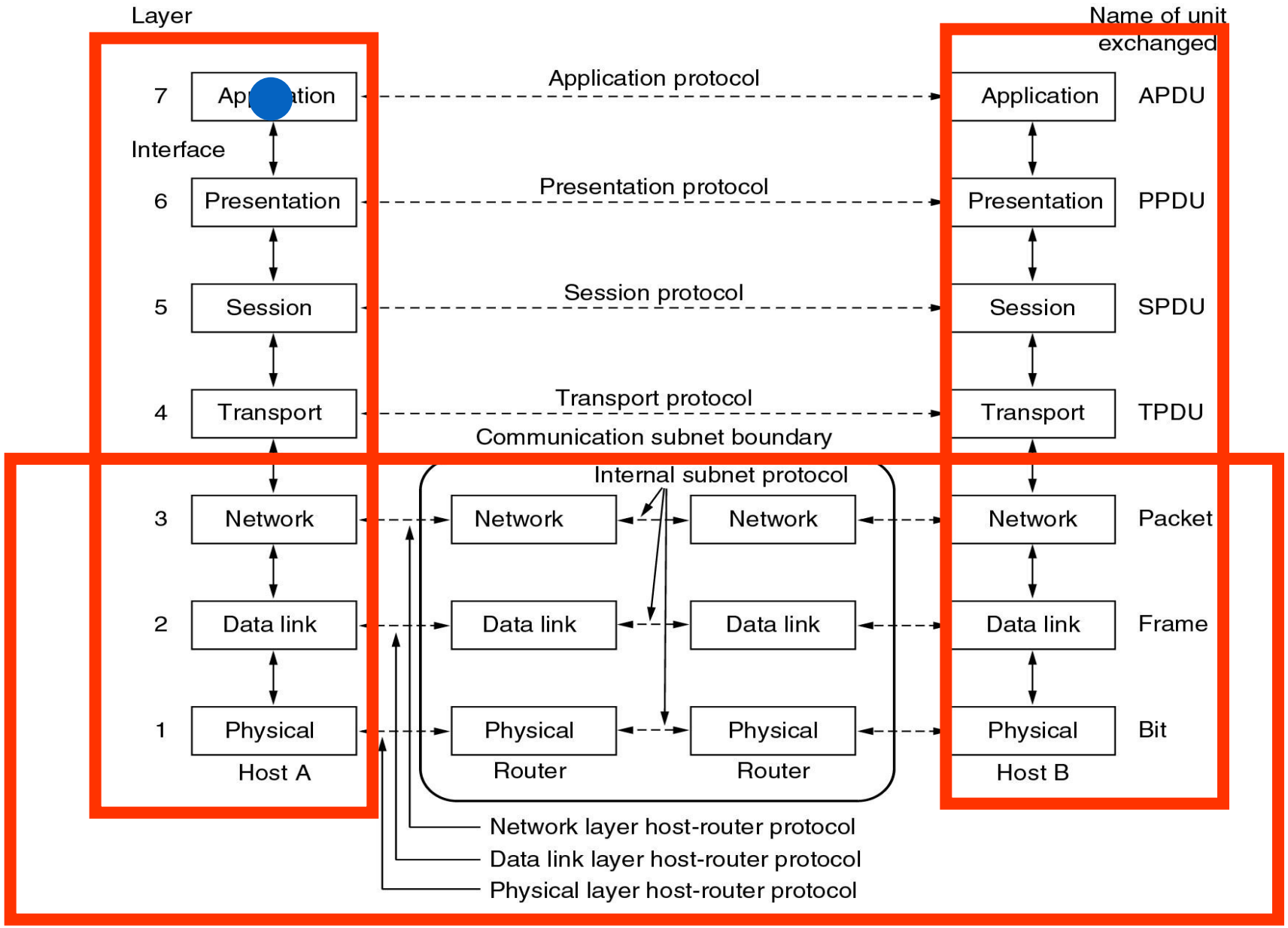
CSC8004 Computer Networks

The Data Link Layer

Dr Ellis Solaiman

The OSI Reference Model



The Data Link Layer

- Role of the Data Link Layer
- Framing
- Error handling
- Flow control

Role of the Data Link Layer

- The Data link Layer implements reliable communications between **adjacent machines** connected by a private communications channel
- The Data Link Layer is responsible for:
  - Framing: delimiting data units flowing on the channel (may be done by Physical Layer as previously described)
  - Error control: detecting and correcting errors introduced by the Physical Layer
  - Flow control: ensuring slow receiver not swamped by fast sender

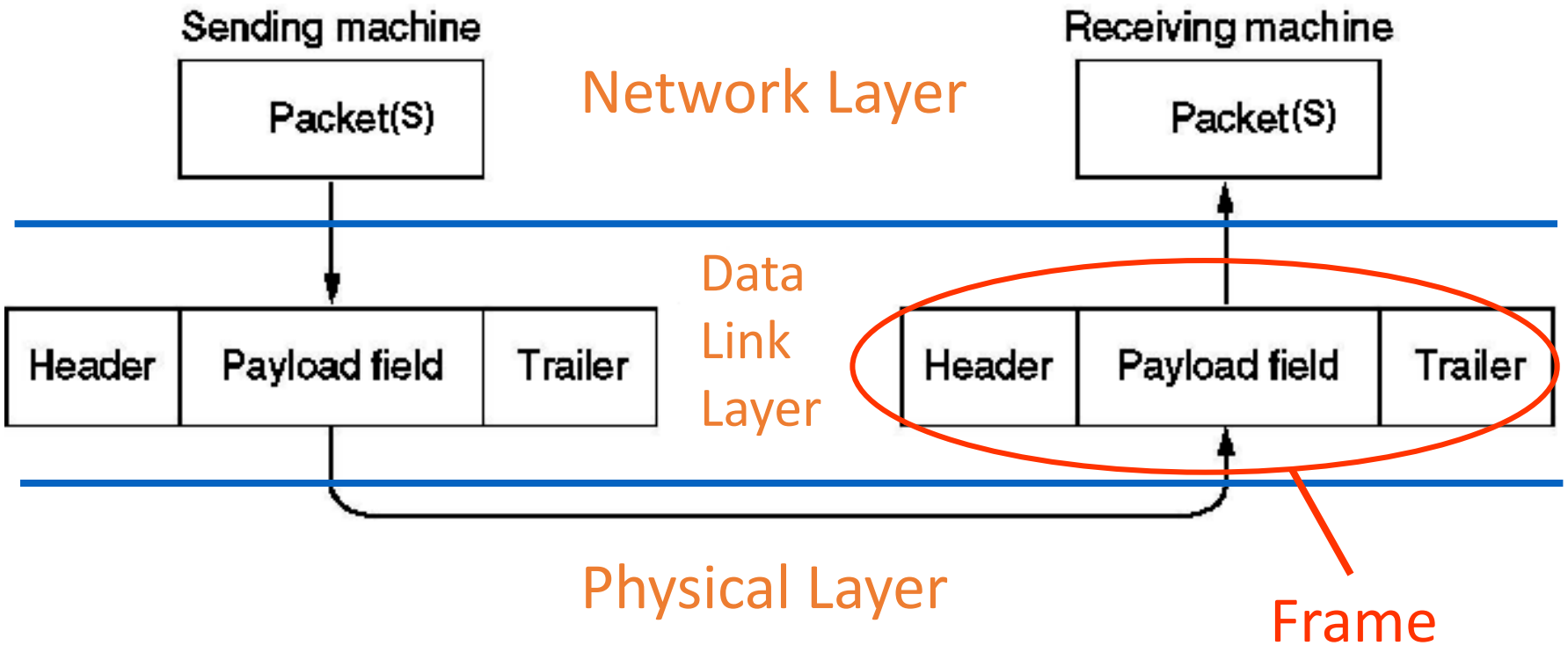
The Data Link Layer

- Role of the Data Link Layer
- Framing
  - Fixed frame size
  - Character count
  - Start and end characters with byte stuffing
  - Start and end flags with bit stuffing
- Error handling
- Flow control

Framing

- The transmitted element in Data Link layer is called a frame
- A frame is regarded either as a block of characters, or as a sequence of bits
- The sender assembles/breaks an incoming sequence of packets (from network layer) into frames
- The receiver must determine the first and last bit/byte of the frame so that the frames can then be extracted
- (Bit synchronization is done by physical layer – remember RZ, Manchester, NRZI?)

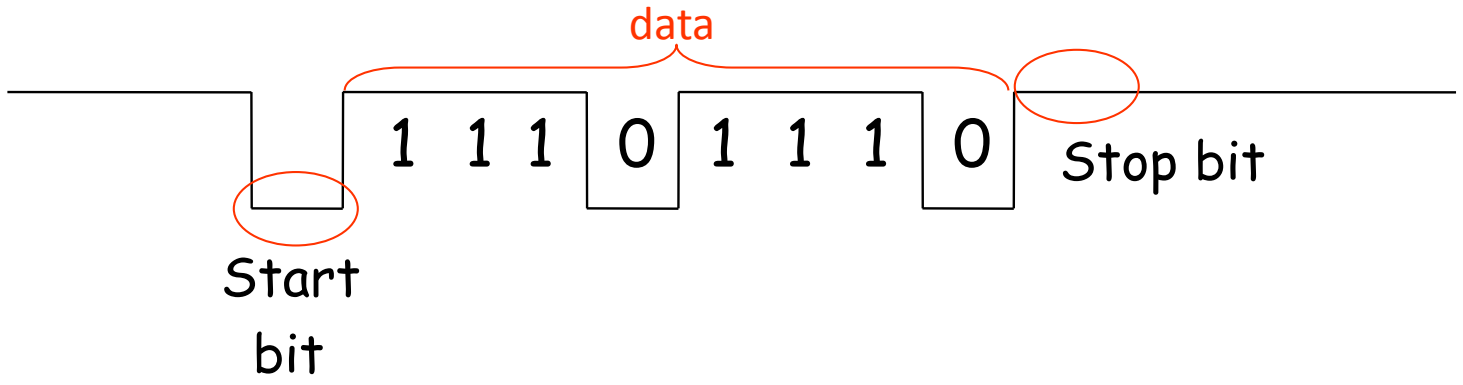
Framing (2)



Relationship between packets and frames

Fixed frame size

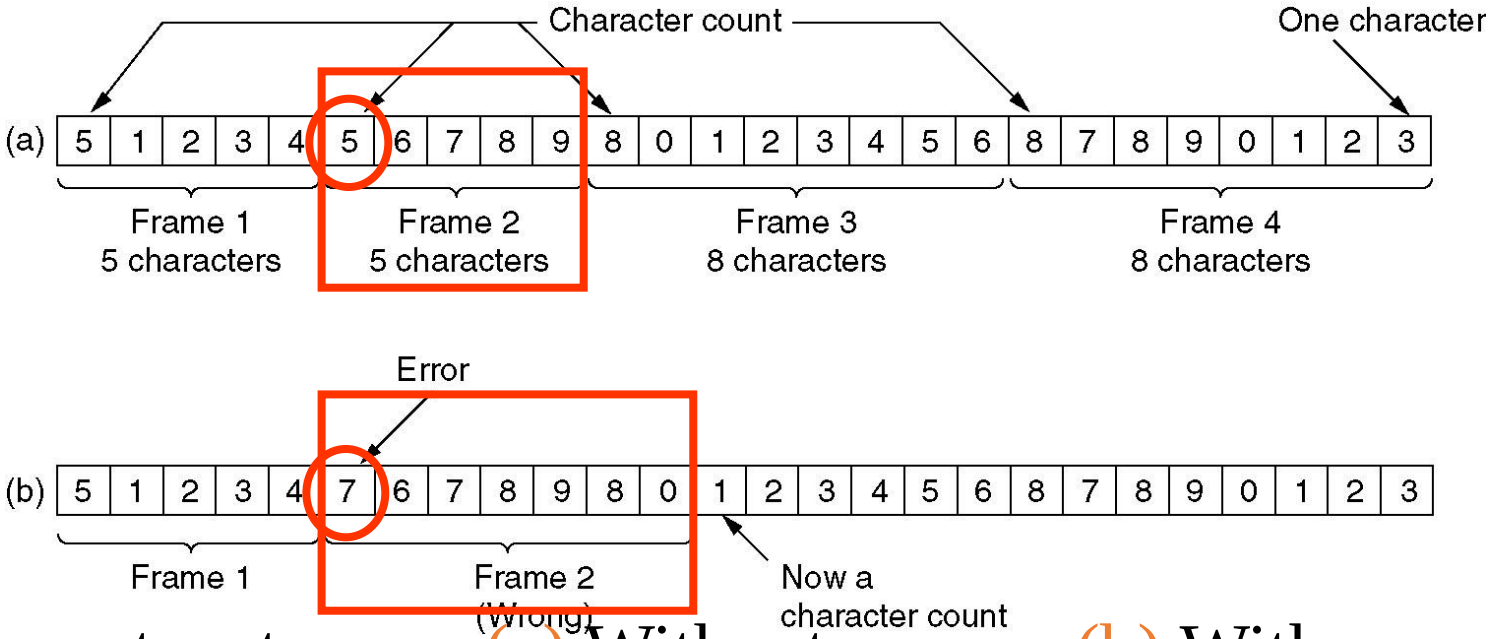
- Payload is transmitted one character at a time. 5 to 8 bits can be accommodated
- Timing only needs maintaining within each character
- There is no common clock at sender and receiver
- Synchronisation takes place with each character through start and stop bits
- This is method used on computer serial cables (eg to RS232 from computer to modem)



Character count

Payload is transmitted as a variable length block of characters preceded by a character count

Count can be corrupted after which it is impossible to determine frame boundaries



A character stream (a) Without errors (b) With one error

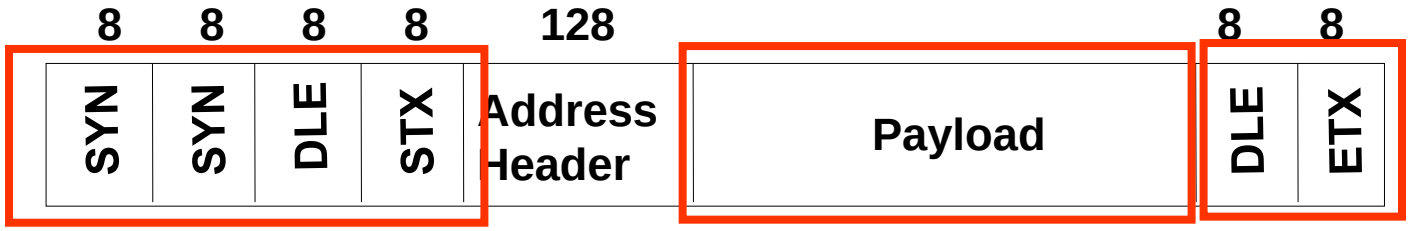
ASCII character codes

ASCII – American Standard Code for Information Interchange

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

http://www.nthelp.com/ascii.htm

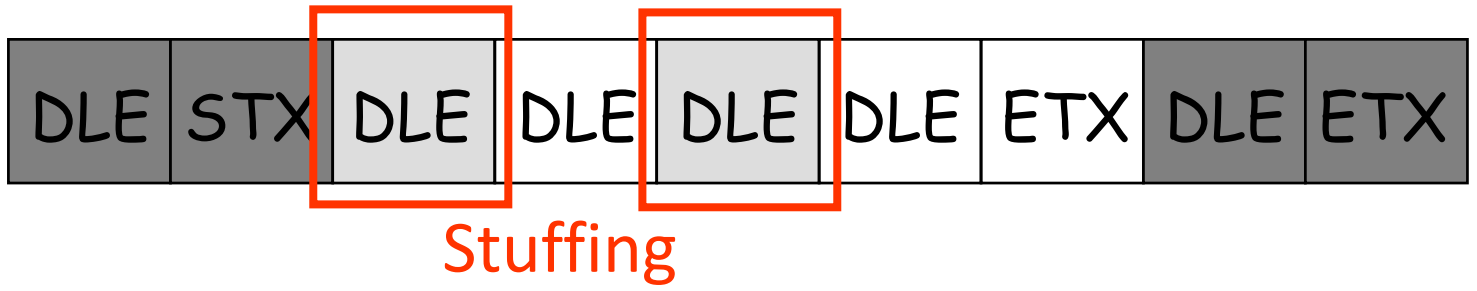
Start and end characters…



- Payload transmitted as a variable length block of characters
- The start of each frame is indicated by a preamble
  - SYN,SYN,DLE,STX
- The end of each frame is indicated by a postamble
  - DLE,ETX
- This is more efficient than fixed frame size transmission
- Problem: the DLE ETX character sequence might appear in the payload of the frame. This would cause the protocol software to assume (incorrectly) that the frame was finished at that point
- This is known as the data transparency problem

... with byte stuffing

- Solution: send an extra DLE character before each DLE character in the payload
- You then regard a payload DLE DLE as a single payload DLE
- This is known as byte stuffing.
- Example: how is the data sequence DLE DLE ETX sent?



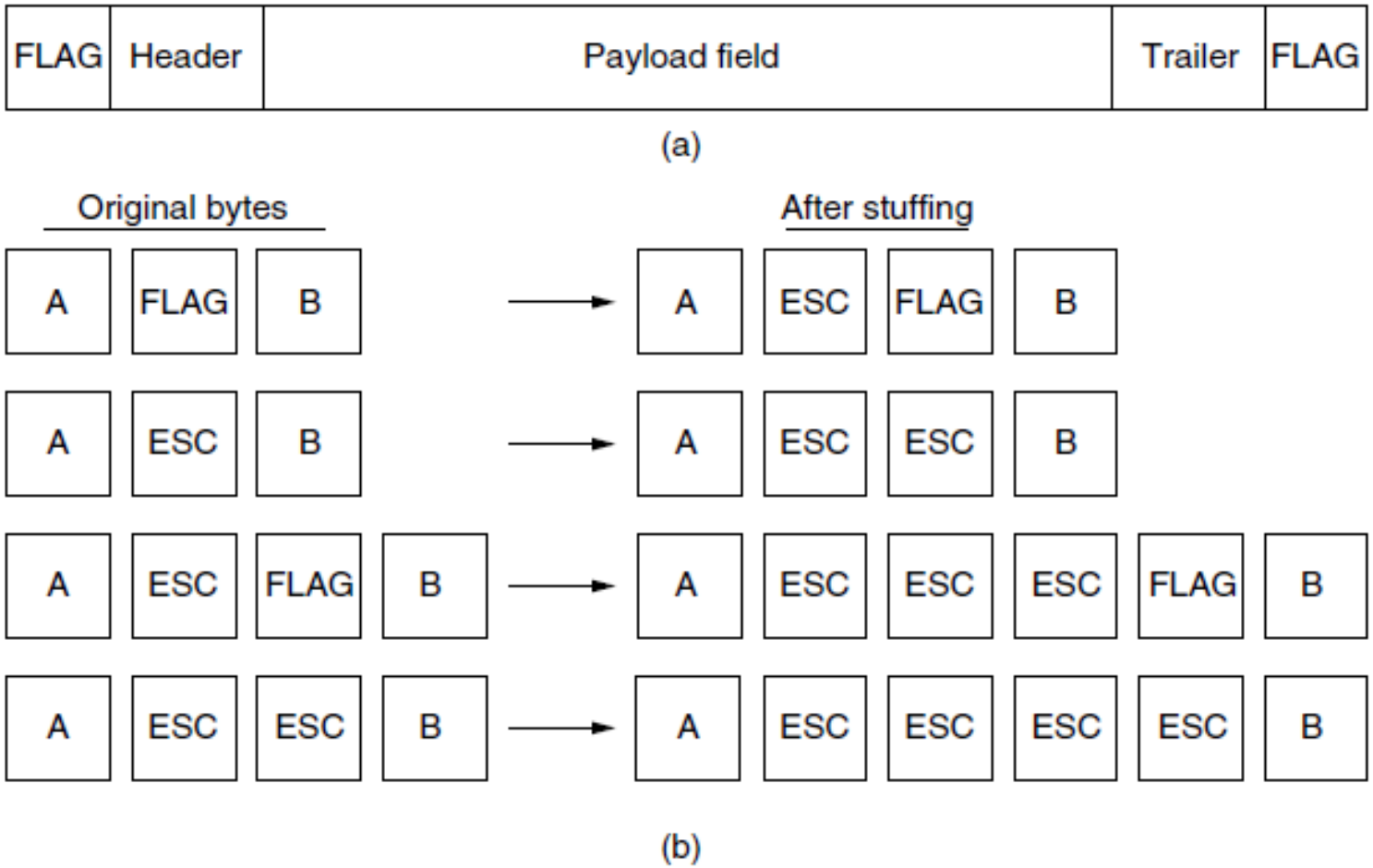


Figure 3-4. (a) A frame delimited by flag bytes. (b) Four examples of byte sequences before and after byte stuffing.

Start and end flags with bit stuffing

- End a frame with a special bit-sequence: 01111110
- Bit Stuffing is now required for transparency:
- Sender: if five consecutive 1s have been transmitted in the body of the message, insert a 0
- Receiver: should five consecutive 1s arrive, look at the next bit(s):
  - if the next bit is a 0: remove it
  - if the next bits are 10: found end-of-frame marker
  - if the next bits are 11: this is an error

Bit stuffing example

Original payload: 00111111100001111100

Stuffed payload: 00111110110000111110100

Received frame: 00111110110000111110100011111110

- Sender: if five consecutive 1s have been transmitted in the body of the message, insert a 0
- Receiver: should five consecutive 1s arrive, look at the next bit(s):
  - if the next bit is a 0: remove it
  - if the next bits are 10: found end-of-frame marker
  - if the next bits are 11: this is an error

The following character encoding is used in a data link protocol:
A: 01000111 B: 11100011 FLAG: 01111110 ESC: 11100000
Show the bit sequence transmitted (in binary) for the four-character frame A B ESC FLAG when each of the following framing methods is used:

(a) Byte count.
(b) Flag bytes with byte stuffing.
(c) Starting and ending flag bytes with bit stuffing.

(a) 00000101 01000111 11100011 11100000 01111110

(b) 01111110 01000111 11100011 11100000 11100000 11100000 01111110 01111110

(c) 01111110 01000111 110100011 111000000 011111010 01111110

The following data fragment occurs in the middle of a data stream for which the byte-stuffing algorithm described in the text is used: A B ESC C ESC FLAG FLAG D. What is the output after stuffing?

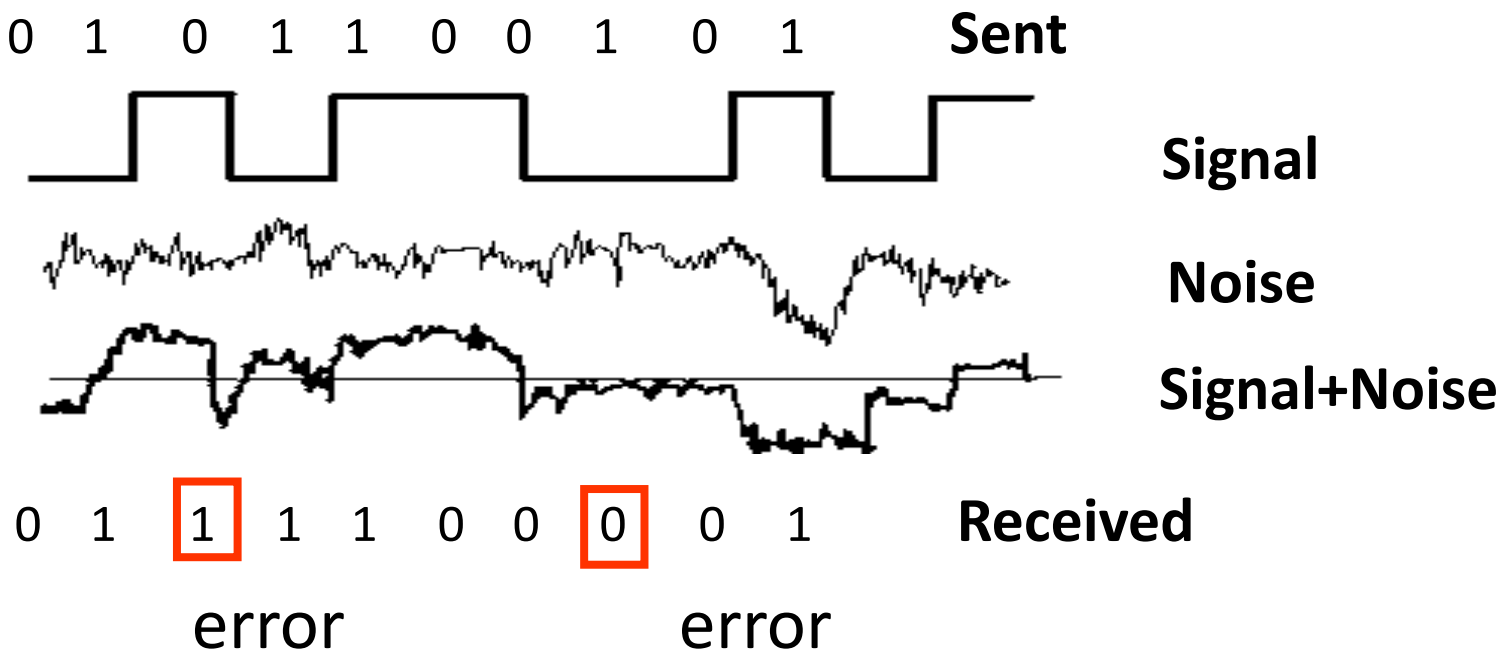
After stuffing, we get A B ESC ESC C ESC ESC ESC FLAG ESC FLAG D.

Data Link Layer

- Role of the Data Link Layer
- Framing
- Error handling
  - Redundant data
  - Parity bit
- Flow control

Error handling

Whenever signals are transmitted over long distances, the signal weakens, and noise is introduced. The receiver cannot resolve the signal levels. This introduces transmission errors.



Redundant data

- Redundant data is added to the message to allow:
  - detection and correction at destination (error correction)
  - detection and subsequent retransmission (error detection)
- Error correction requires more redundancy and depends on assumptions made about types of errors.
- A missing message will defeat any level of redundancy!
- Error correction sometimes used for poor links
- Must be supplemented by error detection and retransmission
- Can occur concurrently at more than one ISO layer

Parity bit

- Consider a byte transmitted without redundant data
  - 256 possible values 00000000 to 11111111

- Now consider the addition of a parity bit
  - 512 possible values 00000000x to 11111111x
  - X is chosen so that the number of 1's is even (or odd)
  - 256 valid values
  - Can detect a single bit error

Checksum

- A checksum performs a mathematical calculation on a sequence of numbers within the data frame, and records the result in the header.
- The recipient will repeat the calculation and compare the result with the transmitted value.

Data Link Layer

- Framing
- Error handling
- Flow control
  - PAR protocol
  - Sliding window protocols

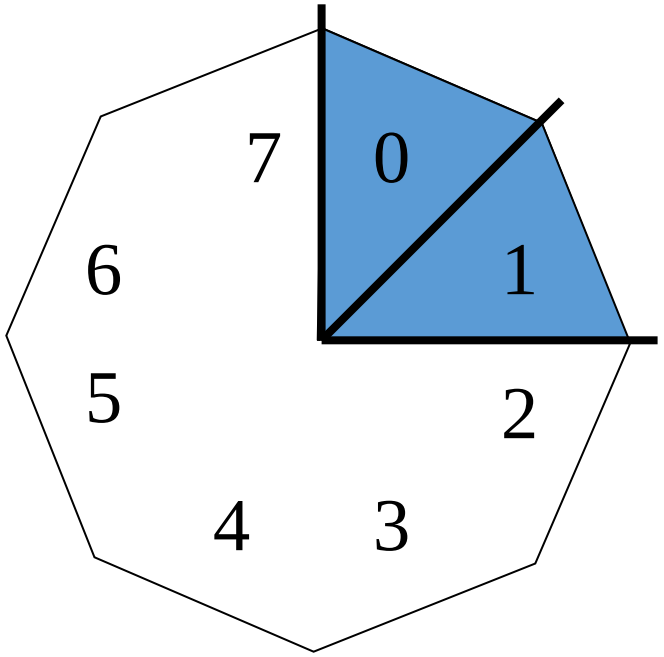
PAR Protocol

- The Positive Acknowledgement/Retransmission protocol (PAR) works under these assumptions:
  - the channel may damage or lose frames. One-bit sequence numbers are used (0 and 1, so they go 0, 1, 0, 1, 0, ……).
- Protocol operation:
  - A sends a frame, then waits for B to acknowledge.
  - If a timeout occurs, the frame is retransmitted.
  - Duplicates are rejected by B. In addition, B also rejects frames with errors.

$$best.utilization = \frac{time.to.transmit.\ frame}{time.to.transmit.\ frame + round.trip.time}$$

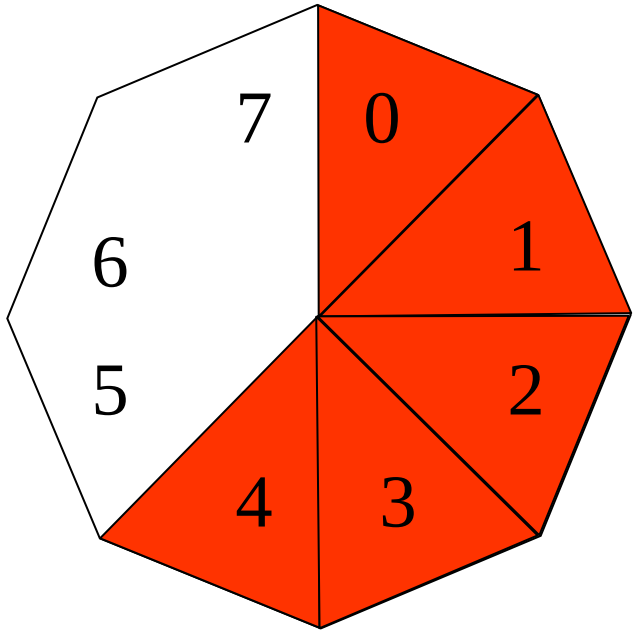
Sliding window protocols

- Sliding windows at sender and receiver
  - Sender window: messages sent but not acknowledged
  - Receiver window: messages that will be accepted



Sender

Send message 0
Send message 1
Receive message 0
Send ack 0
Receive ack 0
Receive message 1
Send ack 1
Receive ack 1



Receiver

Sliding window protocols (2)

- The sender's window and the receiver's window need not have the same lower and upper limits, or even have the same size.
- At the sender side:
  - initially, the window size is zero (the minimum)
  - any new frame sent will get the next highest sequence number. The window's upper edge is advanced by one
  - when an acknowledgement comes in, the lower edge is advanced by one
  - the sender needs to buffer all unacknowledged frames
  - stop transmission if the window is at its maximum size

Sliding window protocols (3)

- At the receiver side:
  - discard any frames outside the window
  - rotate the window by one when a frame is received and matches the lower edge. The acknowledgement is then sent
  - initially, the receiver's window is at its maximum size (the opposite of the sender's window)
- Any host will usually be both sender and receiver simultaneously. However the sender's window and receiver's window on the same device are completely independent of each other.