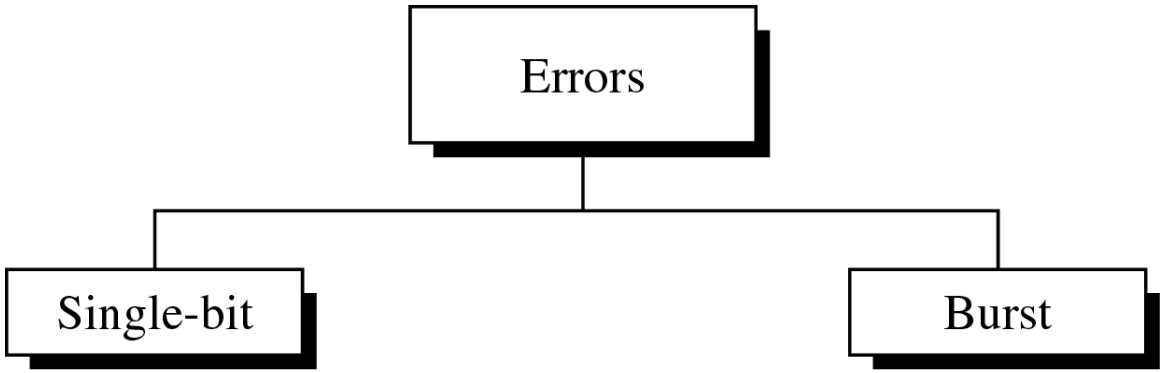
Chapter 10 Error Detection and Correction

1. Introduction
2. Block Coding
3. Checksum

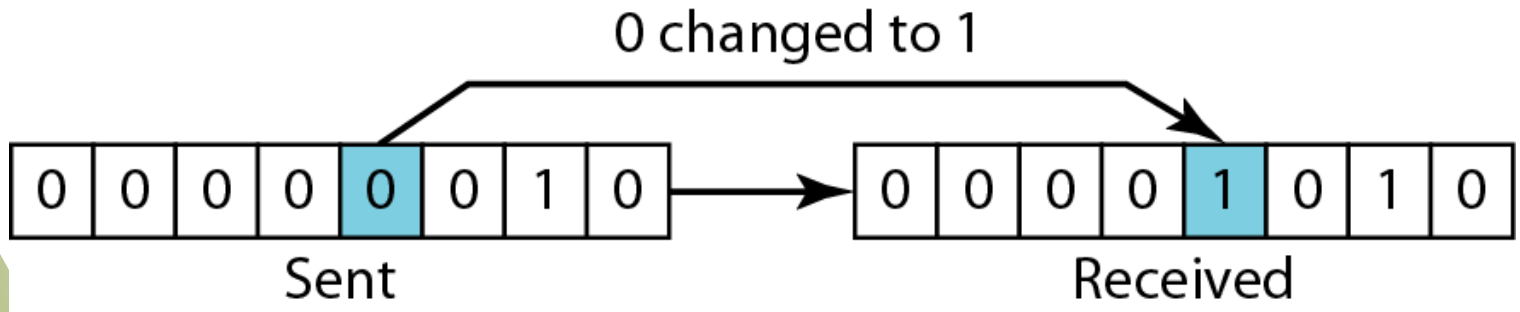
Type of Errors

- An electromagnetic signal is subject to interference from heat, magnetism, and other forms of electricity
- Single-bit error: $0 \rightarrow 1$ or $1 \rightarrow 0$
- Burst error: 2 or more bits have changed



Single-Bit Error

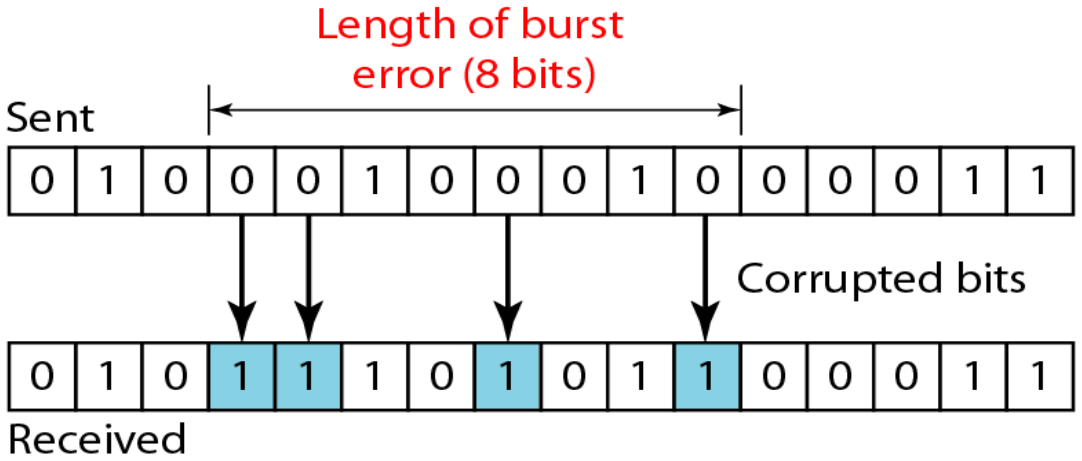
- Only one bit of a given data unit is changed
- The least likely type of error in serial transmission
- Single-bit error can happen in parallel transmission



Dr. Md. Abdur Razzaque
Green Networking Research Group
Dept. of Computer Science and Engineering, University of Dhaka

Burst Error

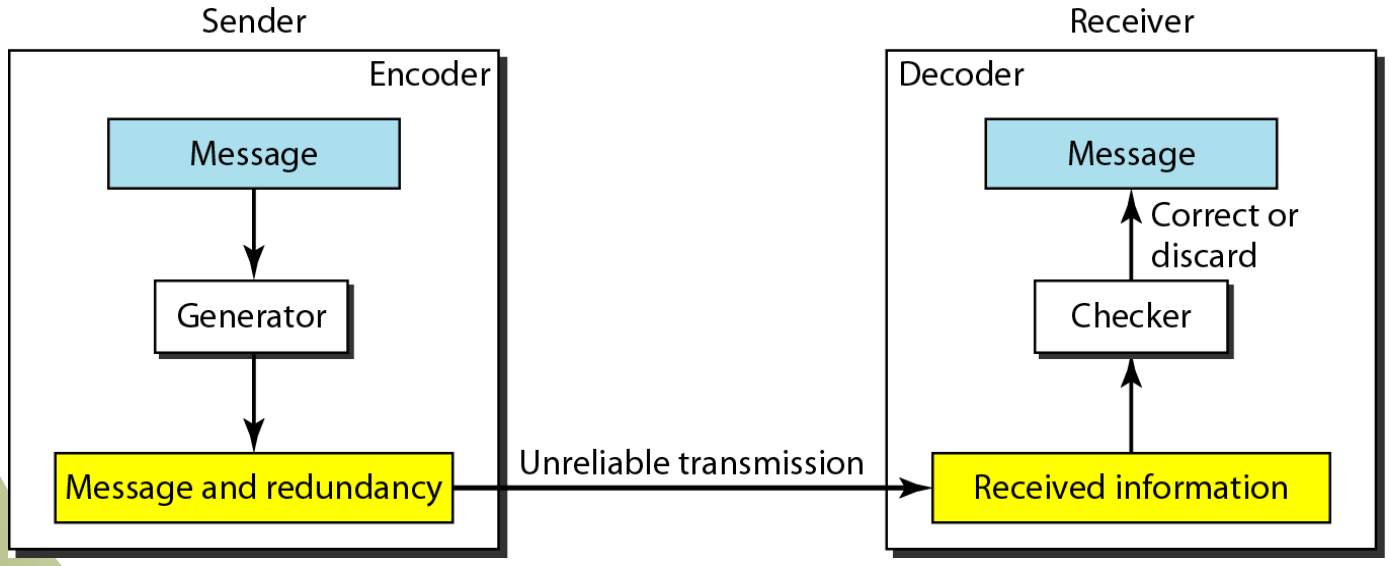
- Two or more bits in the data unit have changed
- Burst error does not necessarily mean that the errors occur in consecutive bits
- Most likely to happen in a serial transmission
- Number of bits affected depends on the data rate and duration of noise



Dr. Md. Abdur Razzaque
Green Networking Research Group
Dept. of Computer Science and Engineering, University of Dhaka

Redundancy

- Error detection uses the concept of *redundancy*, which means adding extra (redundant) bits for detecting errors at the destination



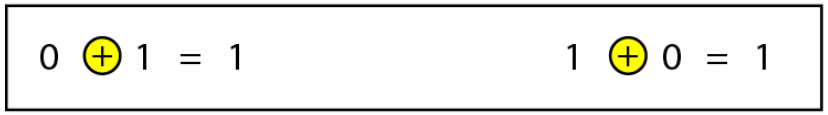
Error Control

- Detection Versus Correction
  - Detection: error ? yes or no
  - Correction: Need to know the exact number of bits that are corrupted, and their location in the message
- Forward Error Correction Versus Retransmission
  - Retransmission (resending) : Backward error correction
- Coding for redundancy
  - Block coding: discussed in our textbook
  - Convolution coding

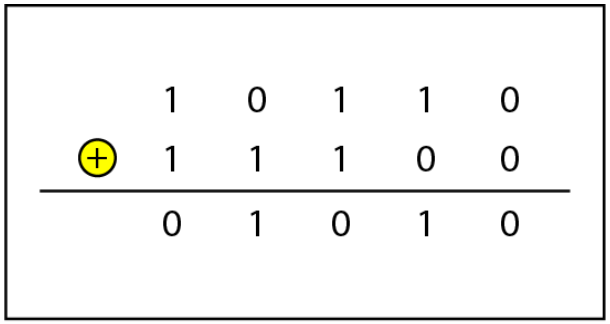
Modular Arithmetic

- In modulo-N arithmetic, we use only the integers in the range 0 to N-1, inclusive.
- Adding: $0 + 0 = 0$ $0 + 1 = 1$ $1 + 0 = 1$ $1 + 1 = 0$
- Subtracting: $0 - 0 = 0$ $0 - 1 = 1$ $1 - 0 = 1$ $1 - 1 = 0$
- XORing of two single bits or two words

a. Two bits are the same, the result is 0.



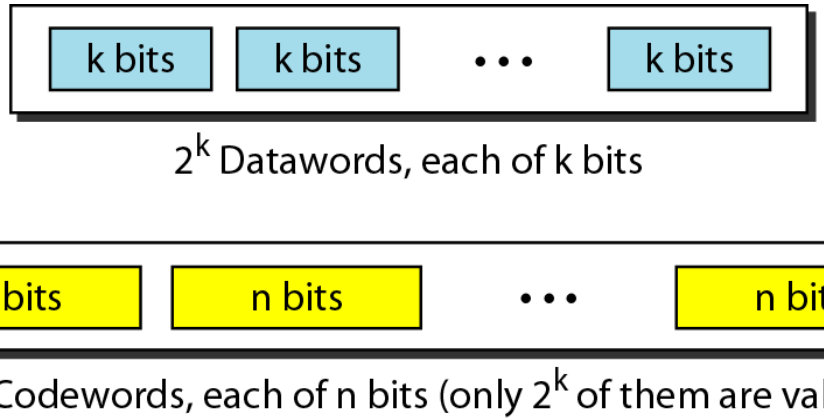
b. Two bits are different, the result is 1.



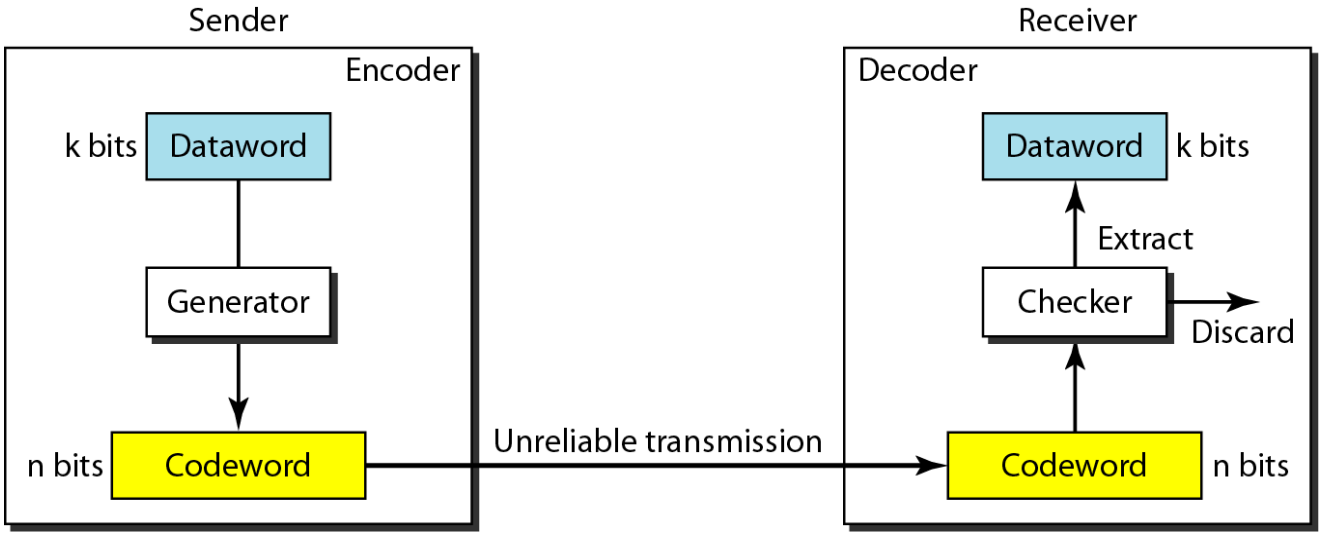
c. Result of XORing two patterns

Block Coding

- Divide the message into blocks, each of *k* bits, called **datawords.**
- Add *r* redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called **codewords**
- Example: 4B/5B block coding
  - k = 4 and n = 5.
  - $2^k = 16$ datawords and $2^n = 32$ codewords.



Error Detection in Block Coding



- **Example:**
  Assume that k = 2 and n = 3 (Table 10.1)

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Error Detection: Example

- Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

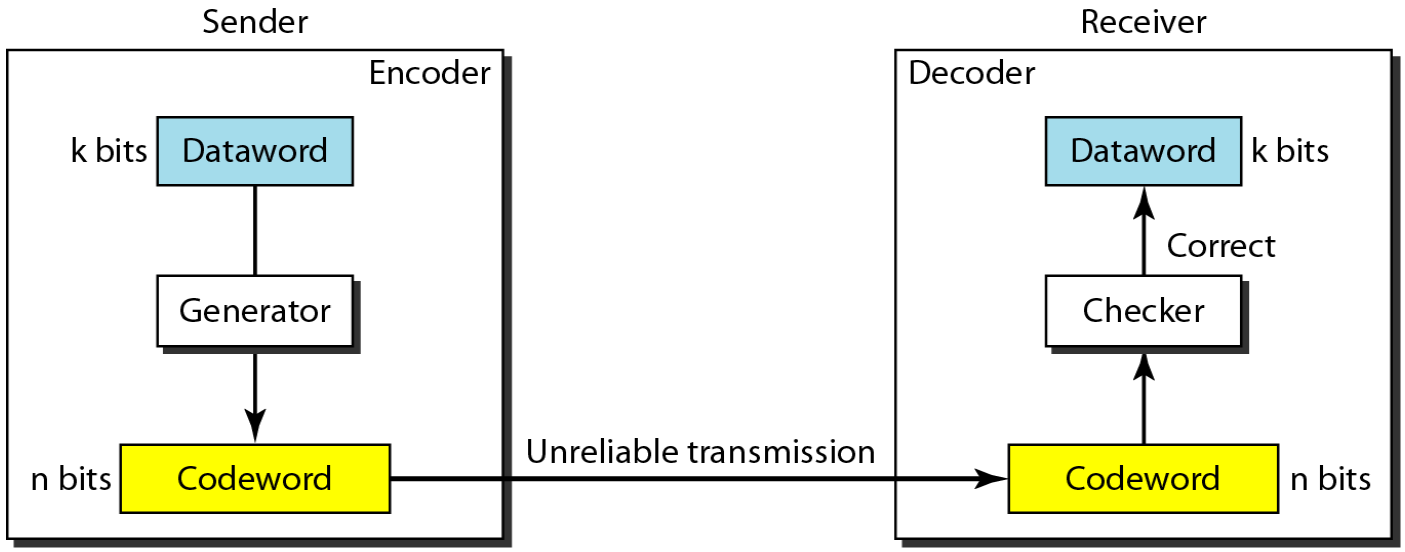
1. The receiver receives 011 which is a valid codeword. The receiver extracts the dataword 01 from it.

2. The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded.

3. The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

→ **An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected**

Error Correction in Block Coding



- **Example:**
  Assume that k = 2 and r = 3
  n = 5 (Table 10.2)

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Error Correction: Example

- Assume the dataword is 01. The sender creates the codeword 01011. The codeword is corrupted during transmission, and 01001 is received. First, the receiver finds that the received codeword is not in the table. This means an error has occurred. The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword

1. *Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits. (the same for third or fourth one in the table)*
2. *The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit.*

Hamming Distance

- The **Hamming distance** between two words is the number of differences between corresponding bits.
- Example: Hamming distance d(10101, 11110) is 3

- The **minimum Hamming distance** is the smallest Hamming distance between all possible pairs in a set of words
- Example for Table 10.1
  - $d_{min} = 2$

$d(000, 011) = 2$	$d(000, 101) = 2$	$d(000, 110) = 2$	$d(011, 101) = 2$
$d(011, 110) = 2$	$d(101, 110) = 2$		

Hamming Distance

- Three parameters to define the coding schemes
  - Codeword size n
  - Dataword size k
  - The minimum Hamming distance d_{min}
- Coding scheme $C(n, k)$ with a separate expression for d_{min}
- Hamming distance and error
  - Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted
- Minimum distance for error detection
  - To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

Minimum Hamming Distance: Example

- The minimum Hamming distance in Table 10.1 is 2. This code guarantees detection of only a single error.
  For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

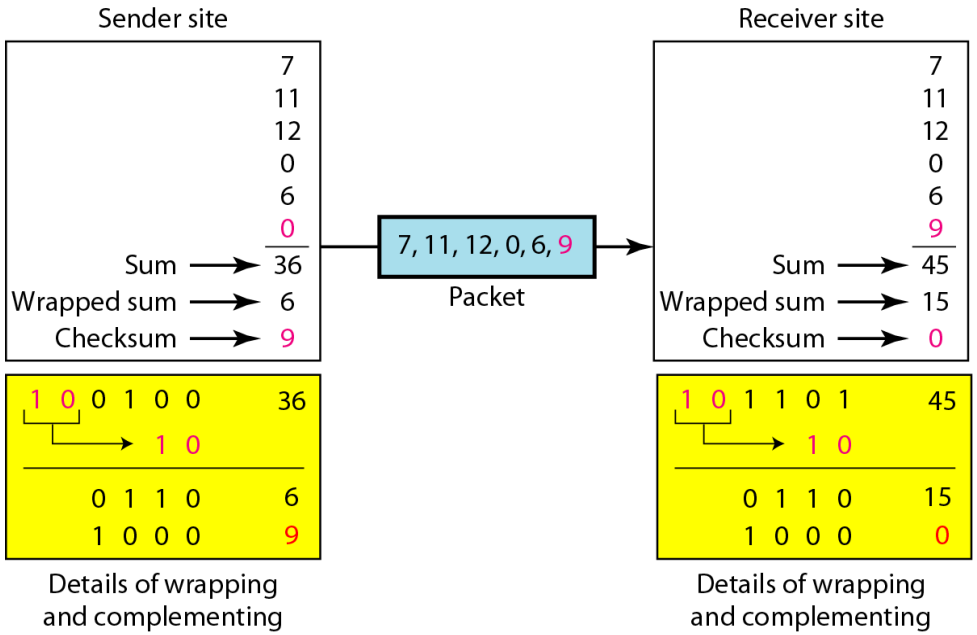
- In Table 10.2, it has $d_{min} = 3$. This code can detect up to two errors.
  When any of the valid codewords is sent, two errors create a codeword which is not in the table of valid codewords. The receiver cannot be fooled. However, some combinations of three errors change a valid codeword to another valid codeword. The receiver accepts the received codeword and the errors are undetected.

Checksum

- Tendency is to replace the checksum with a CRC
- Not as strong as CRC in error-checking capability
- One's complement arithmetic
  - We can represent unsigned numbers between 0 and $2^n - 1$ using only n bits
  - If the number has more than n bits, the extra leftmost bits need to be added to the n rightmost bits (wrapping)
  - A negative number can be represented by inverting all bits. It is the same as subtracting the number from $2^n - 1$

Checksum: Example

- The sender initializes the checksum to 0 and adds all data items and the checksum. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6. The sum is then complemented, resulting in the checksum value 9 (15 − 6 = 9).



Internet Checksum

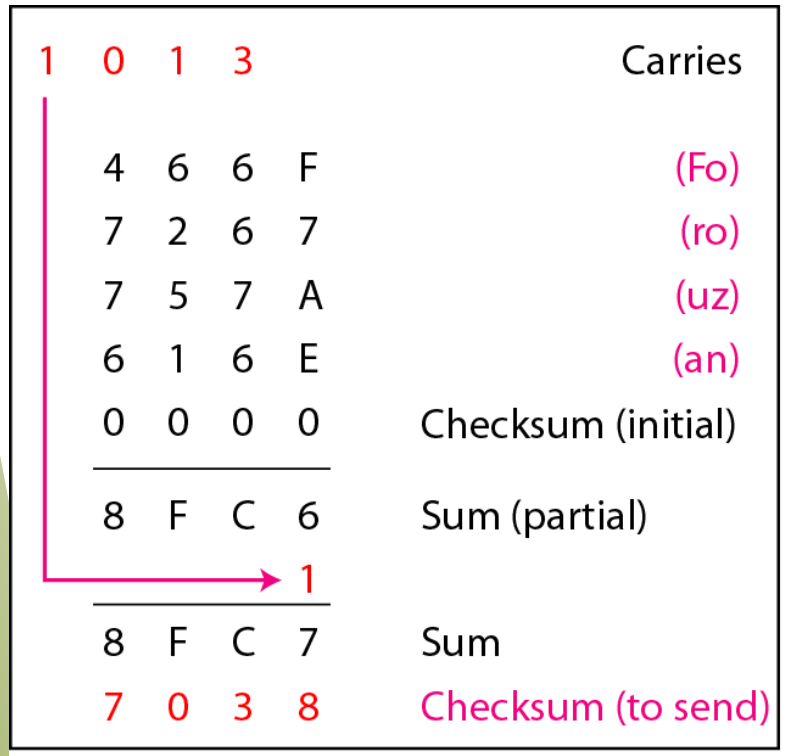
Sender site:

1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

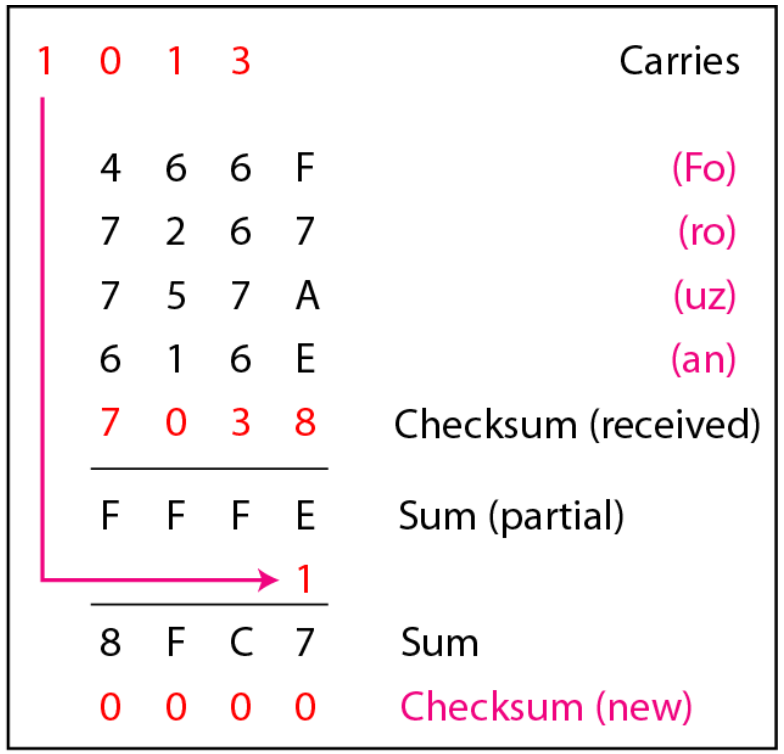
Receiver site:

1. The message (including checksum) is divided into 16-bit words.
2. All words are added using one's complement addition.
3. The sum is complemented and becomes the new checksum.
4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

Internet Checksum: Example



a. Checksum at the sender site



a. Checksum at the receiver site