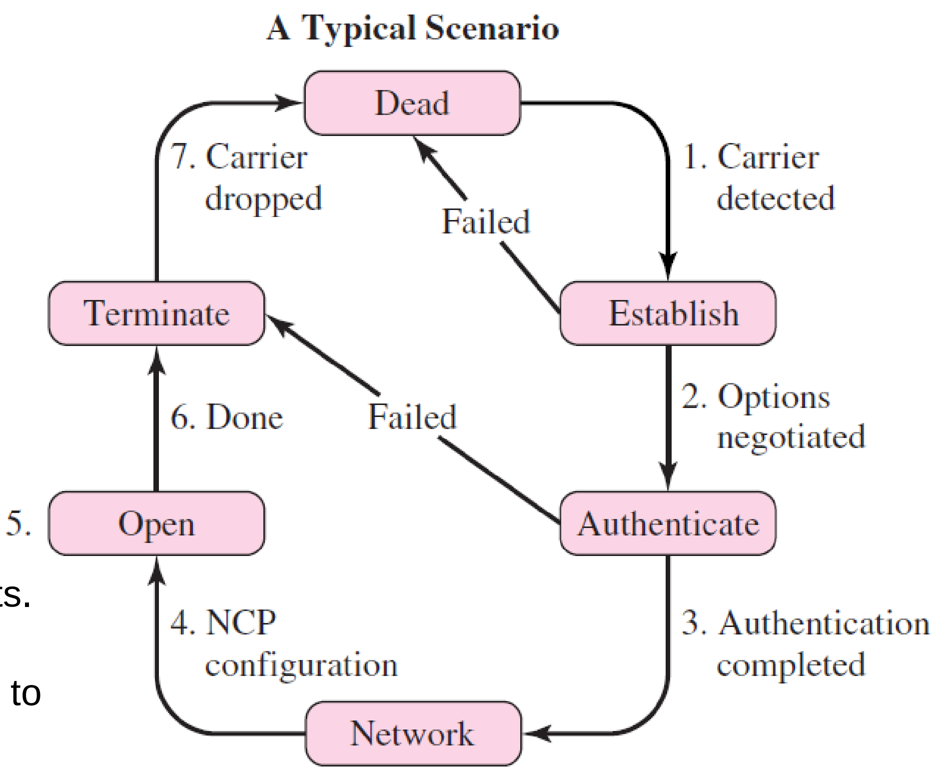
PPP - Point-to-Point Protocol

- Provides a method for encapsulation of network layer packets over point to point links. It provides:
  - An HDLC-like Framing with error detection (Byte oriented)
  - LCP (link control protocol) for bringing up lines, testing, negotiating options
  - NCP (network control protocol) for each supported network layer
- PPP frame format
  - Flag byte: 0111.1110 / character stuffing
  - Address field (default = 1111.1111)
  - Control field (default = 0000.0011)
  - Unnumbered frames with unreliable transmission; no sequence numbers
  - Protocol field: indicates packet type
    - 0x indicates IP, IPX or XNS
    - 1x indicates LCP, NCP or others
  - Payload: 1500 bytes default length
  - Checksum of 2 (default) or 4 bytes

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

Typical PPP Operation

- Example 1: Home PC to Internet Service Provider
  - PC calls router via modem.
  - PC and router exchange LCP packets to negotiate PPP parameters.
  - Check on identities.
  - NCP packets exchanged to configure the network layer, e.g., TCP/IP (requires IP address assignment).
  - Data transport, e.g. send/receive IP packets.
  - NCP used to tear down the network layer connection (free up IP address); LCP used to shut down data link layer connection.
  - Modem hangs up.
- Example 2: PPP use for transmission of packets over SONET SDH links



Data link layer in ATM

- There are no physical layer characteristics specific to ATM
  - ATM cells can be carried by various physical layer services such as SONET/SDH, ADSL or wireless links
- TC (transmission convergence) sub layer corresponds to DL layer
- Transmission: each cell of 53 byte includes 5 byte header
  - Add 4 bytes of identification and 1 byte of HEC (header checksum)
  - No checksum for payload. (BER<< 10^{-12} in fiber optic links)
  - Match speed to underlying physical transmission system by inserting empty or maintenance (OAM) cells
- Reception of a cell
  - Locate cell boundaries
  - Verify header
  - Discard cells with invalid headers
  - Process maintenance cells
  - Pass data to ATM layer

Data link layer in ATM

- How to detect cell boundaries (there are no marks !!)
  - Trick: constantly check HEC field after each incoming bit
  - Use 40-bits shift register to shift the incoming pattern bit-by-bit until the frame header is found
  - After the first header match, jump 53 bytes (length of an ATM frame) and re-check the HEC (is this the right HEC for the assumed cell?)
  - Repeat the header check d times in a row, If all correct, go to synch Mode
  - Change of accidental correct synchronization = 2^{-8d}
  - ‘DeSynch’ after a incorrect headers
  - TC sub layer uses header of ATM field for synchronization: no proper separation of layers. E.g. changing header format for ATM affects TC layer