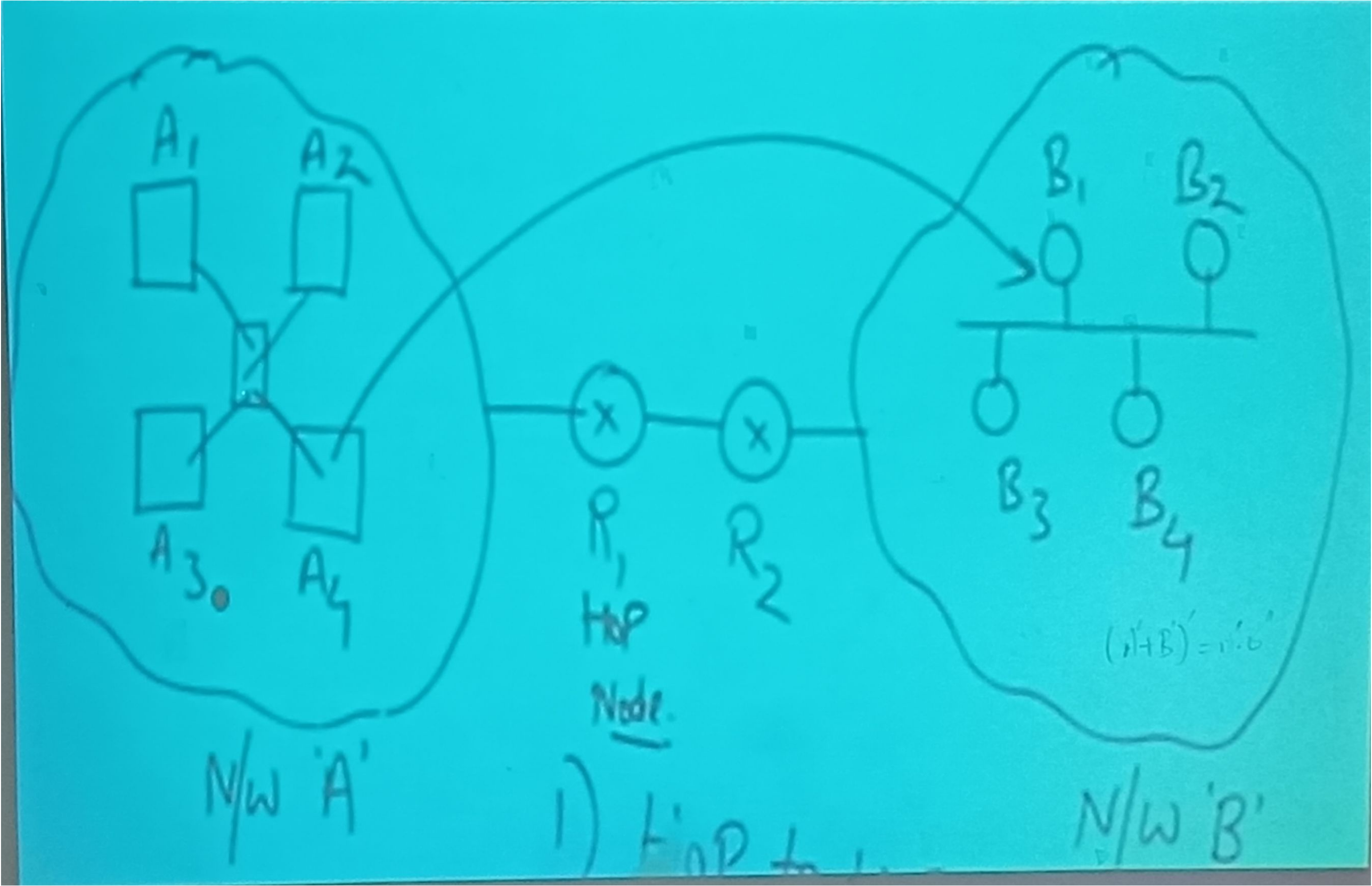

A1
A2
A3
A4
R1
R2
HUB
B1
B2
B3
B4
(A'+B')' = A'.B'
Node.
N/W 'A'
N/W 'B'

Responsibility of Data Link Layer

1. Node to Node delivery
2. Flow Control
3. Error Control
4. Access Control

Framing

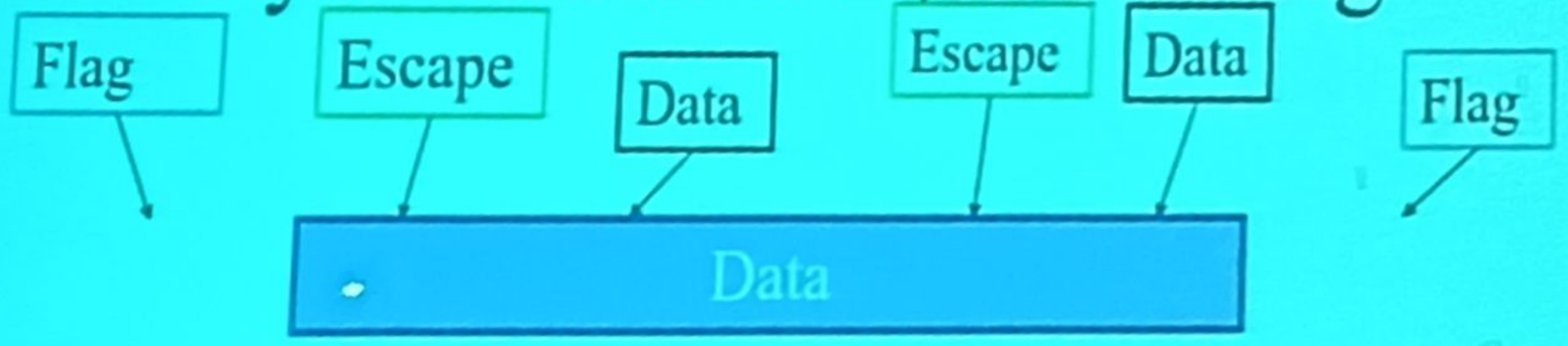
Responsibility of Data Link Layer

1. Node to Node delivery
2. Flow Control
3. Error Control
4. Access Control

Framing

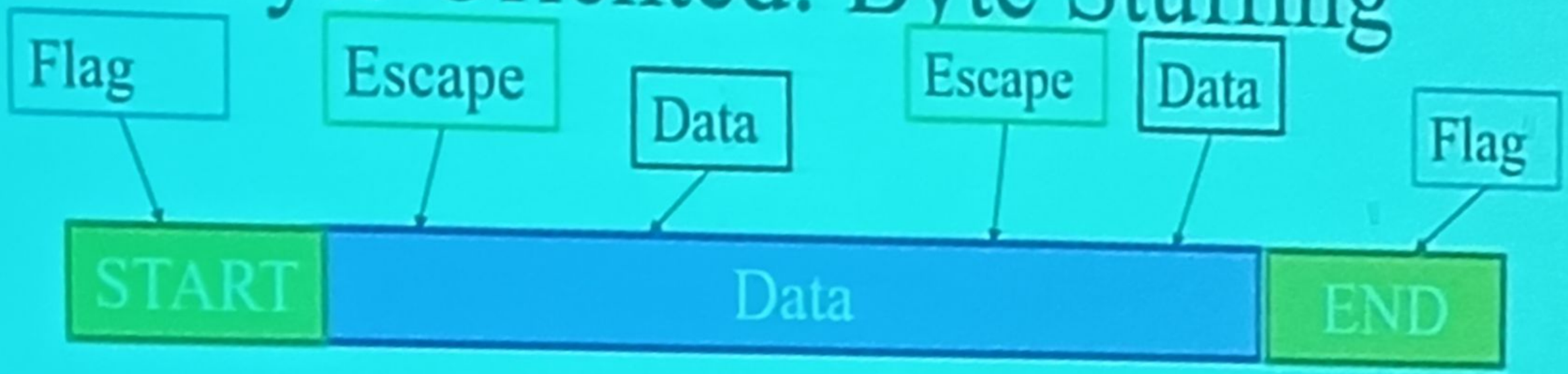
- Types of framing
  - Byte oriented protocols
  - Bit oriented protocols

Byte Oriented: Byte Stuffing



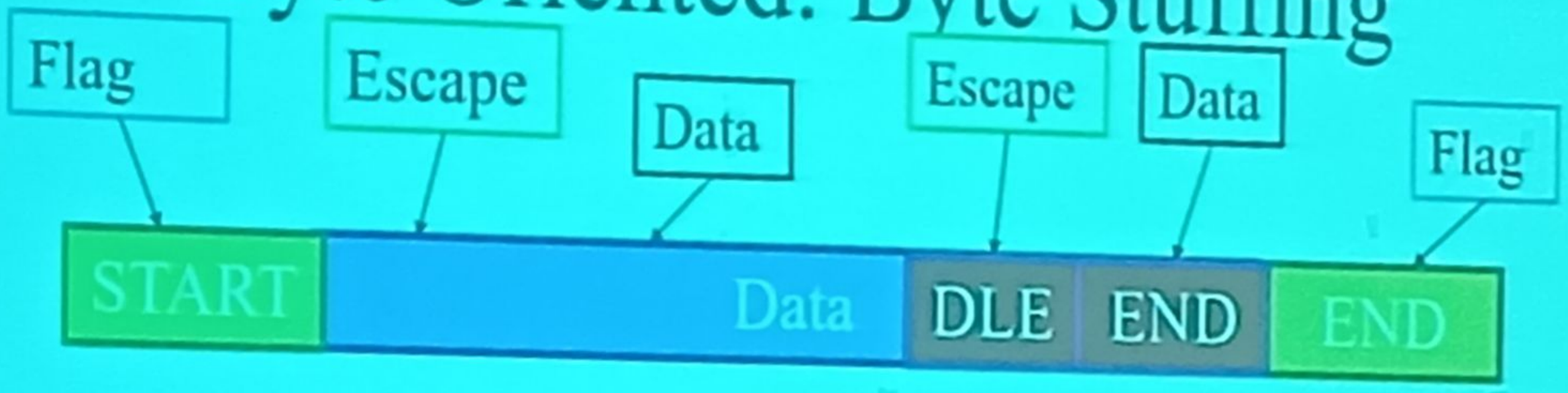
- Add **START** and **END** sentinels to the data

Byte Oriented: Byte Stuffing



- Add **START** and **END** sentinels to the data

Byte Oriented: Byte Stuffing

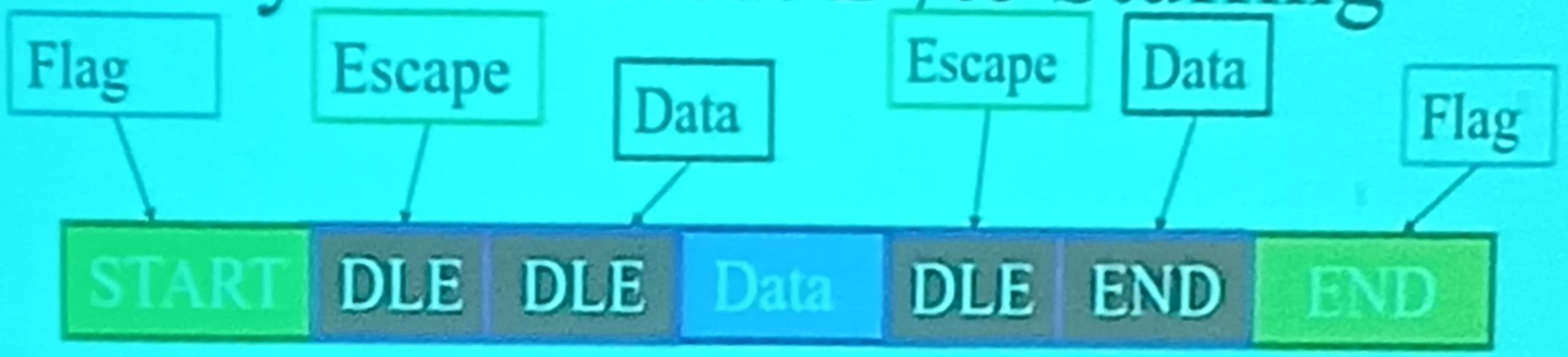


- Add **START** and **END** sentinels to the data
- Problem: what if **END** appears in the data?
  - Add a special **DLE** (Data Link Escape) character before **END**

Byte Oriented: Byte Stuffing

- Add **START** and **END** sentinels to the data
- Problem: what if **END** appears in the data?
  - Add a special **DLE** (Data Link Escape) character before **END**
  - What if **DLE** appears in the data? Add **DLE** before it.

Byte Oriented: Byte Stuffing

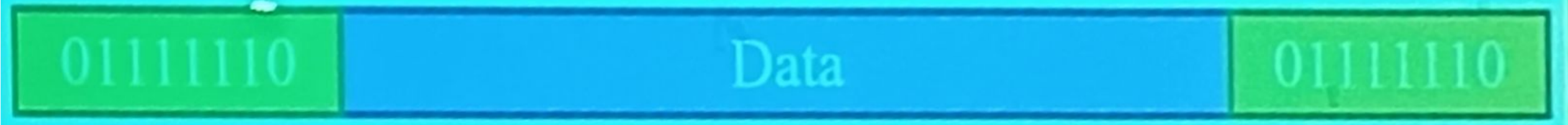


- Add **START** and **END** sentinels to the data

- Problem: what if **END** appears in the data?
  - Add a special **DLE** (Data Link Escape) character before **END**
  - What if **DLE** appears in the data? Add **DLE** before it.
  - Similar to escape sequences in C
    - Printf("You must " quotes in strings");
    - printf("You must \"escape\" quotes in strings");
    - printf("You must \\escape\\ forward slashes as well");

Bit Oriented: Bit Stuffing

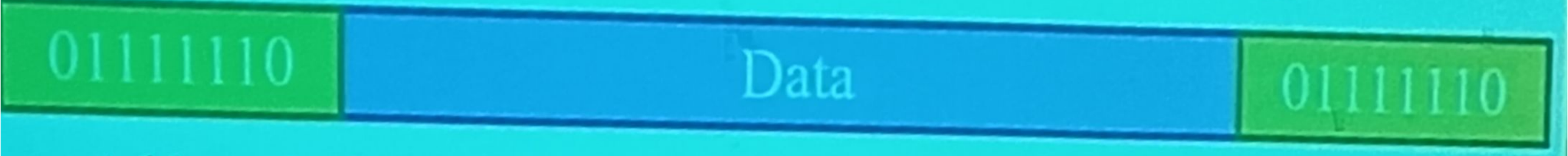
Bit Oriented: Bit Stuffing



- Add sentinels to the start and end of data
  - Both sentinels are the same
  - Example: 01111110 in High-level Data Link Protocol (HDLC)
- Sender: insert a 0 after each 11111 in data
  - Known as "bit stuffing"

$(A'+B')'$

Bit Oriented: Bit Stuffing



- Add sentinels to the start and end of data
  - Both sentinels are the same
  - Example: 01111110 in High-level Data Link Protocol (HDLC)
- Sender: insert a 0 after each 11111 in data
  - Known as "bit stuffing"
- Receiver: after seeing 11111 in the data…
  - 111110 → remove the 0 (it was stuffed)
  - 111111 → look at one more bit
    - 1111110 → end of frame
    - 1111111 → error! Discard the frame
- Disadvantage: 20% overhead at worst
- What happens if error in sentinel transmission?

Error Detection and Correction

Basic concepts

★ Networks must be able to transfer data from one device to another with complete accuracy.

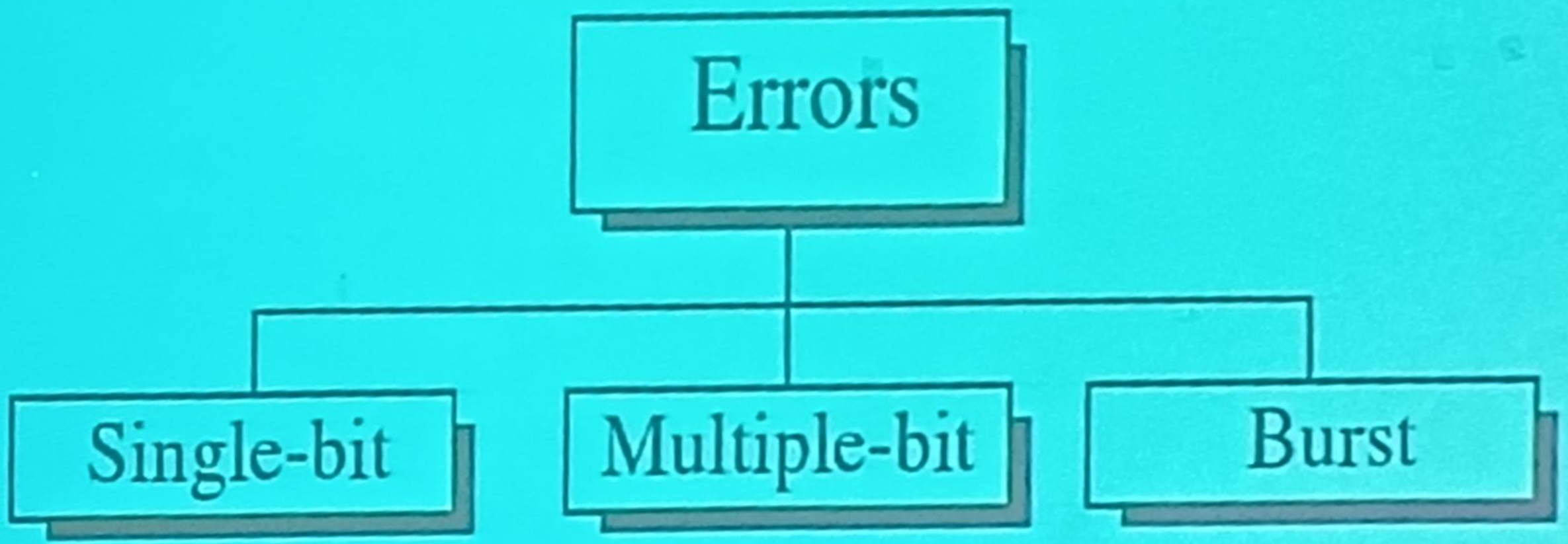
★ Data can be corrupted during transmission.

★ For reliable communication, errors must be detected and corrected.

★ **Error detection and correction** are implemented either at the **data link layer** or the **transport layer** of the OSI model.

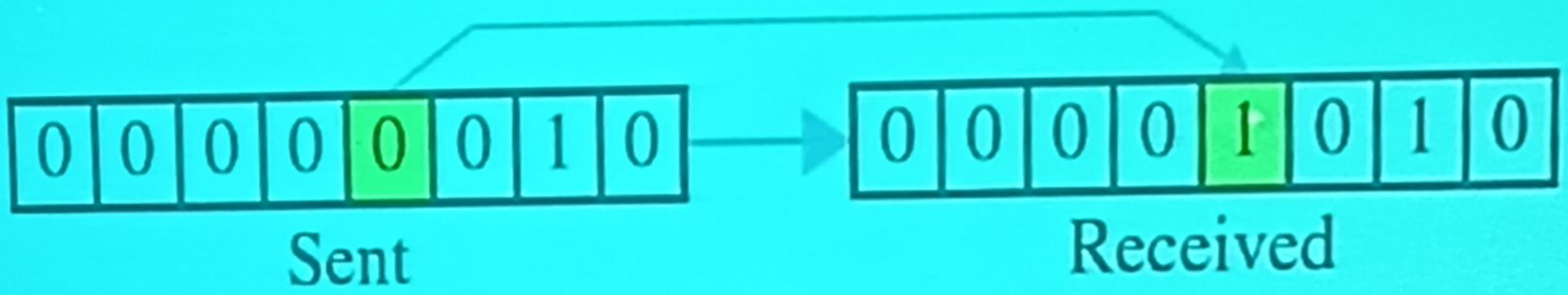
Encoding

Types of Errors



Single-bit error

0 changed to 1



$(A'+B')'$

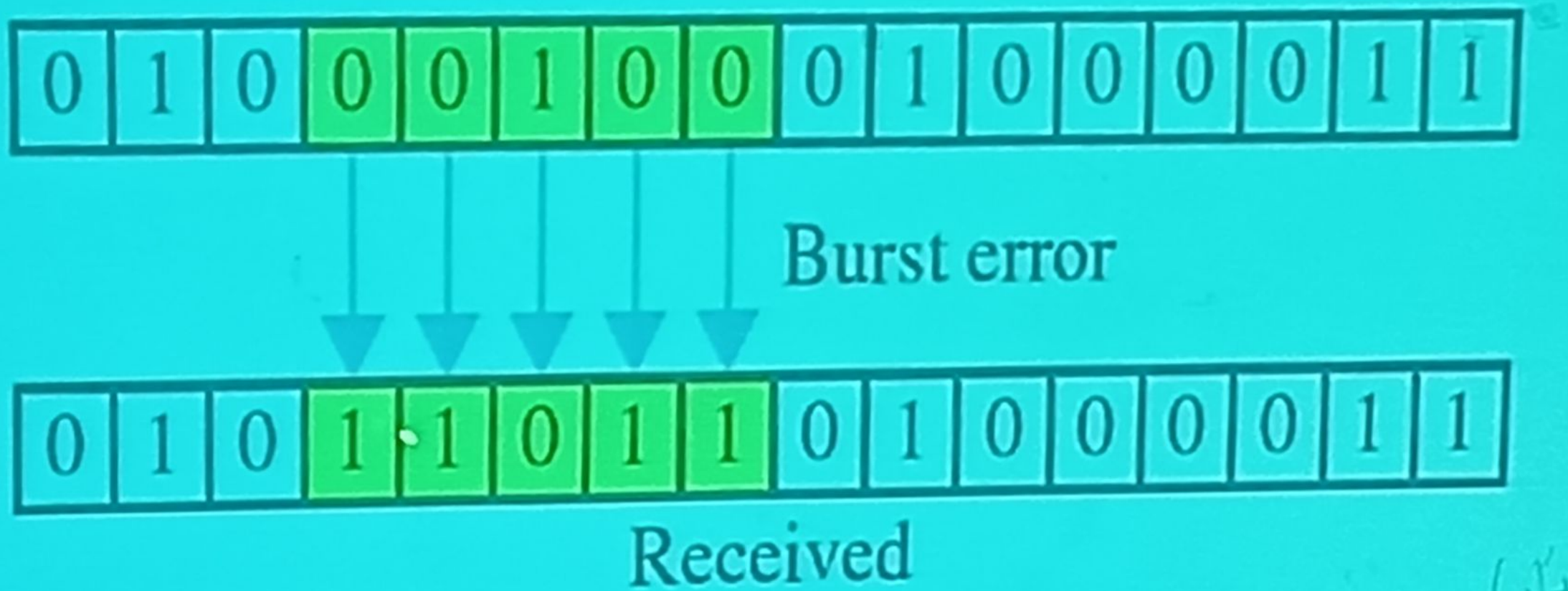
Single bit errors are the **least likely** type of errors in serial data transmission because the noise must have a very short duration which is very rare. However this kind of errors can happen in parallel transmission.

Example:

★If data is sent at 1Mbps then each bit takes only 1/1,000,000 sec. or 1 μs.

★For a single-bit error to occur, the noise must have a duration of only 1 μs, which is very rare.

Burst error


Sent
0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1
Burst error
0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1
Received

The term **burst error** means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

Burst errors does not necessarily **mean that the errors occur in consecutive bits**, the length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

- **Burst error is most likely to happen in serial transmission** since the duration of noise is normally longer than the duration of a bit.
- The number of bits affected depends on the data rate and duration of noise.

Example:

- If data is sent at rate = 1Kbps then a noise of 1/100 sec can affect 10 bits.($1*10^3*(1/100)$)

- If same data is sent at rate = 1Mbps then a noise of 1/100 sec can affect 10,000 bits.($1*10^6*(1/100)$)

Error detection

Error detection means to decide whether the received data is correct or not without having a copy of the original message.

Error detection **uses the concept of redundancy, which means** adding extra bits for detecting errors at the destination.