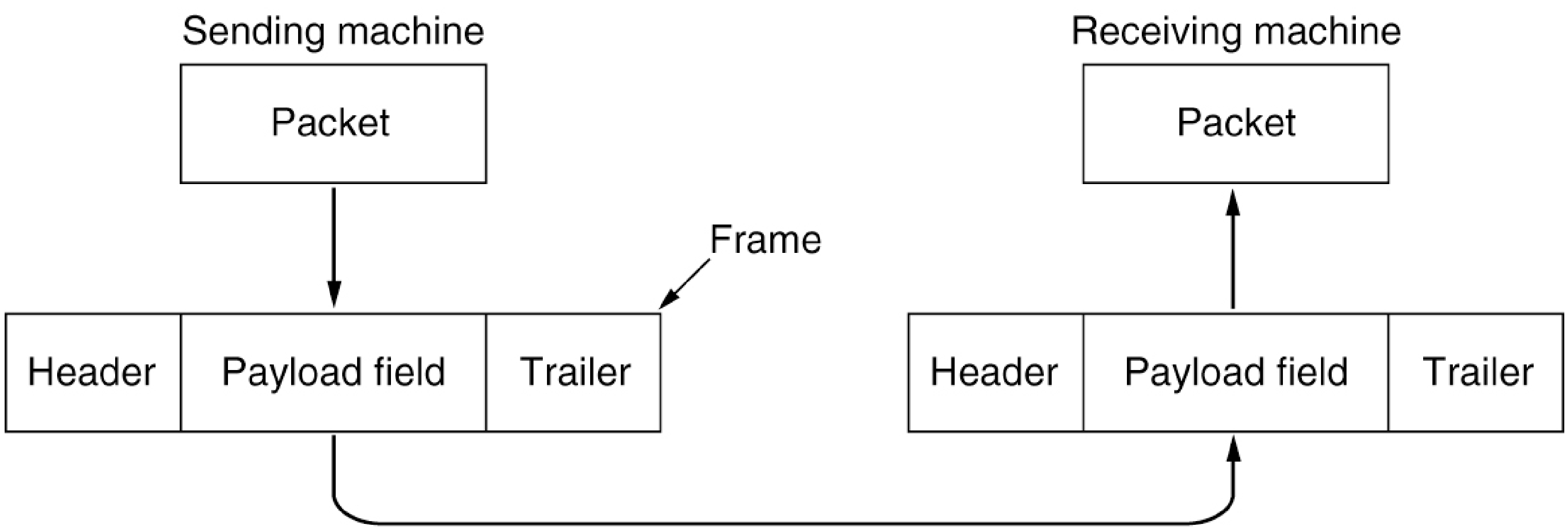
ECE 363
Communication Networks

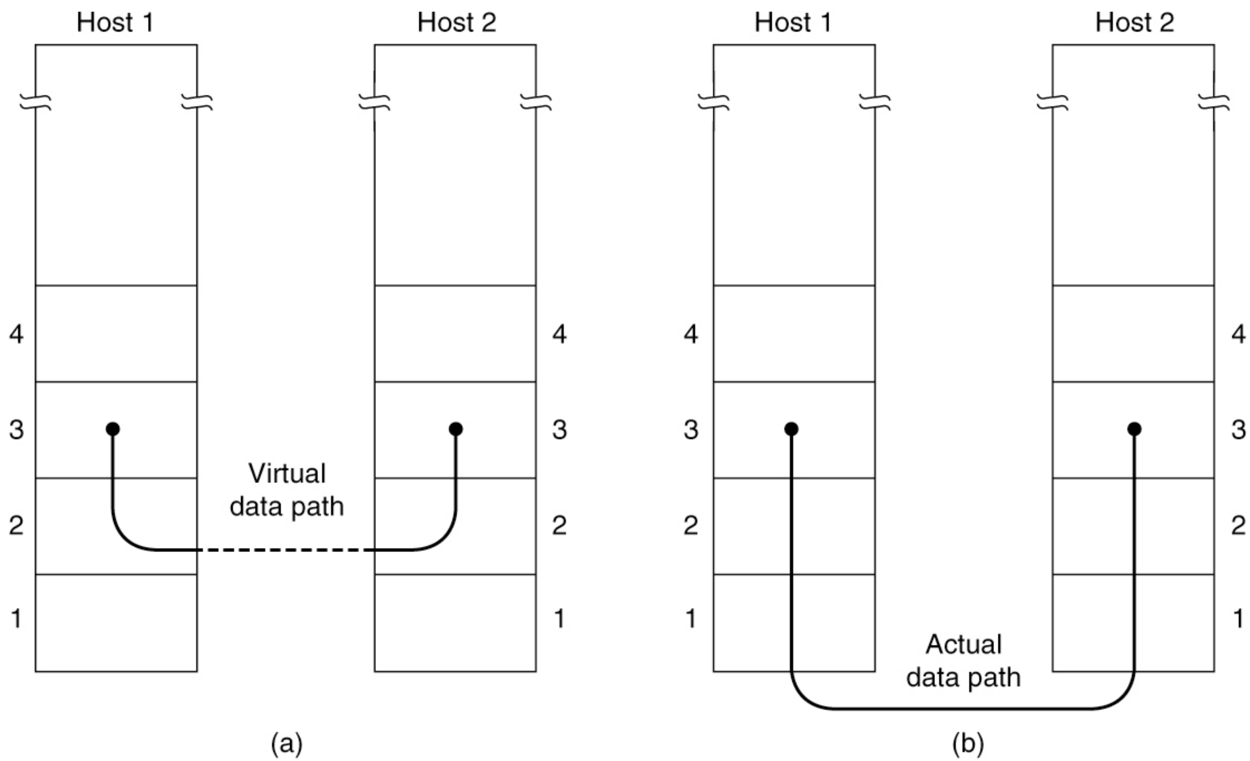
Data Link Layer

Data Link Layer



Relationship between packets and frames

Services Provided to The Network Layer



(a) Virtual communication (b) Actual communication

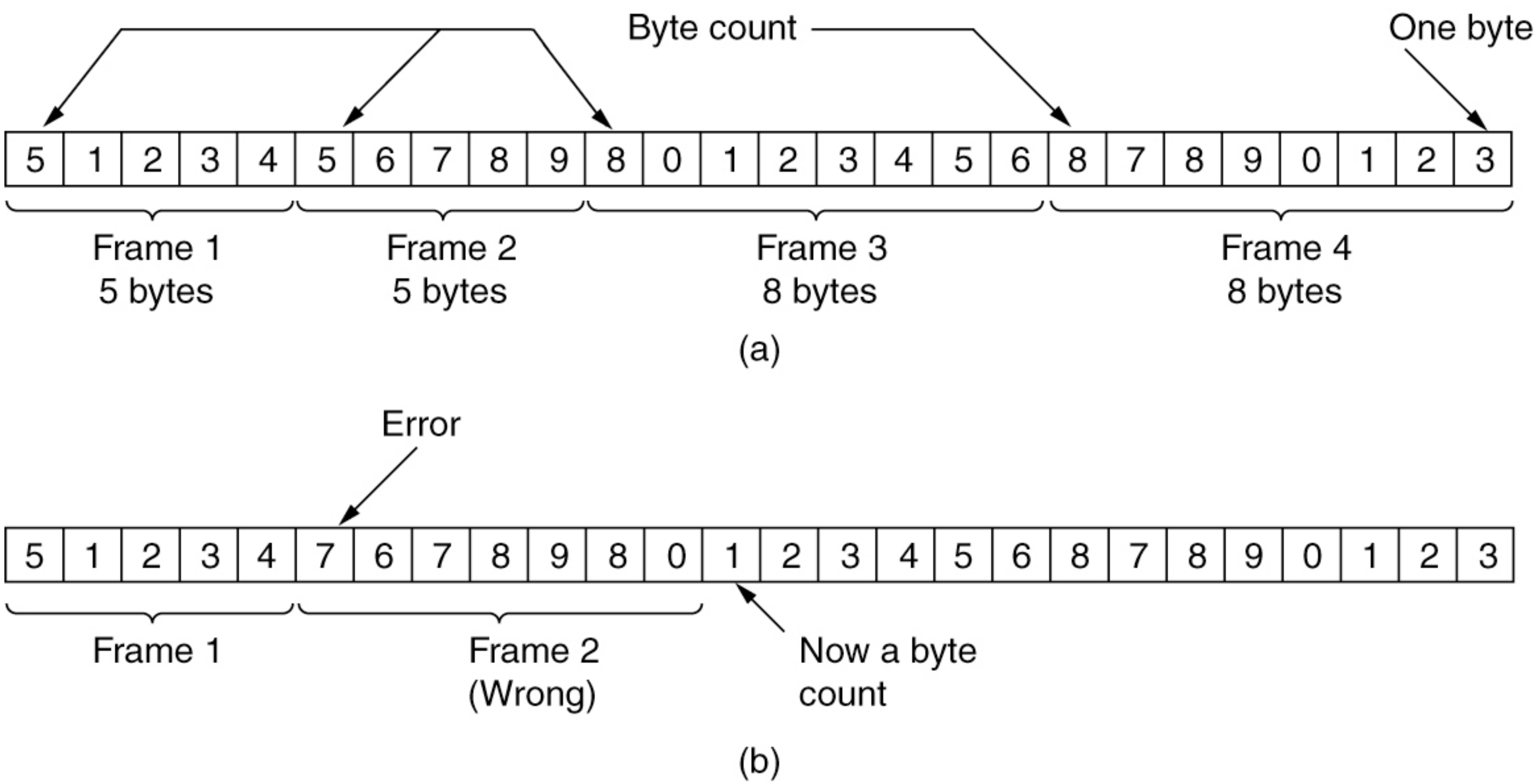
Services Provided to The Network Layer

- Frame delivery
- Error control
- Flow control
- Medium access (with a shared medium)

Framing

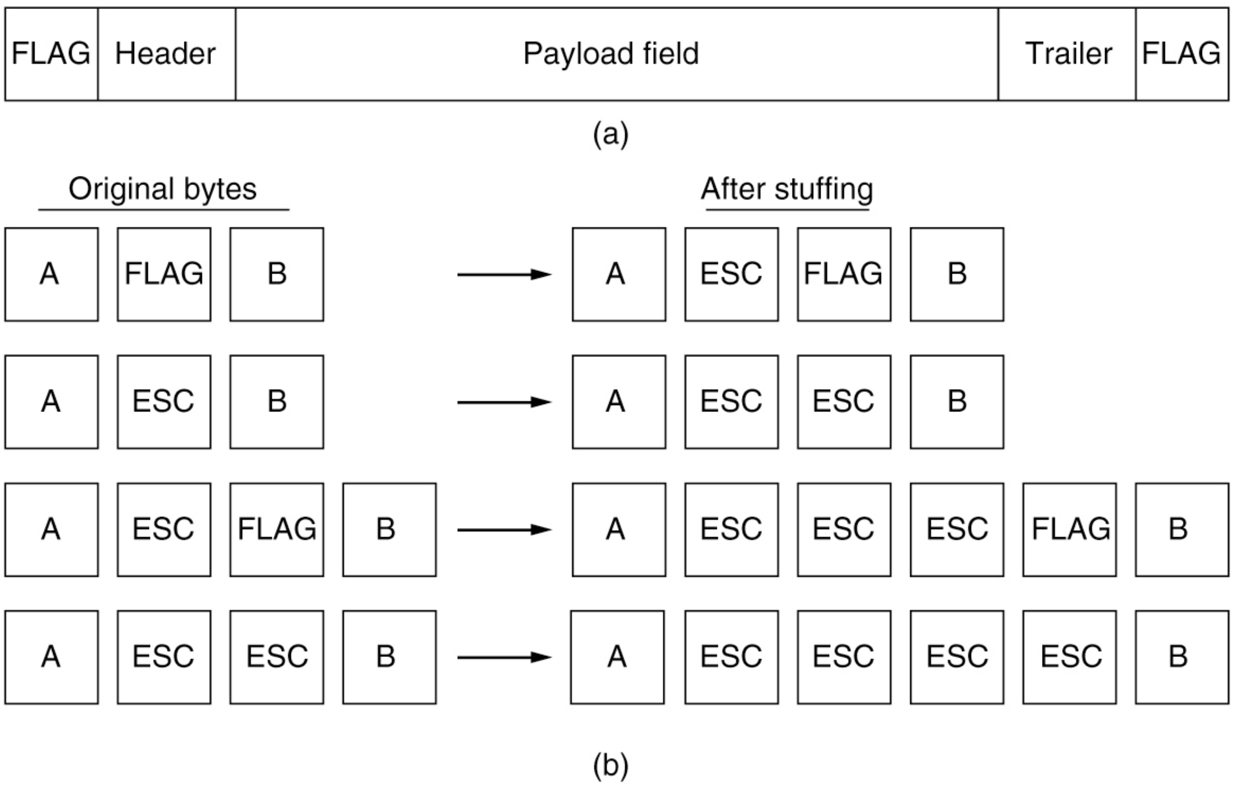
- Framing methods
  - Byte count
  - Flag bytes with byte stuffing
  - Flag bits with bit stuffing
  - Physical layer coding violations

Byte Count



A byte stream (a) without errors and (b) with one error.

Flag Bytes With Byte Stuffing



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

Flag Bits With Bit Stuffing

- Flag bits: 01111110
- Data transparency: bit stuffing
- Sender: insert a 0 after 5 1s
- Receiver: remove a 0 after 5 1s
- Developed for the High-level Data Link Control (HDLC) protocol

Bits	8	8	8	≥ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

Flag Bits With Bit Stuffing

(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.

Physical Layer Coding Violations

Data (4B)	Codeword (5B)	Data (4B)	Codeword (5B)
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

4B/5B mapping

Error Control

- Ensure all frames are eventually delivered
  - to the network layer at the destination
  - in the proper order
- Acknowledgement (ACKs) and timers
  - Checksums are used to determine if the frames are correct or not
- Error correction and detection

Error Correction and Detection

- Error-correcting codes
  - Referred to as Forward Error Correction (FEC)
  - Add redundant information to enable the receiver to estimate what the transmitted data was
- Error-detecting codes
  - Add redundant information to allow the receiver to determine that errors have occurred (but not where) and then request a retransmission

Error Correcting Codes

- Triple repetition code

Codewords

$\mathbf{d} = 0 \quad \mathbf{c} = 000$

$\mathbf{d} = 1 \quad \mathbf{c} = 111$

- Block length $n = 3$
- Message length $m = 1$
- $r = n - m = 2$ parity check bits
- Called an (n,m) code
- Code rate $m/n = 1/3$

Error Correcting Codes

- The number of positions two codewords differ is called the Hamming distance d
- This can be obtained by XORing the codewords and counting the number of 1s
- Example: Codewords 10001001 and 10110001

```
  10001001
+ 10110001
  -------------
  00111000 → d = 3
```

- The smallest Hamming distance between all pairs of codewords is called the minimum distance d_{min}
- For the (3,1) code $d_{min} = 3$

Error Correcting Codes

- To detect d or fewer errors requires $d_{min} = d + 1$
  - d errors will not transform a codeword to another codeword
- To correct d or fewer errors requires $d_{min} = 2d + 1$
  - A received codeword with d errors will be closer to the correct codeword than any other codeword
- Example: For the (3,1) code $d_{min} = 3$
- If **c** = 000 and the received word is **r** = 110
  - **r** is not a valid codeword so the two errors have been detected
- If **c** = 000 and the received word is **r** = 100
  - **r** is closer to the correct codeword **c** so the error is corrected

Hamming Codes

- Consider the design of a code with m message bits that can correct all single bit errors
- $n = m + r$
- There are n single bit error patterns
  - 10…00 to 00…01

  and 1 0-bit error pattern
  - 00…00
- Each codeword is associated with $n + 1$ distinct received words of length n
- 2^m codewords
- 2^n possible received words
- Then $(n + 1)2^m \leq 2^n$
- $n + 1 \leq 2^r \rightarrow m + r + 1 \leq 2^r - 1$

Hamming Codes

- $n = 2^r - 1$
- $m = 2^r - r - 1$
- Optimal single error correcting codes
- Code parameters
  - (7,4)
  - (15,11)
  - (31,26)
  - (63,57)
  - (127,120)
  - (255,247)

Hamming Codes

- Label the bit positions from 1 to *n*
- The *r* check bits are in the positions which are powers of 2: 1, 2, 4, 8, …
- The *m* message bits are in the other positions
- Express the location of message bit *k* as powers of 2
    - Example: 11 = 1 + 2 + 8
    - These are the check bits this data bit contributes to

Hamming Codes

- Example: $n = 7$, $m = 4$, $r = 3$
- The r check bits are in positions 1,2,4
- Let **d** = 0101 place these bits in positions 3,5,6,7

  _ _ 0 _ 1 0 1

- Check bit 1 is 3+5+7 0+1+1 = 0
- Check bit 2 is 3+6+7 0+0+1 = 1
- Check bit 4 is 5+6+7 1+0+1 = 0
- Codeword **c** = 0100101

Hamming Codes

- Suppose 0100101 is received
  - Check 1:   1+3+5+7   0+0+1+1 = 0
  - Check 2:   2+3+6+7   1+0+0+1 = 0
  - Check 4:   4+5+6+7   0+1+0+1 = 0
  - The syndrome is 000 so no errors

    c = 0100101 and **d** = 0101
- Suppose 0100111 is received
  - Check 1:   1+3+5+7   0+0+1+1 = 0
  - Check 2:   2+3+6+7   1+0+1+1 = 1
  - Check 4:   4+5+6+7   0+1+1+1 = 1
  - The syndrome is 110 so the error is in position 6

    c = 0100101

ASCII Character Set

Least Significant Bits

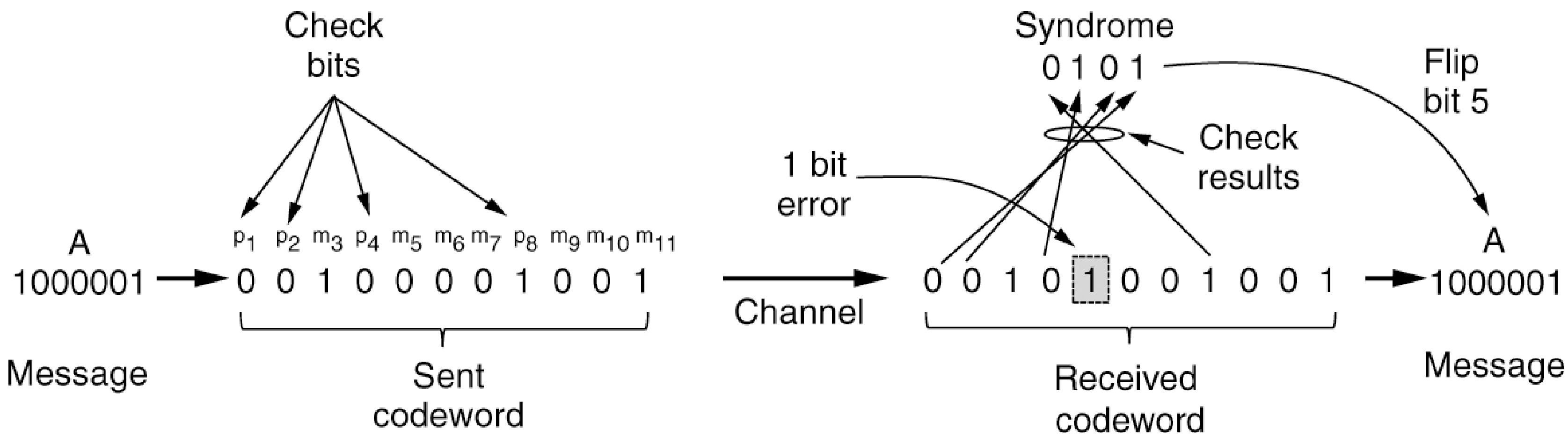
Most Significant Bits

	0 **0000**	**1** **0001**	**2** **0010**	**3** **0011**	**4** **0100**	**5** **0101**	**6** **0110**	**7** **0111**	**8** **1000**	**9** **1001**	**A** **1010**	**B** **1011**	**C** **1100**	**D** **1101**	**E** **1110**	**F** **1111**
0 **000**	**NUL** (0) 00	**SOH** (1) 01	**STX** (2) 02	**ETX** (3) 03	**EOT** (4) 04	**ENQ** (5) 05	**ACK** (6) 06	**BEL** (7) 07	**BS** (8) 08	**HT** (9) 09	**LF** (10) 0A	**VT** (11) 0B	**FF** (12) 0C	**CR** (13) 0D	**SO** (14) 0E	**SI** (15) 0F
1 **001**	**DLE** (16) 10	**DC1** (17) 11	**DC2** (18) 12	**DC3** (19) 13	**DC4** (20) 14	**NAK** (21) 15	**SYN** (22) 16	**ETB** (23) 17	**CAN** (24) 18	**EM** (25) 19	**SUB** (26) 1A	**ESC** (27) 1B	**FS** (28) 1C	**GS** (29) 1D	**RS** (30) 1E	**US** (31) 1F
2 **010**	**SP** (32) 20	**!** (33) 21	**"** (34) 22	**#** (35) 23	**$** (36) 24	**%** (37) 25	**&** (38) 26	**'** (39) 27	**(** (40) 28	**)** (41) 29	***** (42) 2A	**+** (43) 2B	**,** (44) 2C	**-** (45) 2D	**.** (46) 2E	**/** (47) 2F
3 **011**	**0** (48) 30	**1** (49) 31	**2** (50) 32	**3** (51) 33	**4** (52) 34	**5** (53) 35	**6** (54) 36	**7** (55) 37	**8** (56) 38	**9** (57) 39	**:** (58) 3A	**;** (59) 3B	**<** (60) 3C	**=** (61) 3D	**>** (62) 3E	**?** (63) 3F
4 **100**	**@** (64) 40	**A** (65) 41	**B** (66) 42	**C** (67) 43	**D** (68) 44	**E** (69) 45	**F** (70) 46	**G** (71) 47	**H** (72) 48	**I** (73) 49	**J** (74) 4A	**K** (75) 4B	**L** (76) 4C	**M** (77) 4D	**N** (78) 4E	**O** (79) 4F
5 **101**	**P** (80) 50	**Q** (81) 51	**R** (82) 52	**S** (83) 53	**T** (84) 54	**U** (85) 55	**V** (86) 56	**W** (87) 57	**X** (88) 58	**Y** (89) 59	**Z** (90) 5A	**[** (91) 5B	**** (92) 5C	**]** (93) 5D	**^** (94) 5E	**_** (95) 5F
6 **110**	**`** (96) 60	**a** (97) 61	**b** (98) 62	**c** (99) 63	**d** (100) 64	**e** (101) 65	**f** (102) 66	**g** (103) 67	**h** (104) 68	**i** (105) 69	**j** (106) 6A	**k** (107) 6B	**l** (108) 6C	**m** (109) 6D	**n** (110) 6E	**o** (111) 6F
7 **111**	**p** (112) 70	**q** (113) 71	**r** (114) 72	**s** (115) 73	**t** (116) 74	**u** (117) 75	**v** (118) 76	**w** (119) 77	**x** (120) 78	**y** (121) 79	**z** (122) 7A	**{** (123) 7B	**|** (124) 7C	**}** (125) 7D	**~** (126) 7E	**DEL** (127) 7F

Hamming Codes

- Suppose the message is ASCII A
  - **d** = 1000001
- m = 7
- n = m + r = 7 + 4 = 11
- **c** = 00100001001

(11,7) Hamming Code



- Check 1   1+3+5+7+9+11   0+1+1+0+0+1 = 1
- Check 2   2+3+6+7+10+11   0+1+0+0+0+1 = 0
- Check 4   4+5+6+7   0+1+0+0 = 1
- Check 8   8+9+10+11   1+0+0+1 = 0
- Syndrome is 0101
- Error is in position 5

Error Detecting Codes

- Single Parity Check (SPC) codes
- Cyclic Redundancy Check (CRC) codes

Single Parity Check (SPC) Code

ASCII symbols	Codewords
E = 1000101	**c** = 10001011
G = 1000111	**c** = 10001110

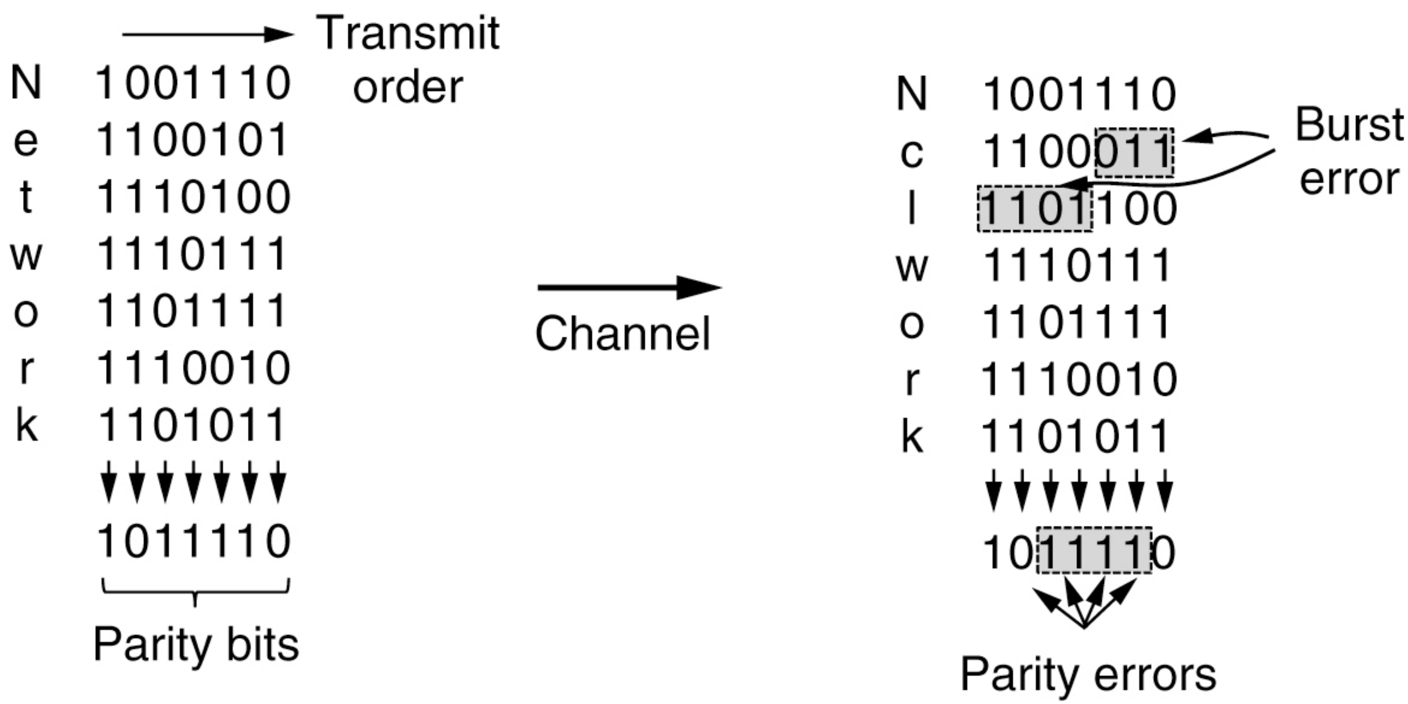
Received word

10001010

Burst Error Detection

- Arrange the data in a rectangular matrix with *k* codewords in columns
- Transmit in row order
- Can detect all burst errors of *k* bits or less

Error Detecting Codes



Interleaving codewords to detect burst errors

Binary Polynomial Arithmetic

- Binary vectors map to polynomials

$$(i_{k-1}, i_{k-2}, \ldots, i_2, i_1, i_0) \rightarrow i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_2x^2 + i_1x + i_0$$

Addition

$$(x^7 + x^6 + 1) + (x^6 + x^5) = x^7 + x^6 + x^6 + x^5 + 1$$

$$= x^7 + (1+1)x^6 + x^5 + 1$$

$$= x^7 + x^5 + 1 \quad \text{since } 1+1=0 \bmod 2$$

Multiplication

$$(x + 1)(x^2 + x + 1) = x(x^2 + x + 1) + 1(x^2 + x + 1)$$

$$= (x^3 + x^2 + x) + (x^2 + x + 1)$$

$$= x^3 + 1$$

CRC Encoding

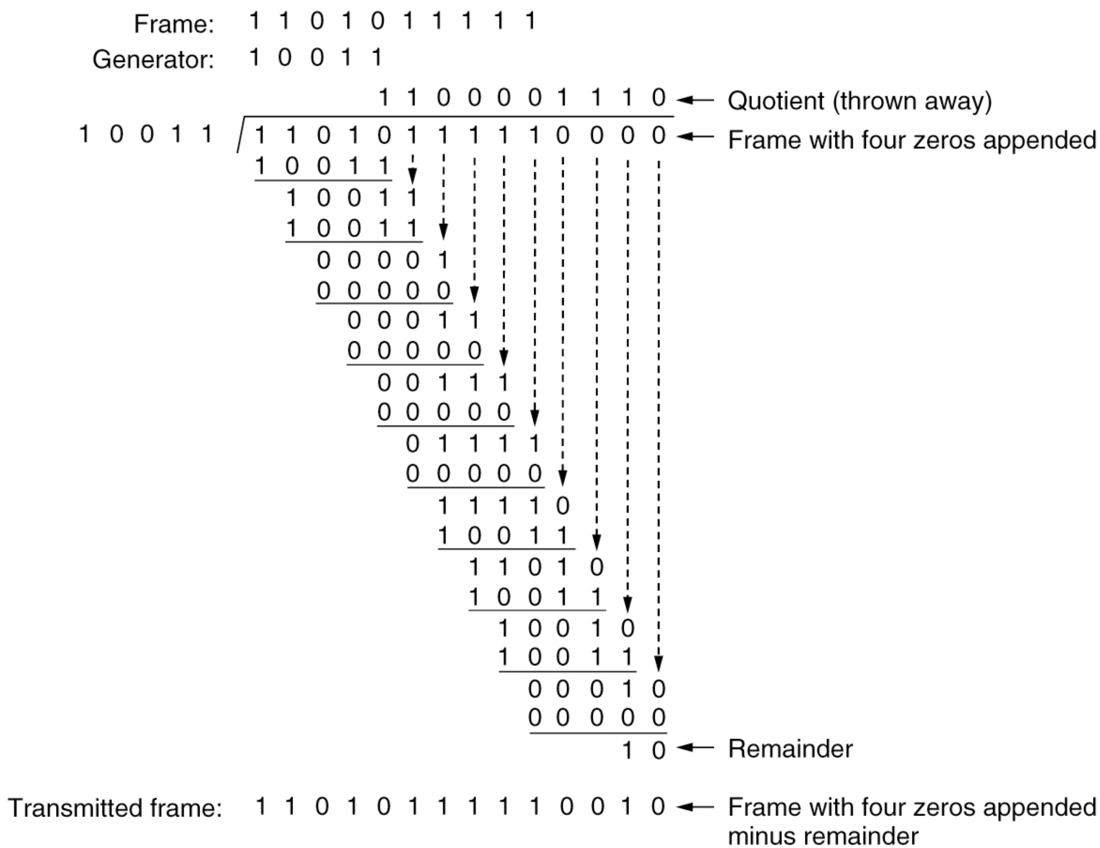
1. Let r be the degree of $G(x)$
2. Multiply $M(x)$ by x^r
3. Divide $x^rM(x)$ by $G(x)$

   $$x^rG(x) = G(x)Q(x) + R(x)$$

3. Add remainder $R(x)$ to $x^rM(x)$

   $T(x) = x^rM(x) + R(x)$ ← transmitted codeword

Error Detecting Codes

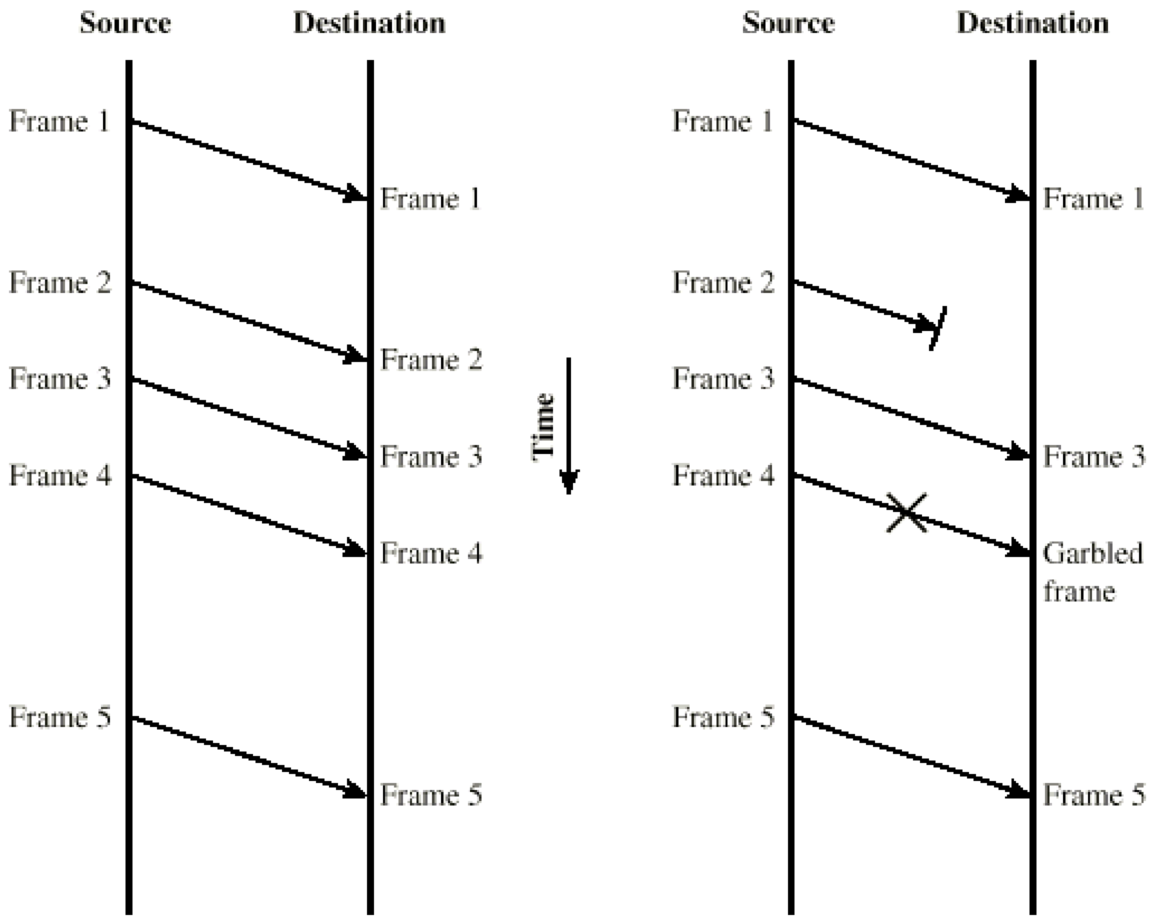


Example CRC calculation

CRC Polynomials

- CRC-4 Interlaken
  - $x^4+x^3+x^2+x+1$
- CRC-8 CDMA2000
  - $(x^5+x^4+x^3+x^2+1)(x^2+x+1)(x+1)=x^8+x^7+x^6+x^4+x^2+1$
- CRC-16 PPP, Bluetooth
  - $x^{16}+x^{12}+x^5+1$
- CRC-32 Ethernet, IEEE 802
  - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Frame Transmission



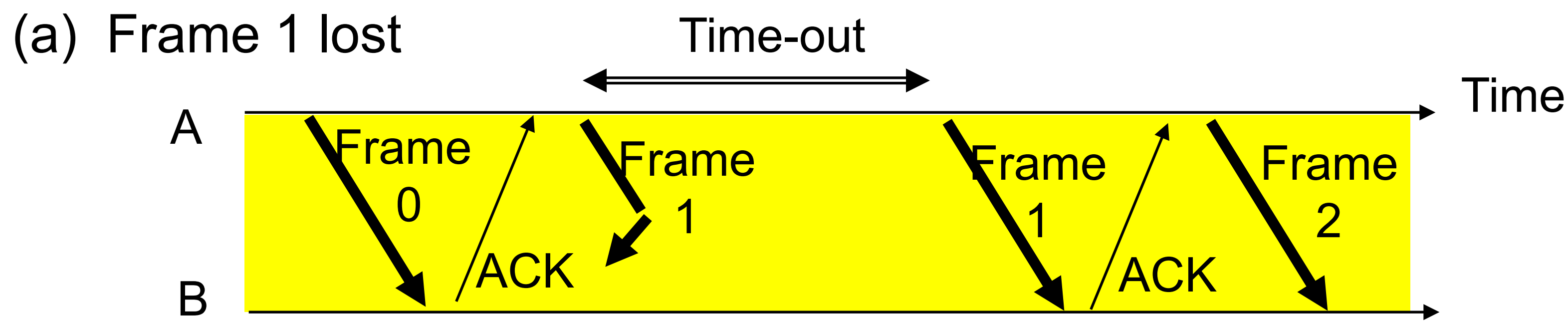
(a) Error-free transmission

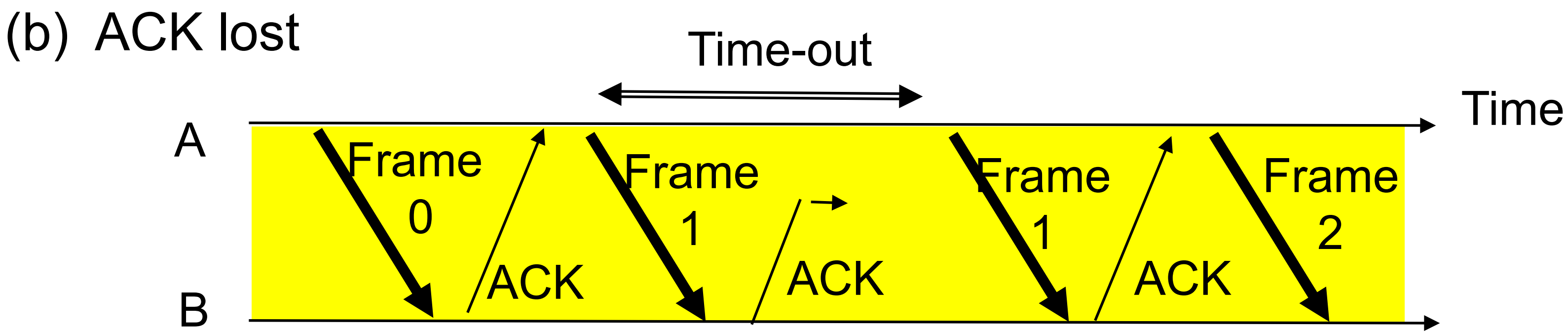
(b) Transmission with losses and errors

Stop and Wait Protocol

- Sender transmits a frame
- Receiver receives the frame and replies with an acknowledgement (ACK)
- Sender waits for ACK before sending the next frame

Stop and Wait Protocol





- In cases (a) and (b), A acts the same way, but in (b), B accepts frame 1 twice
- Question: How can the receiver know the second frame is also frame 1?
- Answer: Add a frame sequence number in the header

Automatic Repeat Request (ARQ)

- Basic elements of ARQ
  - Error detection
  - ACKs (positive acknowledgments)
  - NAKs (negative acknowledgments)
  - Timeouts

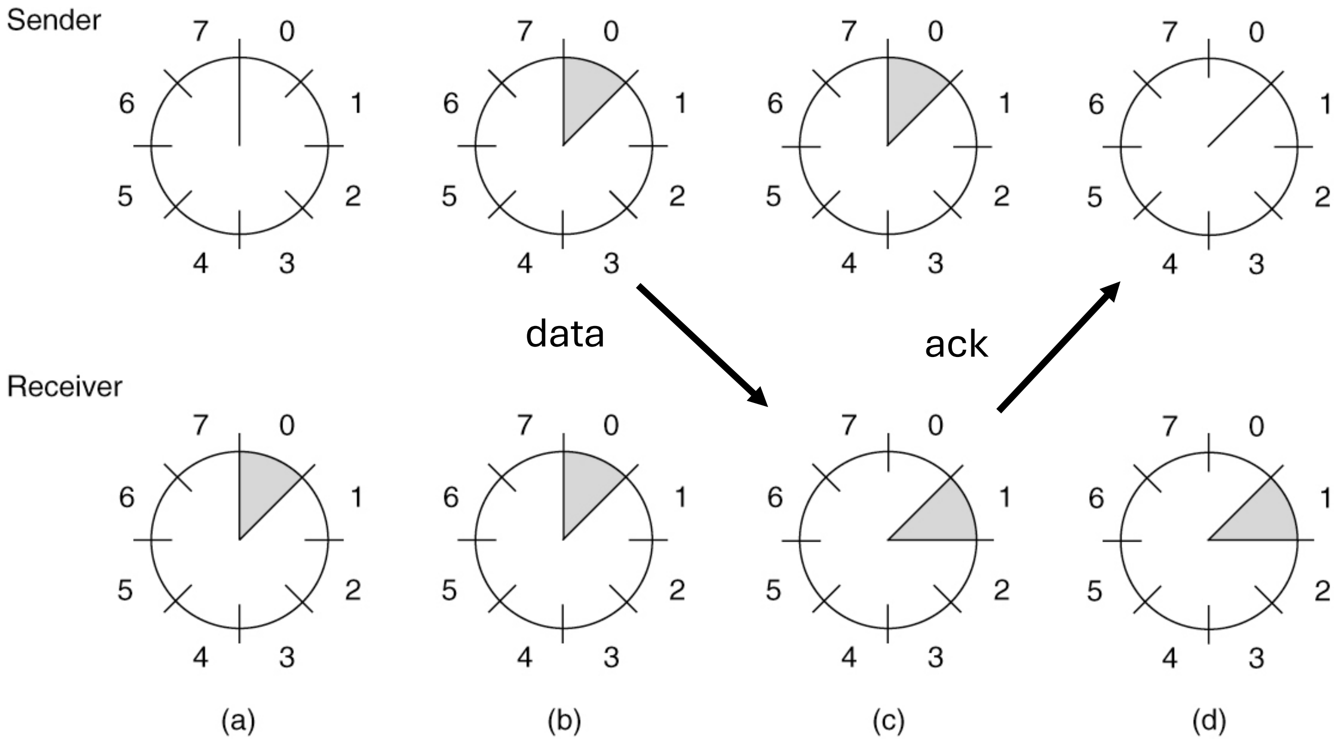
Sliding Window Flow Control

- Allow multiple frames to be in transit
- Sending window
  - List of unacknowledged frames
- Receiving window
  - List of frames it is permitted to receive
- Each frame is numbered
- Sequence numbers are bounded by the size of the field *n* bits
  - Frames are numbered modulo 2^n

Sliding Window Protocols

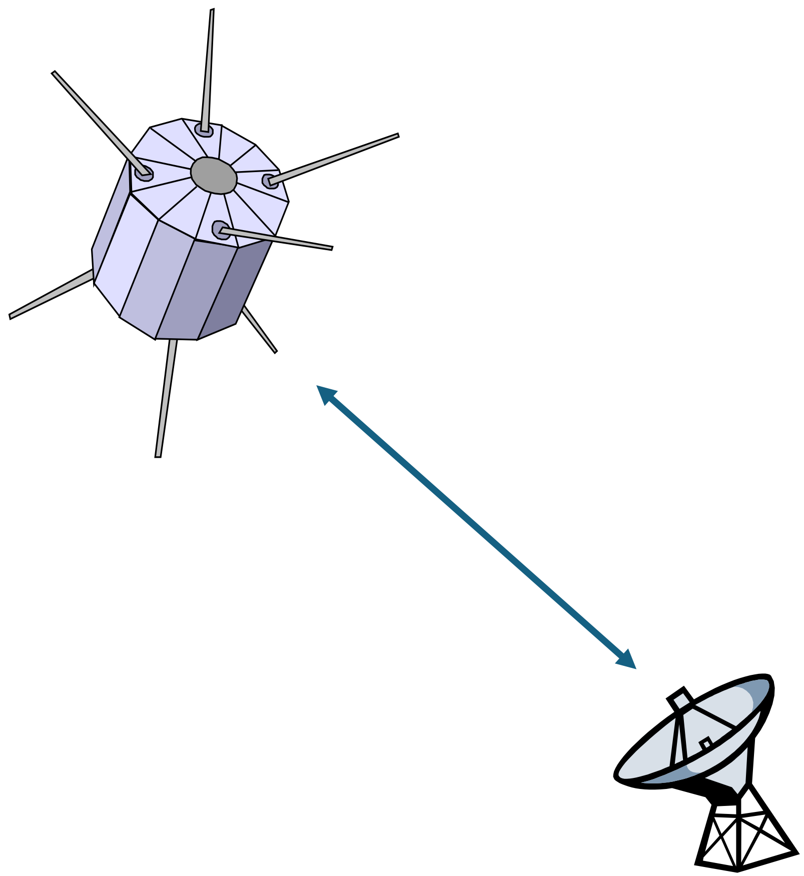
- One-bit sliding window
- Go-Back-N (GBN)
- Selective Repeat (SR)
- At any instant of time
  - Sender maintains a set of sequence numbers corresponding to frames it is permitted to send
  - Frames are said to fall within the sending window
  - Receiver maintains a receiving window corresponding to the set of frames it is permitted to accept
- Differ in terms of efficiency, complexity, and buffer requirements

Flow Control

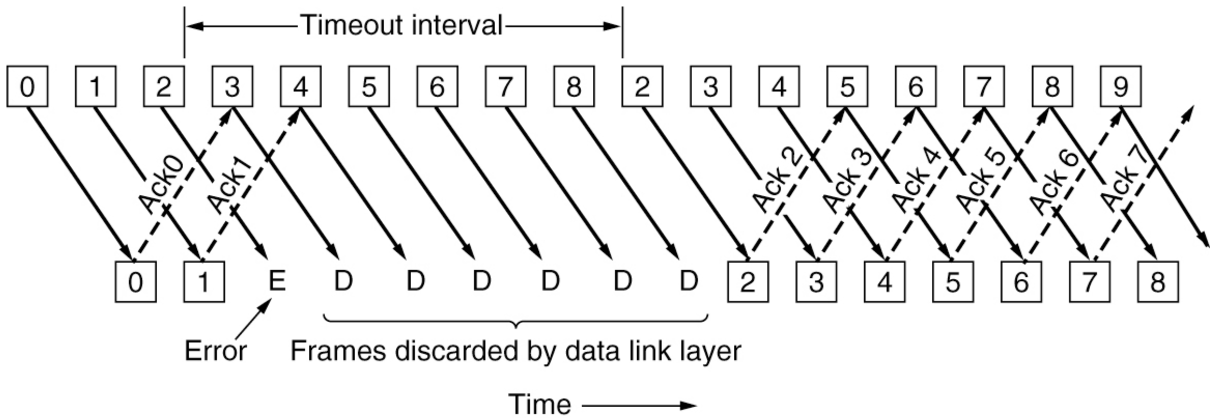


A sliding window of size 1 with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

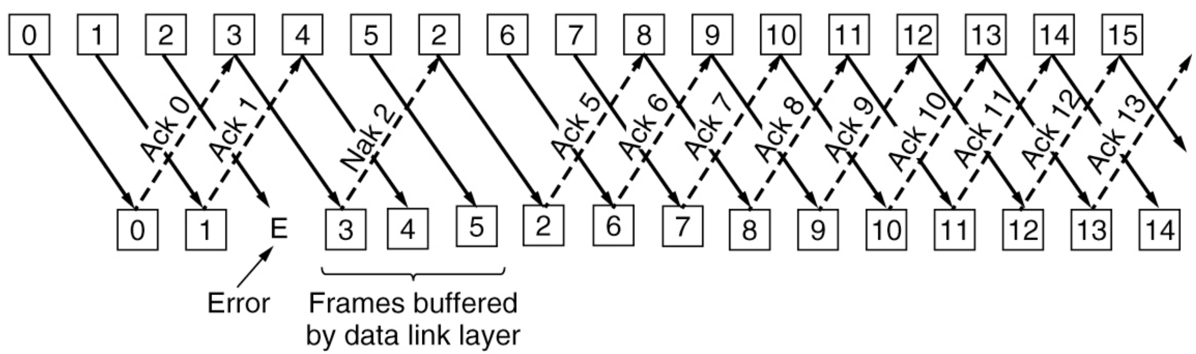
Satellite Link



Link Layer Retransmissions



(a)



(b)

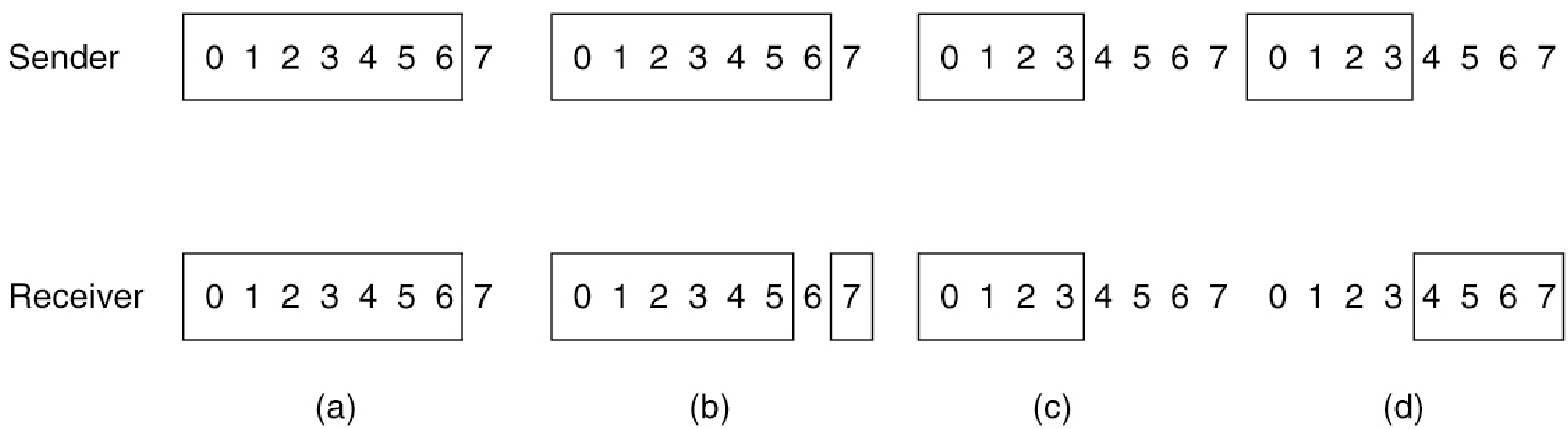
Pipelining and error recovery. Effect of an error when (a) receiver window size is 1 and (b) receiver window size is large.

Acknowledgements

- Positive acknowledgments (ACKs)
  - Cumulative acknowledgment
    - Acknowledge frame *x*: acknowledges frames

      …, *x*-2, *x*-1, *x*
  - Selective acknowledgment
    - Acknowledge only frame *x*
- Negative acknowledgments (NAKs)
  - Report that frame *x* is corrupted or missing

Examples



(a) Initial situation with a window of size 7. (b) After 7 frames have been sent and received but not acknowledged. (c) Initial situation with a window size of 4. (d) After 4 frames have been sent and received but not acknowledged.

Maximum Window Size

- *n* bit sequence numbers
  - 0 to 2^n-1
- The maximum window size is
  - GBN: 2^n-1
  - SR: 2^{n-1}

Point-to-Point Protocol

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

PPP frame format for unnumbered mode operation