Introduction to Computer Networks

CS640

Framing and Error Handling

https://pages.cs.wisc.edu/~mgliu/CS640/F22/

Ming Liu
mgliu@cs.wisc.edu

Today

Last lecture

- How to transmit bits reliably and efficiently across the link?

Today

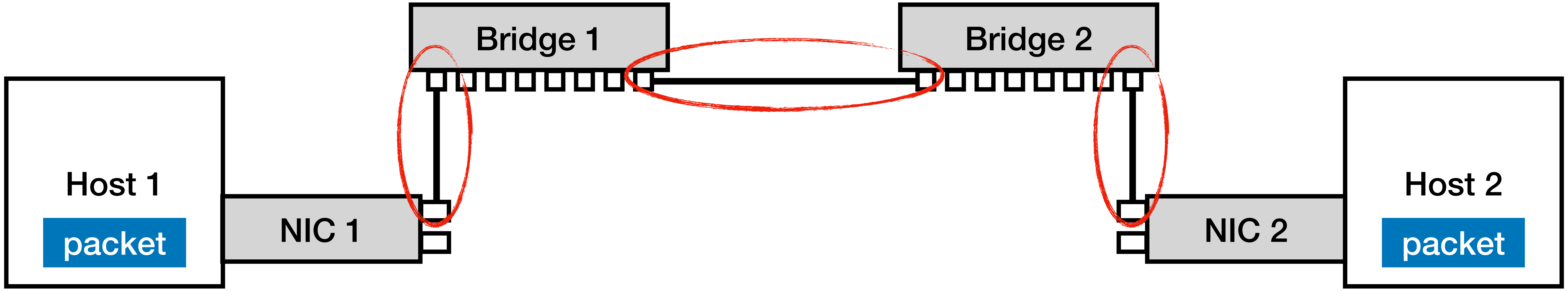
- How to identify a frame from bit streams?
- How to handle transmission errors?

Announcements

- Quiz1 is released
- Lab1 is due today
- Lab2 is released today

Physical layer

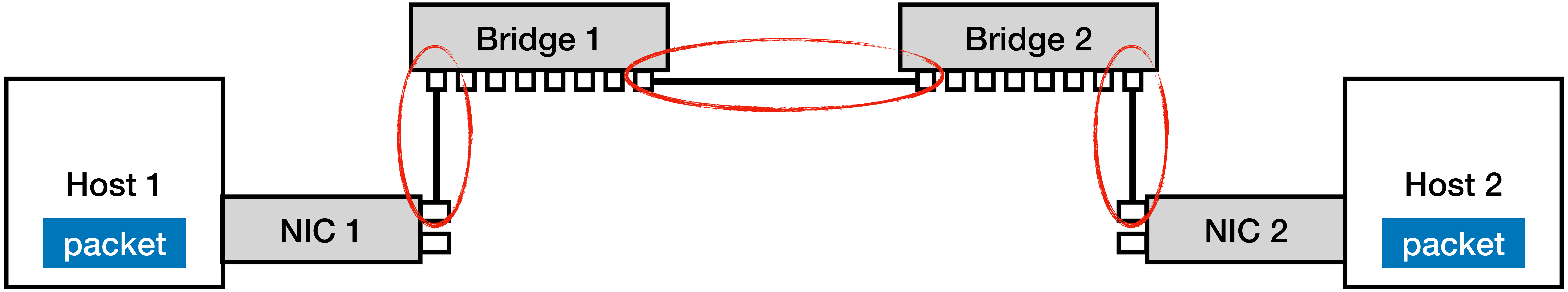
A reliable (and efficient) bit delivery channel over a link



Physical layer

A reliable (and efficient) bit delivery channel over a link

Q: How to transmit a packet reliably between two NICs in a small-scaled network?



Physical layer | A reliable (and efficient) bit delivery channel over a link

Q: How to transmit a packet reliably between two NICs in a small-scaled network?

Q1: How to identify a packet from bit streams? => Framing
Q2: How to handle transmission errors? => Error handling
Q3: How do packets traverse NICs/bridges? => L2 switching
Q4: How to coordinate the NIC on two sides? => Reliable transmission
Q5: How to orchestrate concurrent transmissions? => Access control

frame: a unit of data in the data link layer

Q: How to transmit a ~~packet~~ reliably between two NICs in a small-scaled network?

Q1: How to identify a frame from bit streams? => Framing
Q2: How to handle transmission errors? => Error handling
Q3: How do frames traverse NICs/bridges? => L2 switching
Q4: How to coordinate the NIC on two sides? => Reliable transmission
Q5: How to orchestrate concurrent transmissions? => Access control

Q1: How to identify a frame from bit streams?

Q1: How to identify a frame from bit streams?

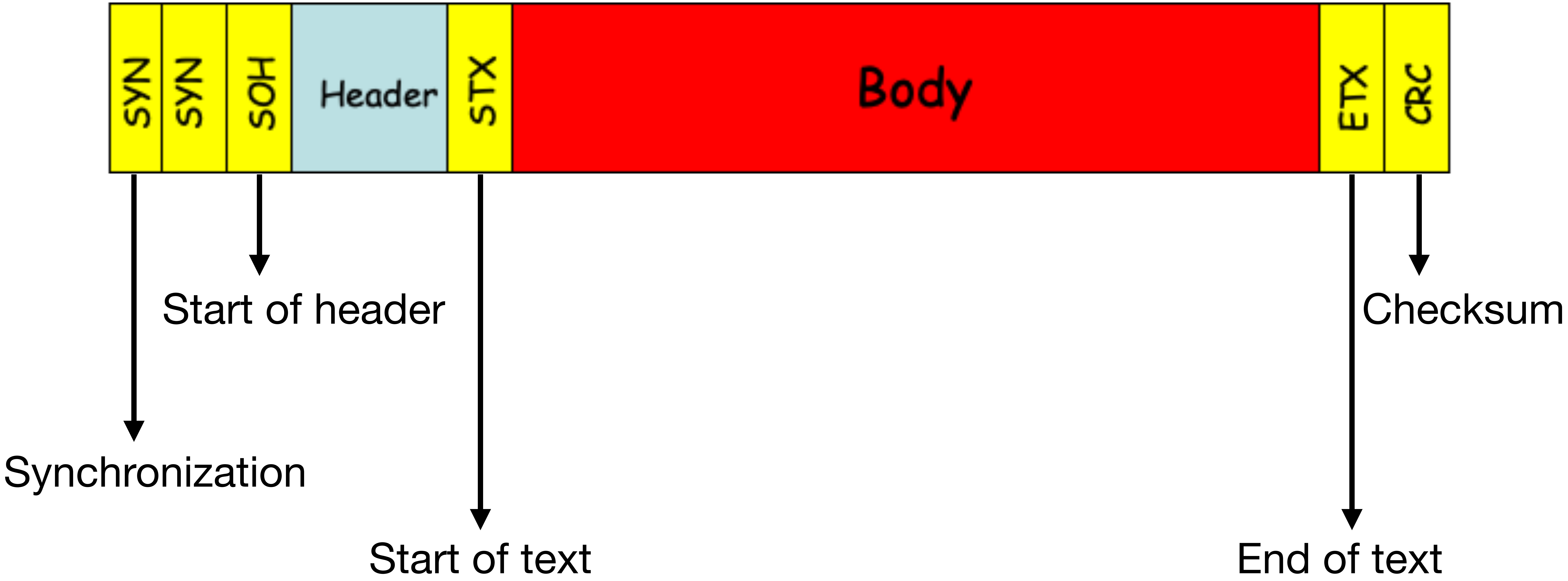
Challenge:

- A frame has the maximum length, but not the fixed length

Q1: How to identify a frame from bit streams?

A: Mark the beginning and end of a frame

Technique #1: Byte Stuffing



Technique #1: Byte Stuffing

Mark the frame with special characters

- Used by the Binary Synchronous Communication (BISYNC) protocol

Technique #1: Byte Stuffing

Mark the frame with special characters

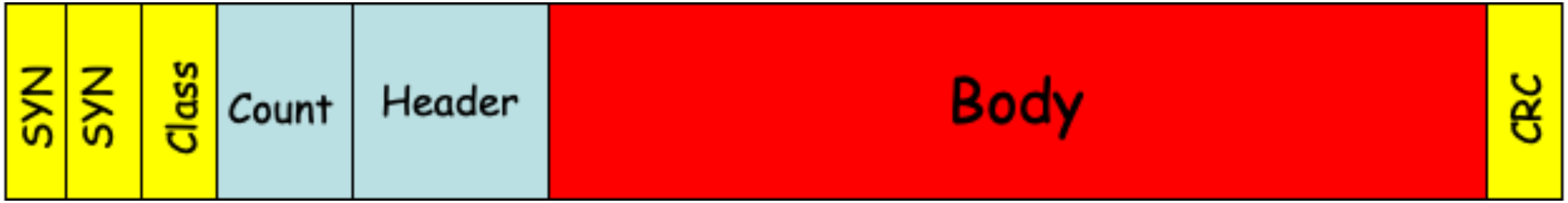
- Used by the Binary Synchronous Communication (BISYNC) protocol

What happens when the user sends a special char?

- Use an escape character (DLE)

Technique #2: Byte Counting

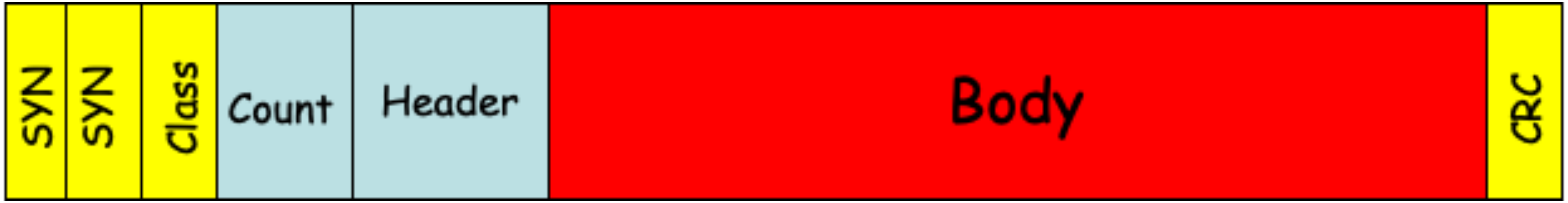
Technique #2: Byte Counting



Encode the number of bytes into a frame

- Used by the Digital Data Communication Message Protocol (DDCMP)

Technique #2: Byte Counting



Encode the number of bytes into a frame

- Used by the Digital Data Communication Message protocol (DDCMP)

Corruptions of the count field may cause frame errors

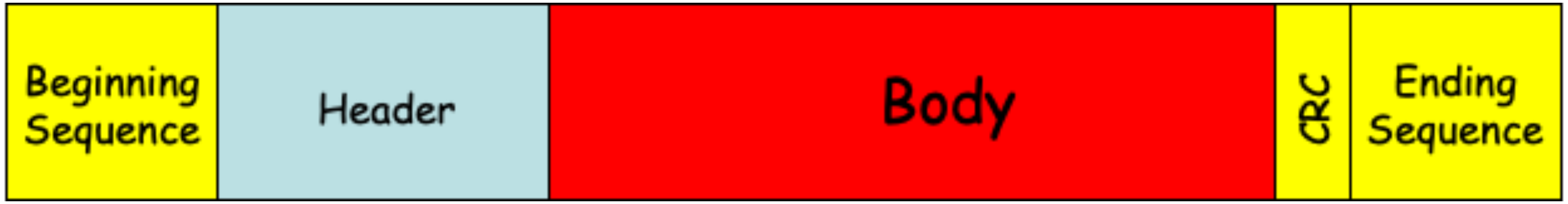
- The receiver requires an error-check mechanism

Technique #3: Bit Stuffing

Start of the frame

End of the frame

Technique #3: Bit Stuffing



Mark the frame with special bit sequences

- Used by High-Level Data Link Control (HDLC) protocol
- #1: The sender/receiver agrees on a special beginning/ending flag: 01111110
- #2: The sender inserts a 0 before transmitting the next bit when transmitting five consecutive 1s
- #3: The receiver unstuffs when receiving five consecutive 1s
  - If the next bit is 1, the end-of-frame marker or an error
  - If the next bit is 0, remove it

Q2: How to handle transmission errors?

Bit flip

Q2: How to handle transmission errors?

Bit flip

Q2: How to handle transmission errors?

A: Data redundancy

- Error detection
- Error correction

Technique #1: Parity bit

Even parity

- Append a parity bit to 7 bits of data to make an even number of 1's
- Odd parity is defined accordingly

1010100	1
1010101	0

Technique #1: Parity bit

Even parity

- Append a parity bit to 7 bits of data to make an even number of 1's
- Odd parity is defined accordingly

1010100 1

1010101 0

Efficacy

- 1 in 8 bits of overhead
- Can detect a single error

1010101 1

1000010 0

Technique #2: 2-D Parity

0110100

1011010

0010110

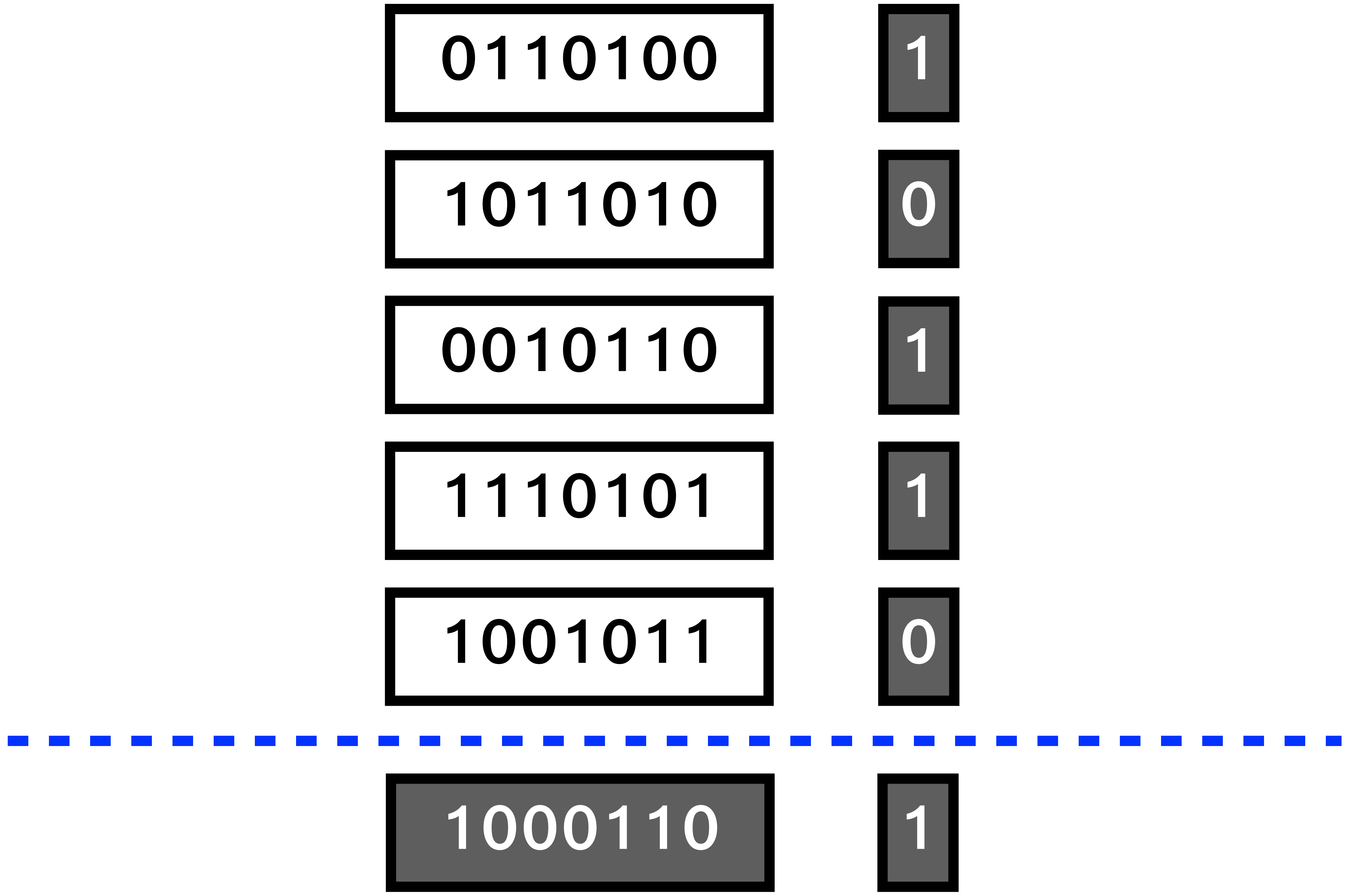
1110101

1001011

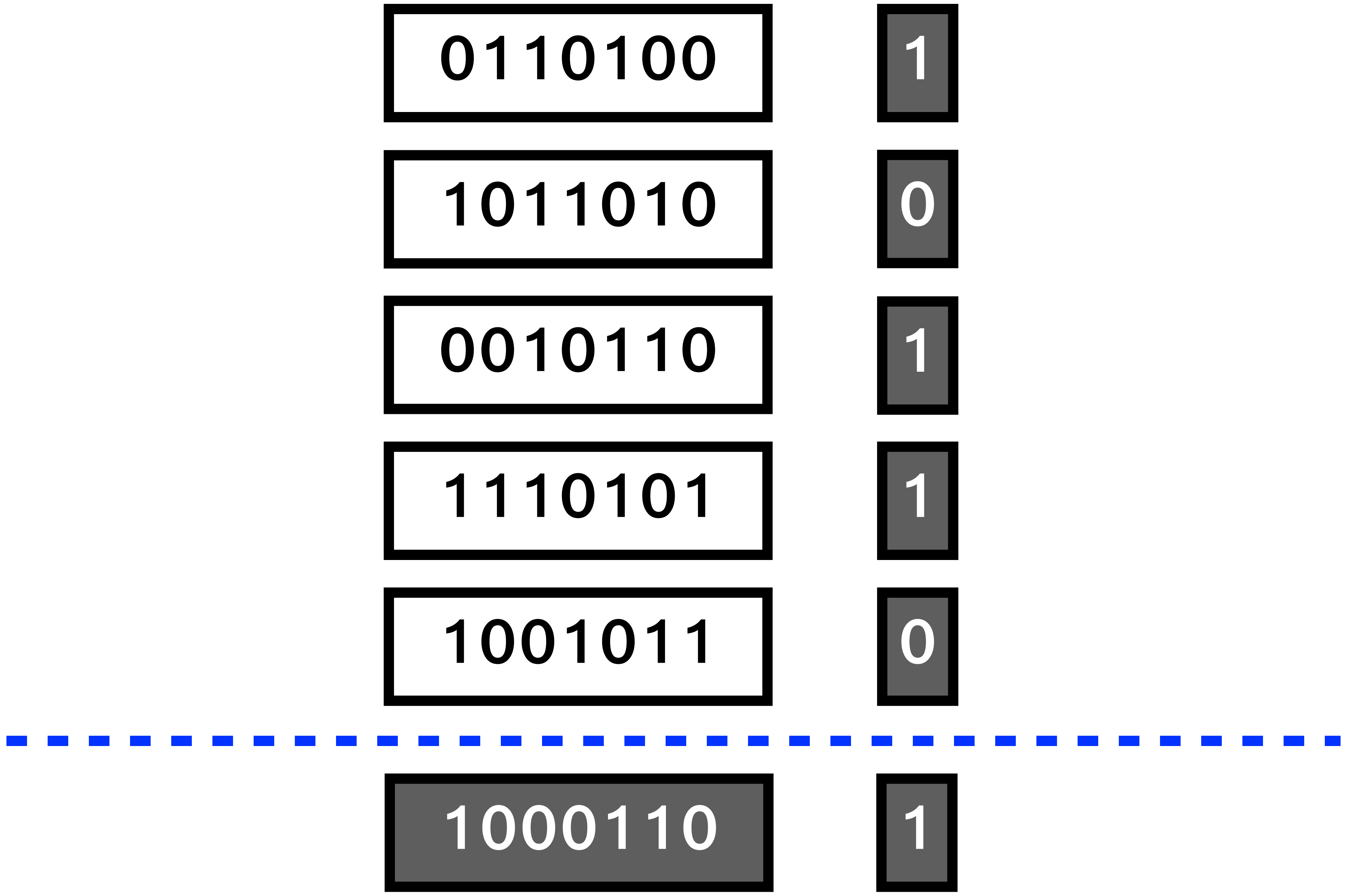
Technique #2: 2-D Parity

0110100	1
1011010	0
0010110	1
1110101	1
1001011	0

Technique #2: 2-D Parity



Technique #2: 2-D Parity

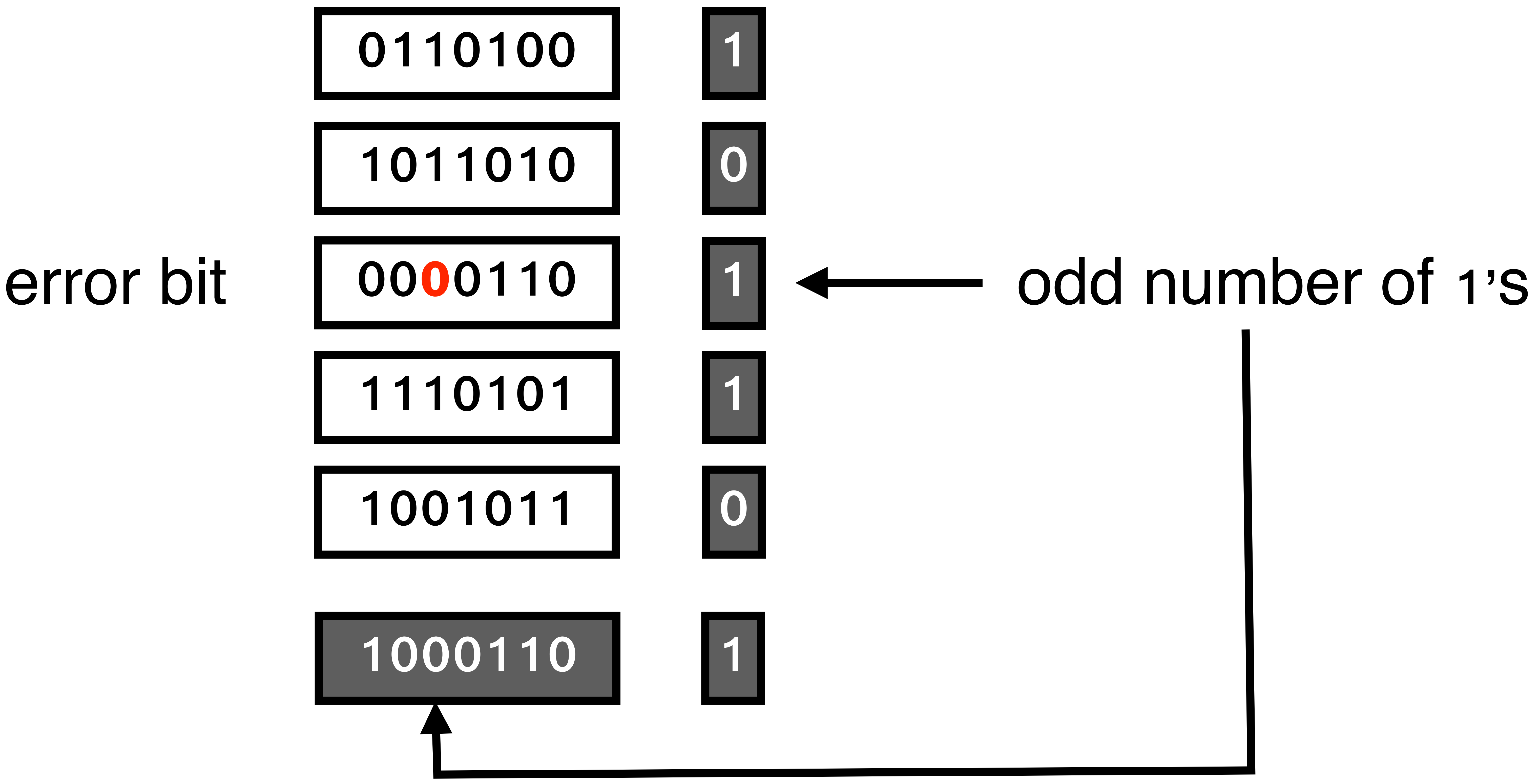


Workflow:

- Make each byte even parity
- Add a parity byte for all bytes of the packet

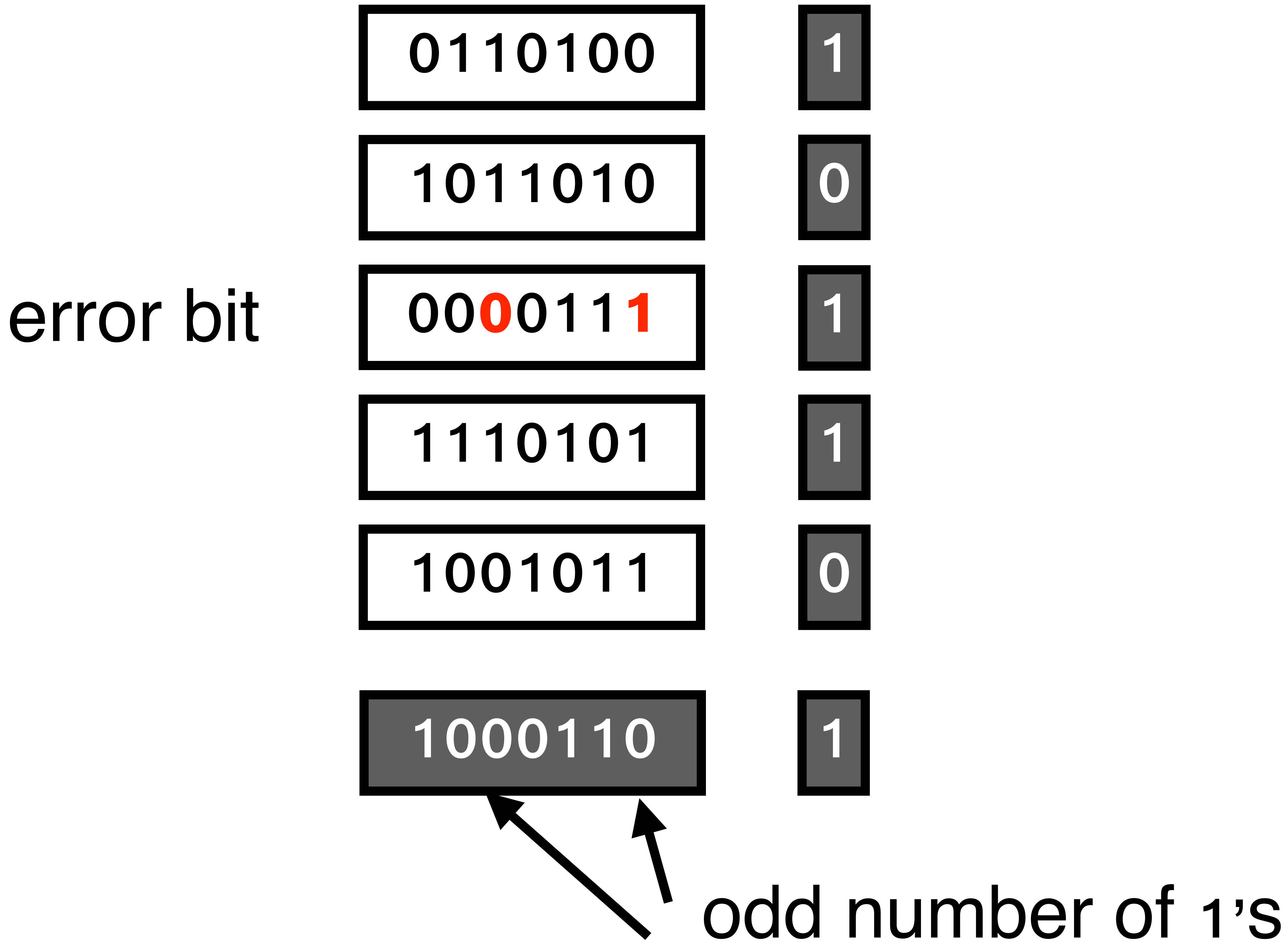
Efficacy of 2-D Parity

1-bit errors can be detected and corrected



Efficacy of 2-D Parity (cont'd)

2-bit errors can also be detected



Technique #3: Internet Checksum

Checksum = add up all the words that are transmitted

- The receiver performs the same calculation on the transmitted data
- E.g., ones complement arithmetic

Simple but not robust

- 16 redundant bits for the whole message
- Easy to be implemented in the software
- Concurrent errors without hurting the sum cannot be detected => low probability

Technique #4: Cyclic Redundancy Check (CRC)

Commonly used codes w/ error detection properties

- Ethernet frame check sequence (CRC-32)

High-level ideas

- #1: View an (n+1)-bit message as an n degree polynomial M(x)
- #2: The sender and receiver agree on the same divisor polynomial C(x)
- #3: Error detection is performed by polynomial arithmetics

CRC is hardware friendly

- k-bit shift register and XOR gates

Terminology

1. Host
2. NIC
3. Multi-port I/O bridge
4. Protocol
5. RTT
6. Packet
7. Header
8. Payload
9. BDP
10. Baud rate
11. Frame/Framing
12. Parity bit
13. Checksum

Principle

1. Layering

Technique

1. NRZ Encoding
2. NRZI Encoding
3. Manchester Encoding
4. 4B/5B Encoding
5. Byte Stuffing
6. Byte Counting
7. Bit Stuffing
8. 2-D Parity
9. CRC

Summary

Today's takeaways

#1: Framing relies on the byte/bit marker

#2: Error handling relies on redundant bits/bytes

Next lecture

- L2 Switching