Error Detection for 8-Byte Messages

1. Introduction

In digital communication, data can get corrupted due to noise or interference. To make sure the message received is the same as the one sent, we add an **error-detecting code**. This report explains the problem in a simple way and shows a small solution.

---

2. Error Types

When sending bits of data, errors can happen such as:

- **Single-bit error:** one bit flips from 0 to 1 or 1 to 0.
- **Burst error:** several nearby bits flip together.
- **Random errors:** bits flip in different positions due to noise.

---

3. Common Error Detection Methods

- **Parity bit:** Add 1 extra bit that makes the total number of 1’s even or odd. Simple, but misses some errors.
- **Checksum:** Add numbers together and send the sum. Detects many errors but not all.
- **CRC (Cyclic Redundancy Check):** A stronger method that can detect many types of errors, including bursts.

---

4. Chosen Method

For our task, we will use **CRC-16**. Reasons:

- More reliable than parity or checksum.
- Can detect all single-bit and double-bit errors.
- Can detect burst errors up to 16 bits.
- Only adds 2 extra bytes to the 8-byte message.

Limitation: CRC cannot correct errors, it only detects them. If an error is found, the message must be resent.

---

5. Demonstration System

We built a simple transmitter/receiver model:

1. **Transmitter**: Takes an 8-byte message, calculates the CRC-16, and attaches it.
2. **Channel**: Simulates errors by flipping some bits.
3. **Receiver**: Recomputes the CRC and checks if it matches. If not, an error is reported.

---

6. Example (in Python)

```python
def crc16(data: bytes):
    poly = 0x1021
    crc = 0xFFFF
    for b in data:
        crc ^= (b << 8)
        for _ in range(8):
            if crc & 0x8000:
                crc = ((crc << 1) ^ poly) & 0xFFFF
            else:
                crc = (crc << 1) & 0xFFFF
    return crc

# Transmit
data = b"ABCDEFGH"  # 8 bytes
crc = crc16(data)
print("Message:", data)
print("CRC:", hex(crc))
```

This code calculates the CRC for the message. If the receiver gets a different CRC, it knows an error happened.

---

7. Conclusion

- Errors are common in communication.
- Simple methods like parity are not strong enough.
- CRC-16 is a good balance for small messages like 8 bytes.
- Our system demonstrates how errors can be detected using CRC.

---

8. References

- Basic communication systems textbooks.