The Data Link Layer

Chapter 3

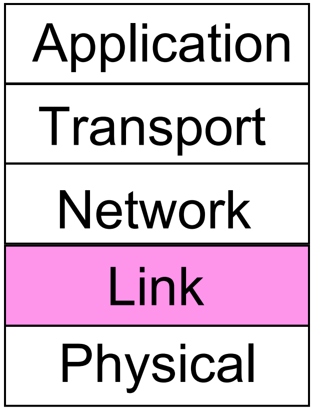
- Data Link Layer Design Issues
- Error Detection and Correction
- Elementary Data Link Protocols
- Sliding Window Protocols
- Example Data Link Protocols

Revised: August 2011

The Data Link Layer

Responsible for delivering frames of information over a single link

- Handles transmission errors and regulates the flow of data

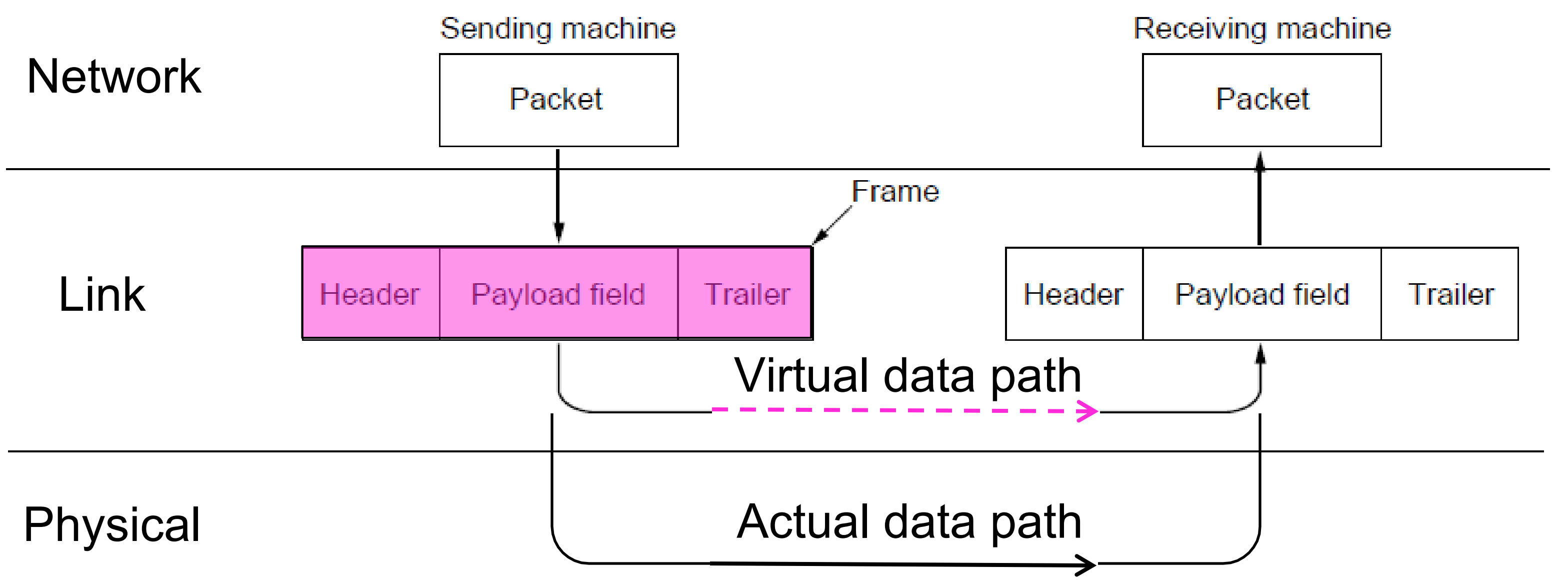


Data Link Layer Design Issues

- Frames »
- Possible services »
- Framing methods »
- Error control »
- Flow control »

Frames

Link layer accepts packets from the network layer, and encapsulates them into frames that it sends using the physical layer; reception is the opposite process



Possible Services

Unacknowledged connectionless service

- Frame is sent with no connection / error recovery
- Ethernet is example

Acknowledged connectionless service

- Frame is sent with retransmissions if needed
- Example is 802.11

Acknowledged connection-oriented service

- Connection is set up; rare

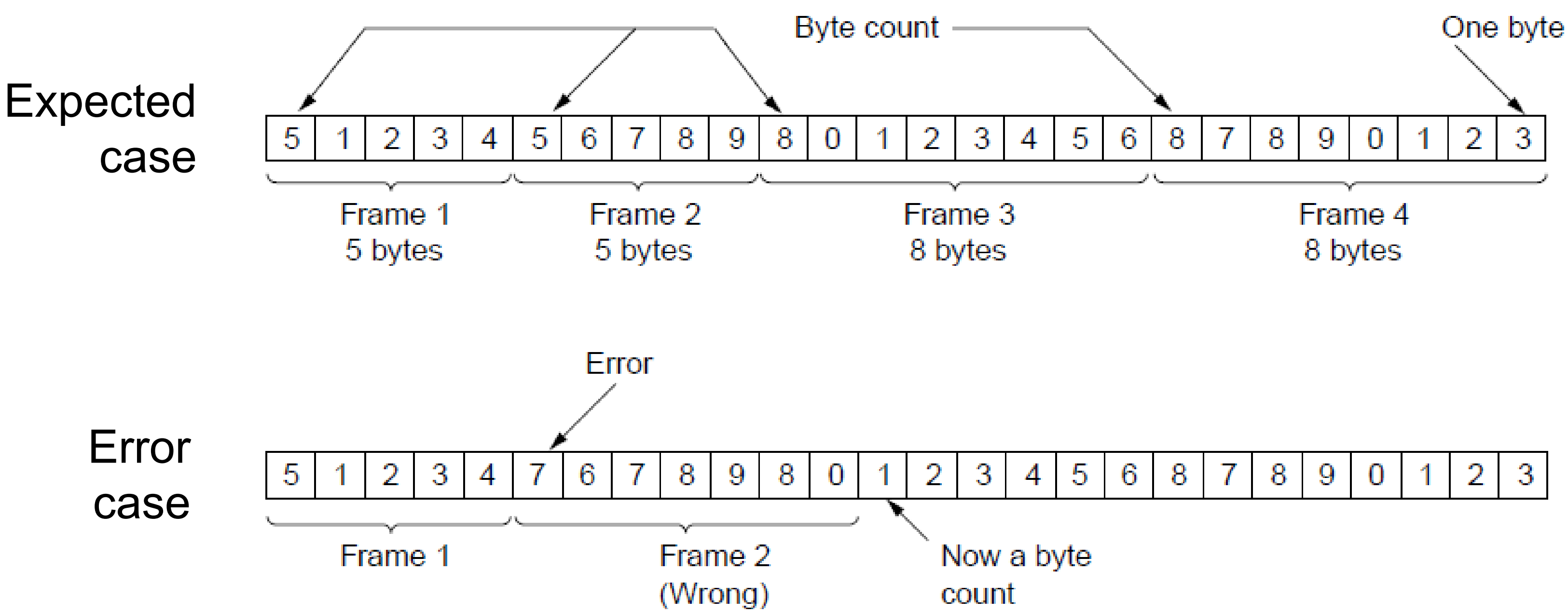
Framing Methods

- Byte count »
- Flag bytes with byte stuffing »
- Flag bits with bit stuffing »
- Physical layer coding violations
  - Use non-data symbol to indicate frame

Framing – Byte count

Frame begins with a count of the number of bytes in it

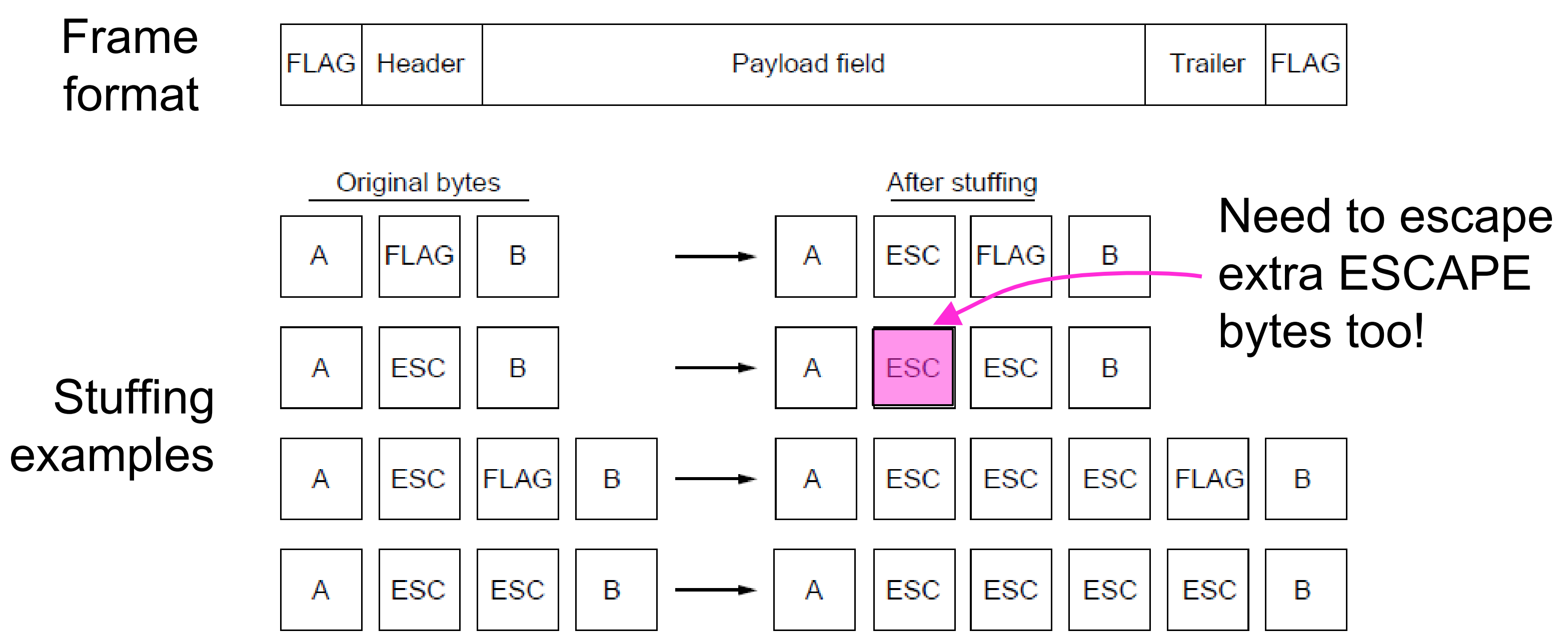
- Simple, but difficult to resynchronize after an error



Framing – Byte stuffing

Special flag bytes delimit frames; occurrences of flags in the data must be stuffed (escaped)

- Longer, but easy to resynchronize after error



Framing – Bit stuffing

Stuffing done at the bit level:

- Frame flag has six consecutive 1s (not shown)
- On transmit, after five 1s in the data, a 0 is added
- On receive, a 0 after five 1s is deleted

Data bits	0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
Transmitted bits with stuffing	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

Error Control

Error control repairs frames that are received in error

- Requires errors to be detected at the receiver
- Typically retransmit the unacknowledged frames
- Timer protects against lost acknowledgements

Detecting errors and retransmissions are next topics.

Flow Control

Prevents a fast sender from out-pacing a slow receiver

- Receiver gives feedback on the data it can accept
- Rare in the Link layer as NICs run at “wire speed”
  - Receiver can take data as fast as it can be sent

Flow control is a topic in the Link and Transport layers.

Error Detection and Correction

Error codes add structured redundancy to data so errors can be either detected, or corrected.

Error correction codes:

- Hamming codes »
- Binary convolutional codes »
- Reed-Solomon and Low-Density Parity Check codes
  - Mathematically complex, widely used in real systems

Error detection codes:

- Parity »
- Checksums »
- Cyclic redundancy codes »

Error Bounds – Hamming distance

Code turns data of n bits into codewords of n+k bits

Hamming distance is the minimum bit flips to turn one valid codeword into any other valid one.

- Example with 4 codewords of 10 bits (n=2, k=8):
  - 0000000000, 0000011111, 1111100000, and 1111111111
  - Hamming distance is 5

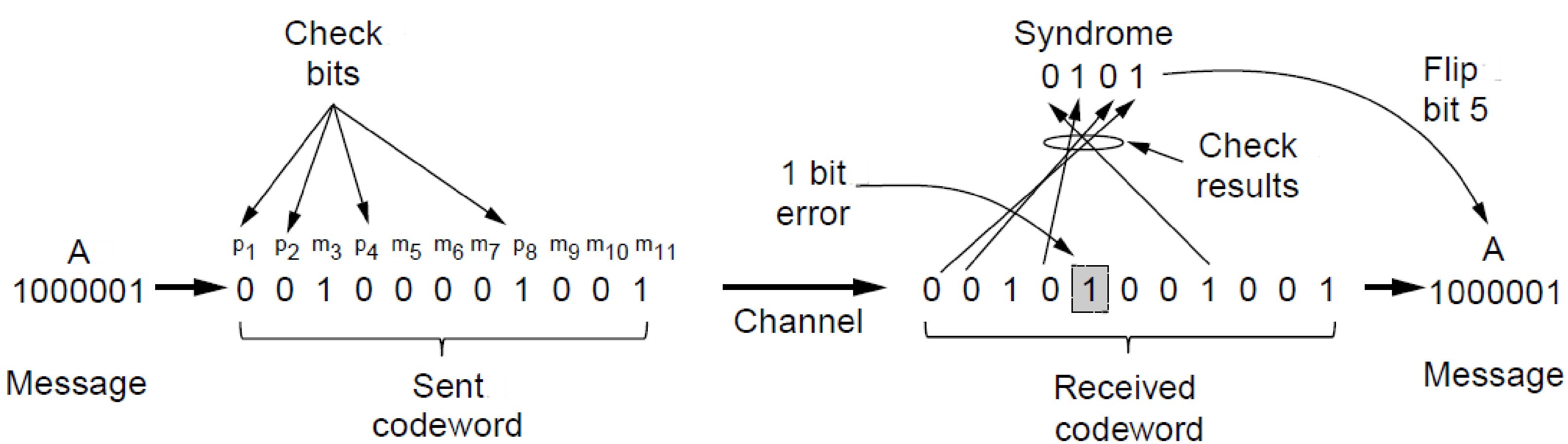
Bounds for a code with distance:

- 2d+1 – can correct d errors (e.g., 2 errors above)
- d+1 – can detect d errors (e.g., 4 errors above)

Error Correction – Hamming code

Hamming code gives a simple way to add check bits and correct up to a single bit error:

- Check bits are parity over subsets of the codeword
- Recomputing the parity sums (<u>syndrome</u>) gives the position of the error to flip, or 0 if there is no error

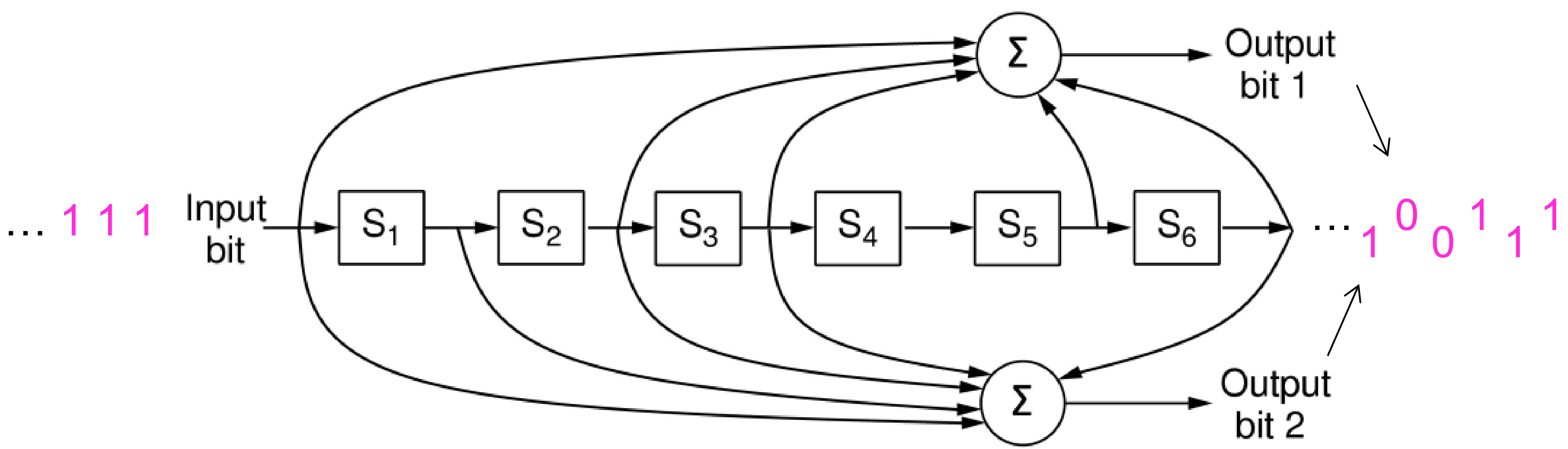


(11, 7) Hamming code adds 4 check bits and can correct 1 error

Error Correction – Convolutional codes

Operates on a stream of bits, keeping internal state

- Output stream is a function of all preceding input bits
- Bits are decoded with the Viterbi algorithm



Popular NASA binary convolutional code (rate = ½) used in 802.11

Error Detection – Parity (1)

Parity bit is added as the modulo 2 sum of data bits

- Equivalent to XOR; this is even parity
- Ex: 1110000 → 11100001
- Detection checks if the sum is wrong (an error)

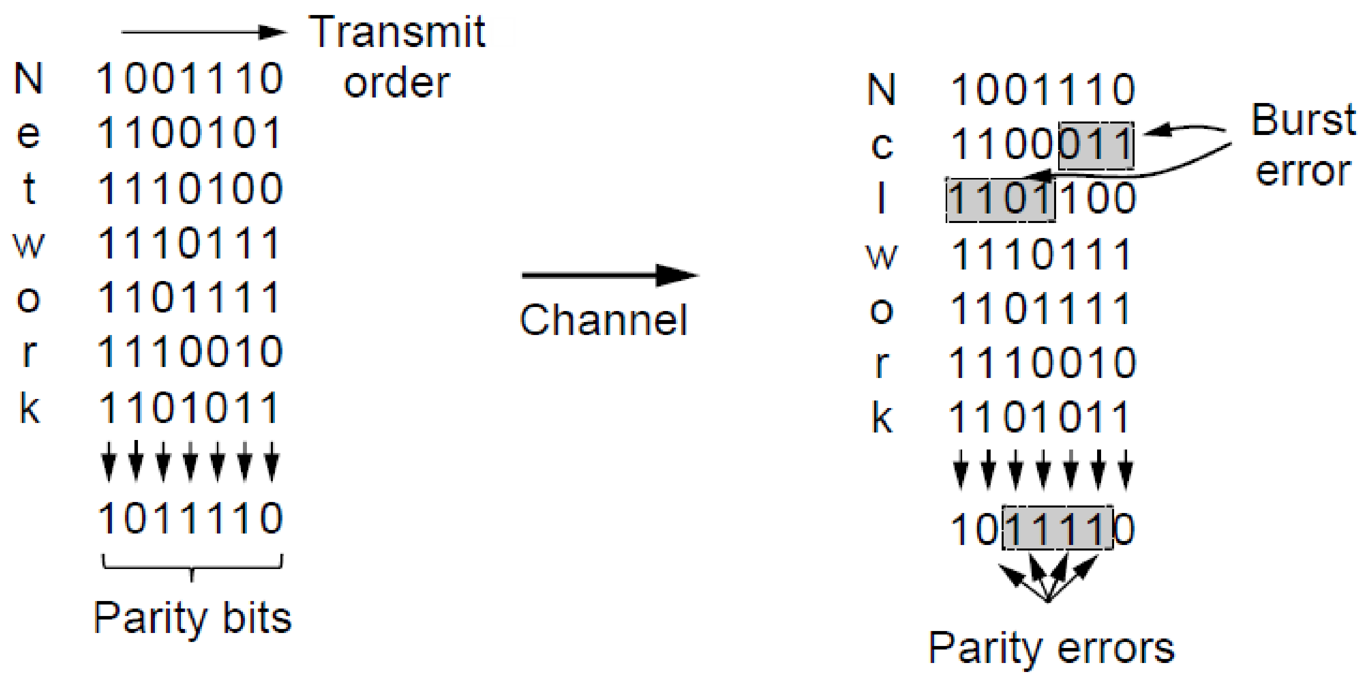
Simple way to detect an *odd* number of errors

- Ex: 1 error, 11100101; detected, sum is wrong
- Ex: 3 errors, 11011001; detected sum is wrong
- Ex: 2 errors, 11101101; *not detected*, sum is right!
- Error can also be in the parity bit itself
- Random errors are detected with probability ½

Error Detection – Parity (2)

<u>Interleaving</u> of N parity bits detects burst errors up to N

- Each parity sum is made over non-adjacent bits
- An even burst of up to N errors will not cause it to fail



Error Detection – Checksums

Checksum treats data as N-bit words and adds N check bits that are the modulo 2^N sum of the words

- Ex: Internet 16-bit 1s complement checksum

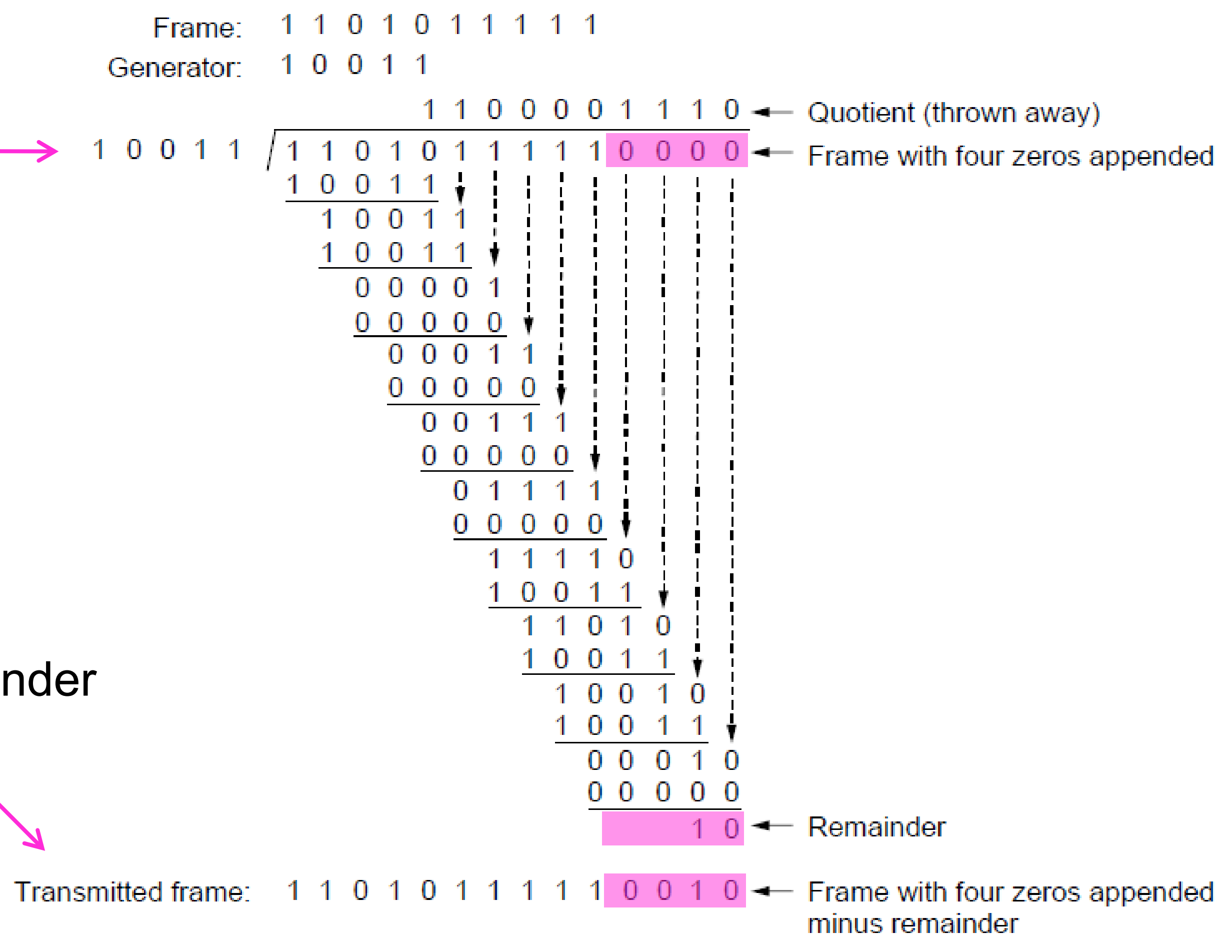
Properties:

- Improved error detection over parity bits
- Detects bursts up to N errors
- Detects random errors with probability $1-2^N$
- Vulnerable to systematic errors, e.g., added zeros

Error Detection – CRCs (1)

Adds bits so that transmitted frame viewed as a polynomial is evenly divisible by a generator polynomial

Start by adding 0s to frame and try dividing



Offset by any reminder to make it evenly divisible

Error Detection – CRCs (2)

Based on standard polynomials:

- Ex: Ethernet 32-bit CRC is defined by:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$$

- Computed with simple shift/XOR circuits

Stronger detection than checksums:

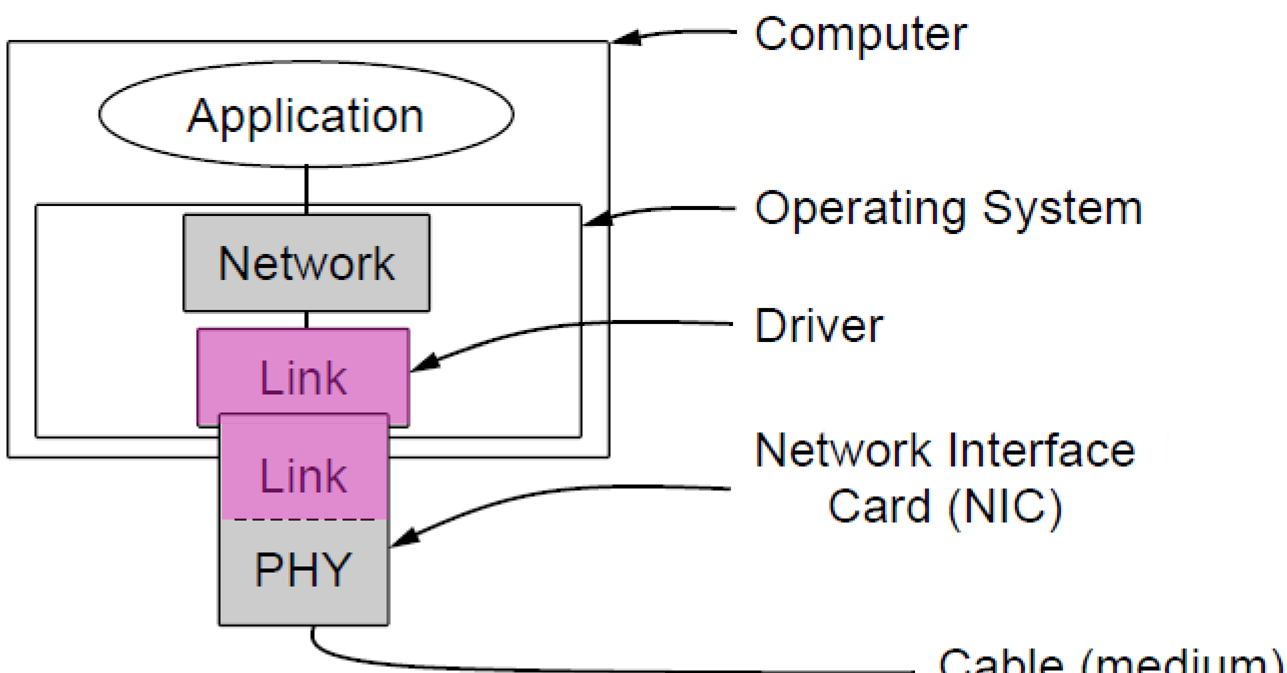
- E.g., can detect all double bit errors
- Not vulnerable to systematic errors

Elementary Data Link Protocols

- Link layer environment »
- Utopian Simplex Protocol »
- Stop-and-Wait Protocol for Error-free channel »
- Stop-and-Wait Protocol for Noisy channel »

Link layer environment (1)

Commonly implemented as NICs and OS drivers; network layer (IP) is often OS software



Link layer environment (2)

Link layer protocol implementations use library functions

- See code (`protocol.h`) for more details

Group	Library Function	Description
Network layer	from_network_layer(&packet) to_network_layer(&packet) enable_network_layer() disable_network_layer()	Take a packet from network layer to send Deliver a received packet to network layer Let network cause “ready” events Prevent network “ready” events
Physical layer	from_physical_layer(&frame) to_physical_layer(&frame)	Get an incoming frame from physical layer Pass an outgoing frame to physical layer
Events & timers	wait_for_event(&event) start_timer(seq_nr) stop_timer(seq_nr) start_ack_timer() stop_ack_timer()	Wait for a packet / frame / timer event Start a countdown timer running Stop a countdown timer from running Start the ACK countdown timer Stop the ACK countdown timer

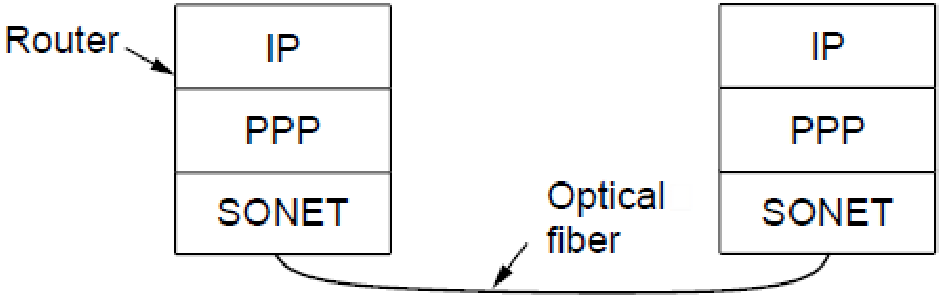
Example Data Link Protocols

- Packet over SONET »
- PPP (Point-to-Point Protocol) »
- ADSL (Asymmetric Digital Subscriber Loop) »

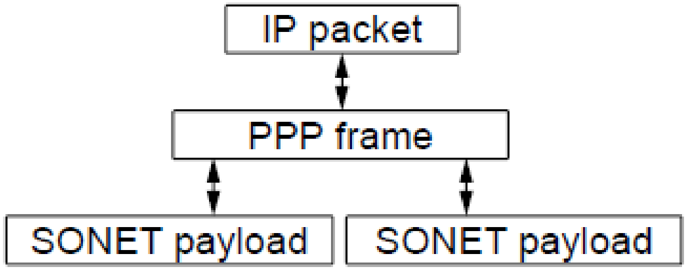
Packet over SONET

Packet over SONET is the method used to carry IP packets over SONET optical fiber links

- Uses PPP (Point-to-Point Protocol) for framing



Protocol stacks

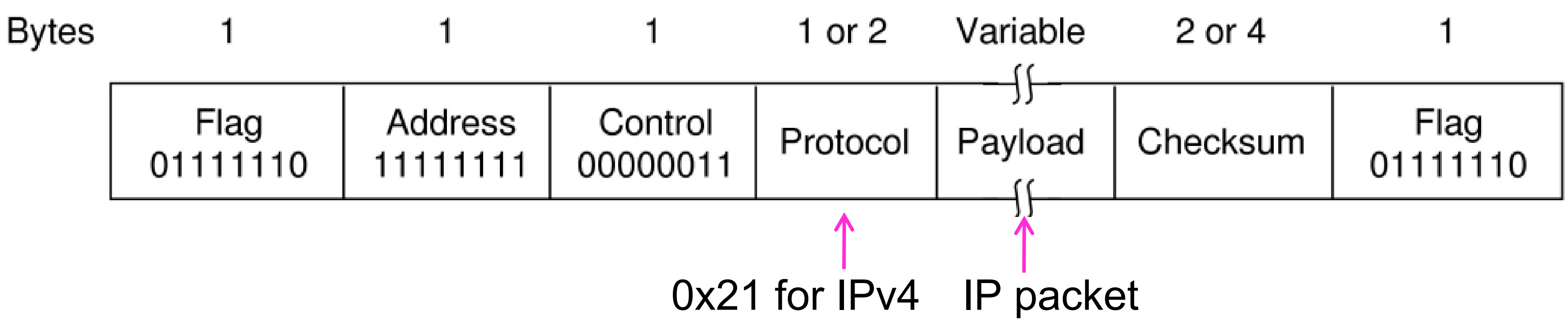


PPP frames may be split over SONET payloads

PPP (1)

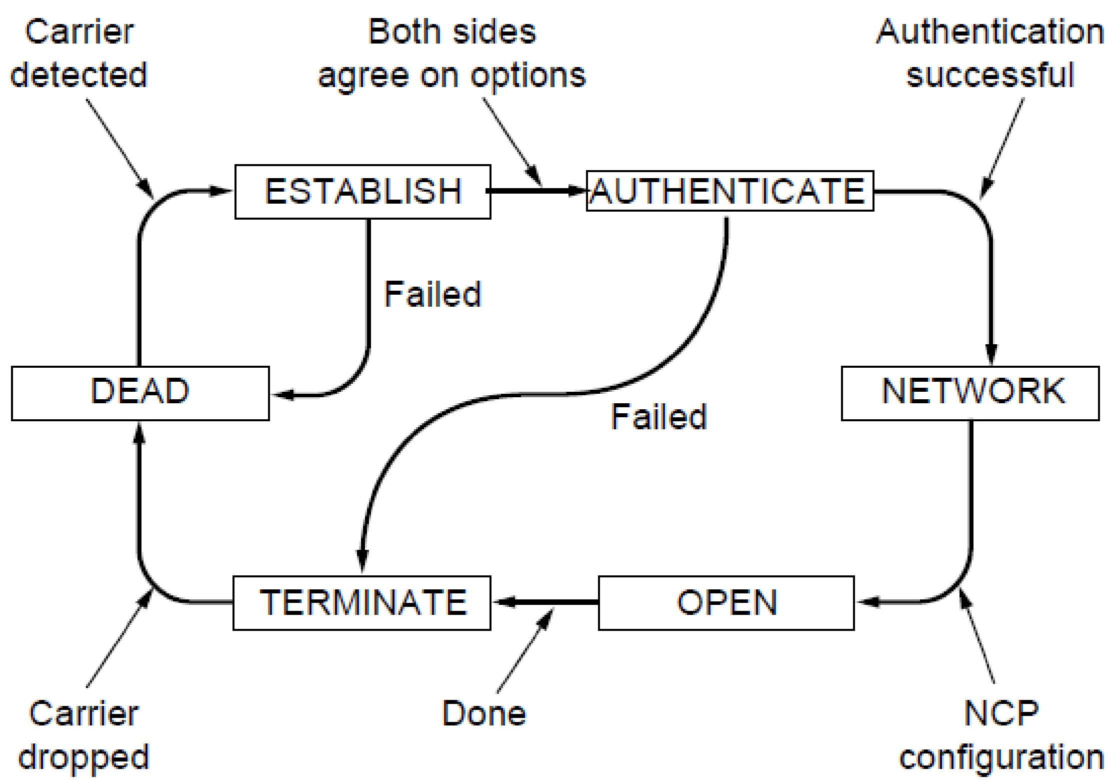
PPP (Point-to-Point Protocol) is a general method for delivering packets across links

- Framing uses a flag (0x7E) and byte stuffing
- “Unnumbered mode” (connectionless unacknow-ledged service) is used to carry IP packets
- Errors are detected with a checksum



PPP (2)

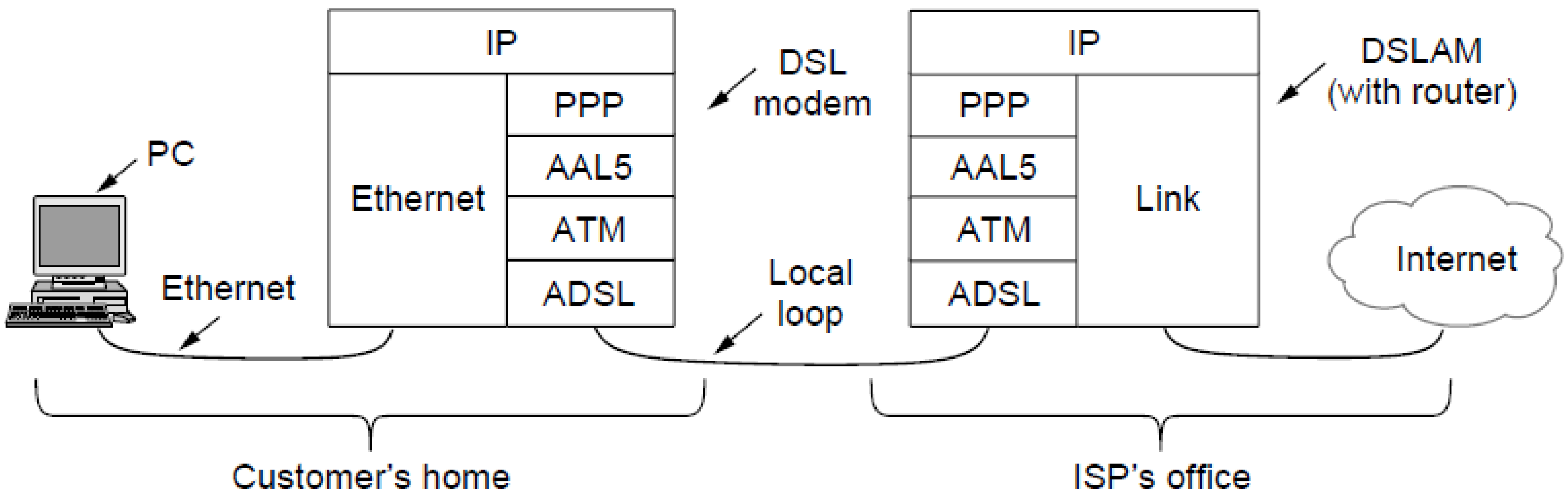
A link control protocol brings the PPP link up/down



State machine for link control

ADSL (1)

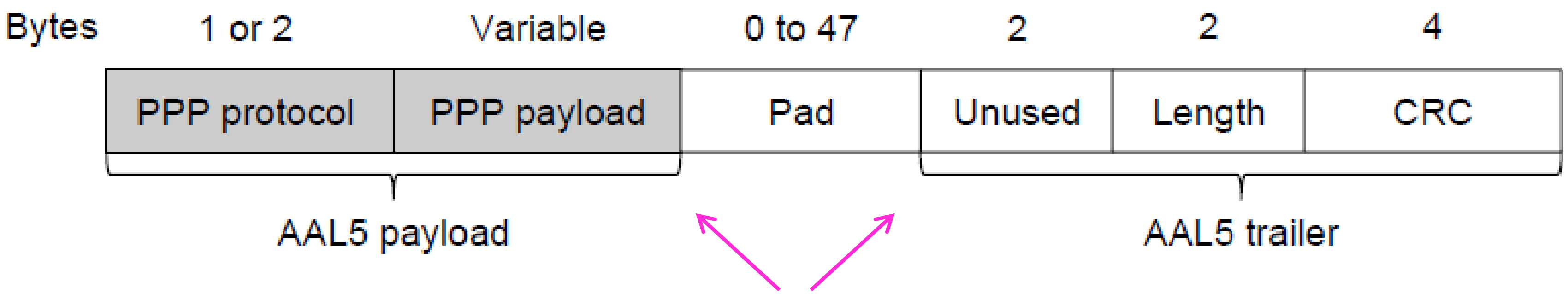
Widely used for broadband Internet over local loops
- ADSL runs from modem (customer) to DSLAM (ISP)
- IP packets are sent over PPP and AAL5/ATM (over)



ADSL (2)

PPP data is sent in AAL5 frames over ATM cells:

- ATM is a link layer that uses short, fixed-size cells (53 bytes); each cell has a virtual circuit identifier
- AAL5 is a format to send packets over ATM
- PPP frame is converted to a AAL5 frame (PPPoA)



AAL5 frame is divided into 48 byte pieces, each of which goes into one ATM cell with 5 header bytes

End

Chapter 3