DATA COMMUNICATION

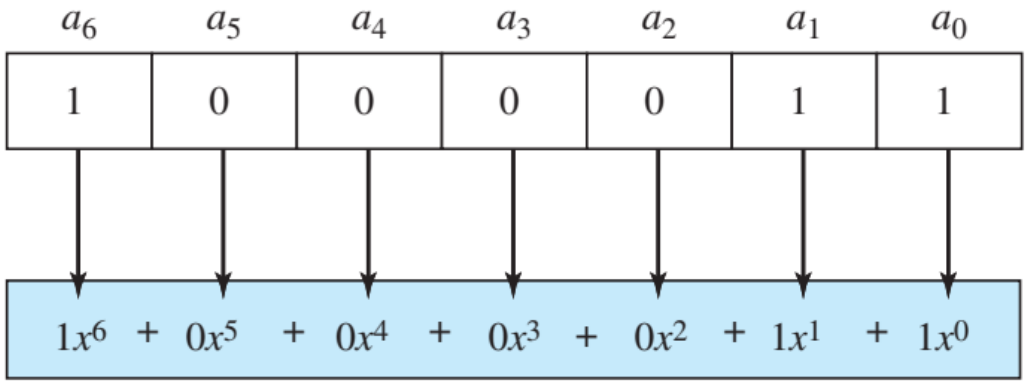
CSE 225/233

WEEK-11, LESSON-1 & 2

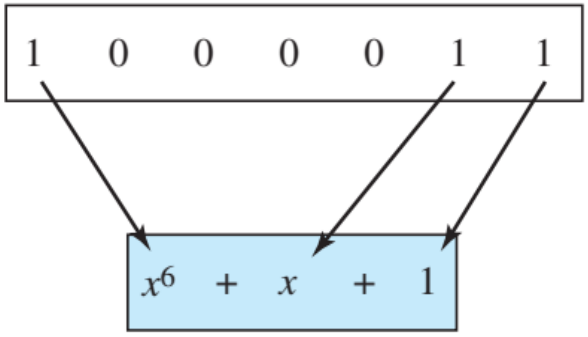
ERROR DETECTION

Polynomial

A better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials. A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit.

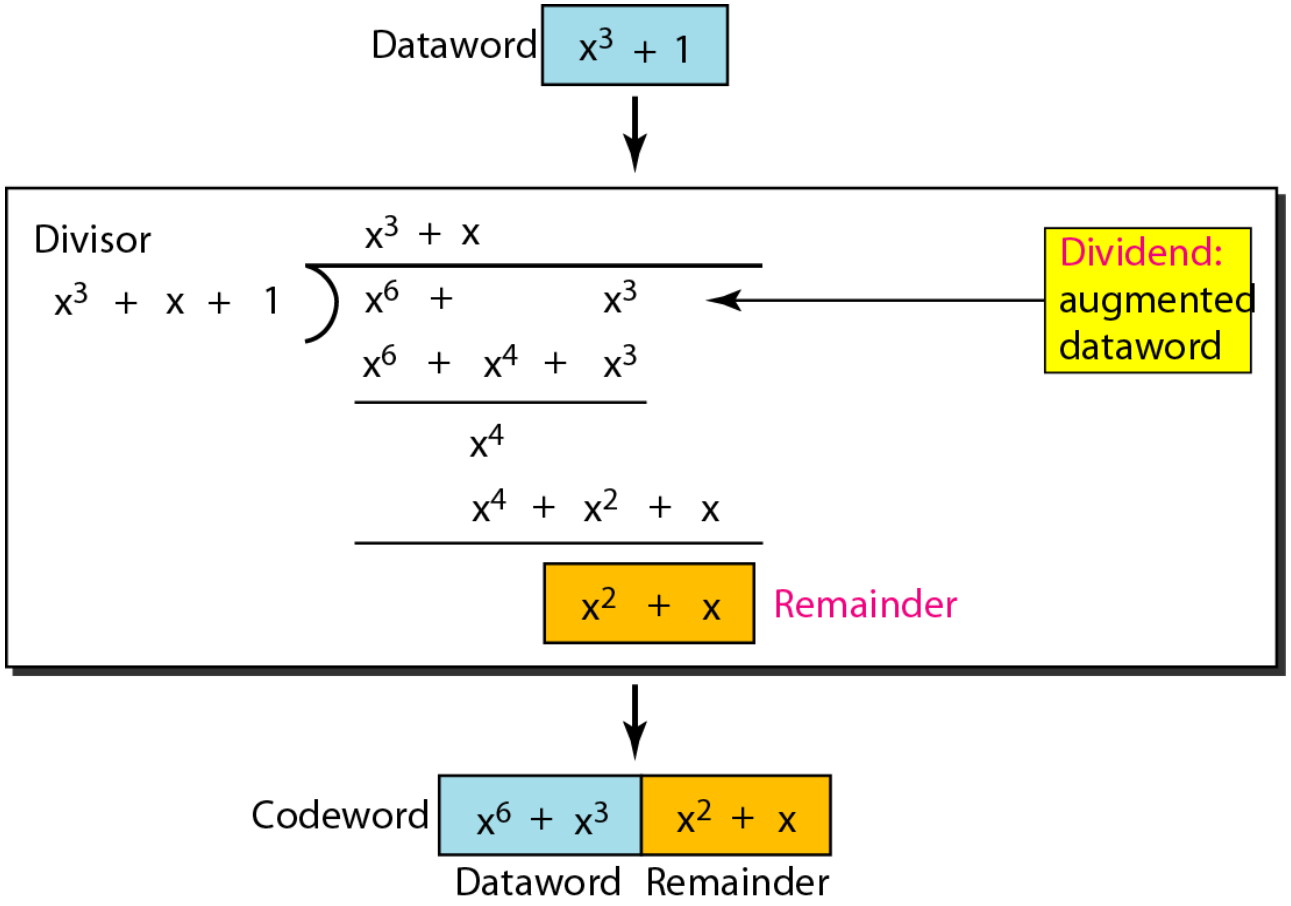


a. Binary pattern and polynomial



b. Short form

CRC Division using polynomial



Standard Polynomial

Name	*Polynomial*	*Used in*
CRC-8	$x^8 + x^2 + x + 1$ **100000111**	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ **11000110101**	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ **1000100000100001**	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ **10000010011000001000111011011 0111**	LANs

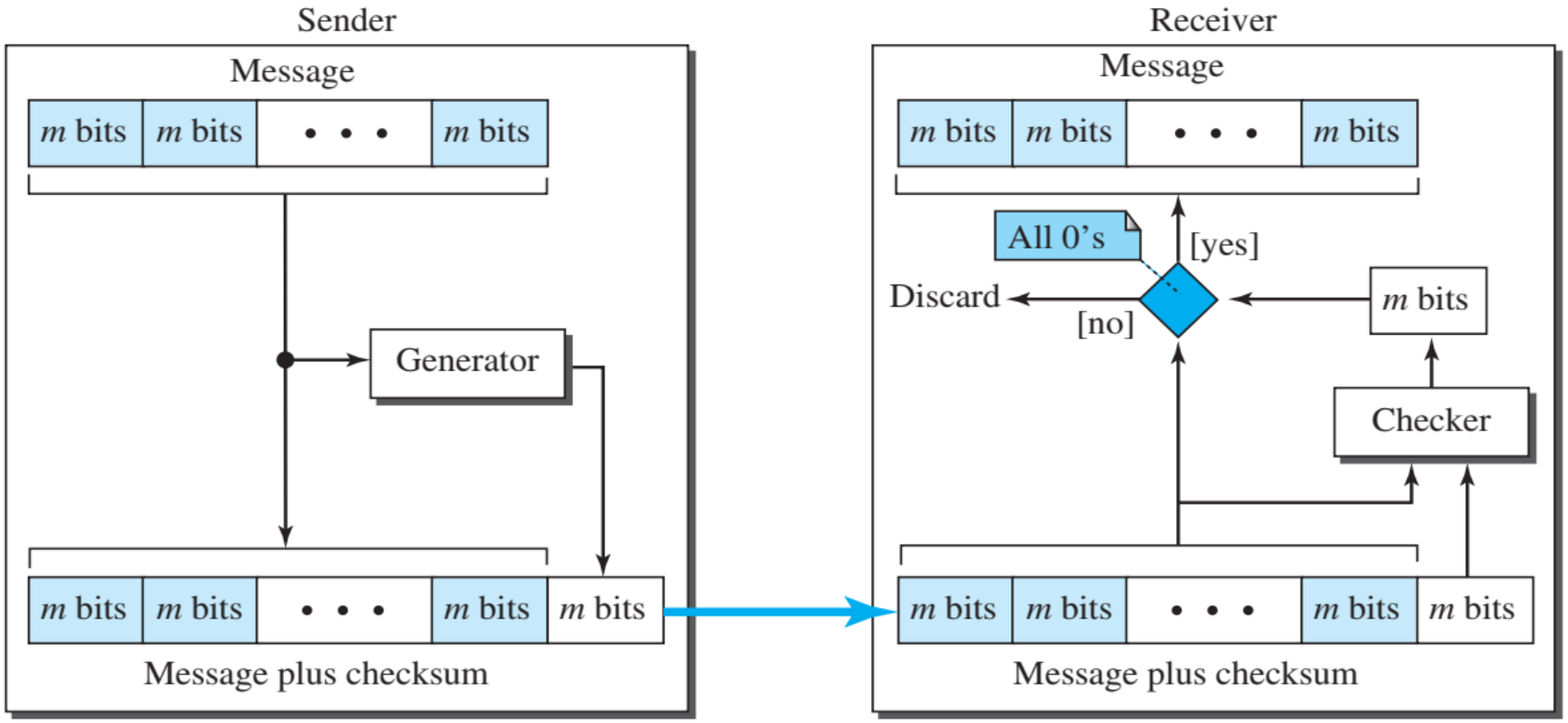
Checksum

Checksum is an error-detecting technique that can be applied to a message of any length. In the Internet, the checksum technique is mostly used at the network and transport layer rather than the data link layer.

At the source, the message is first divided into m-bit units. The generator then creates an extra m-bit unit called the checksum, which is sent with the message. At the destination, the checker creates a new checksum from the combination of the message and sent checksum. If the new checksum is all 0s, the message is accepted; otherwise, the message is discarded.

Conceptual figure of checksum in presented in the next slide.

Checksum (Concept)



Example

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

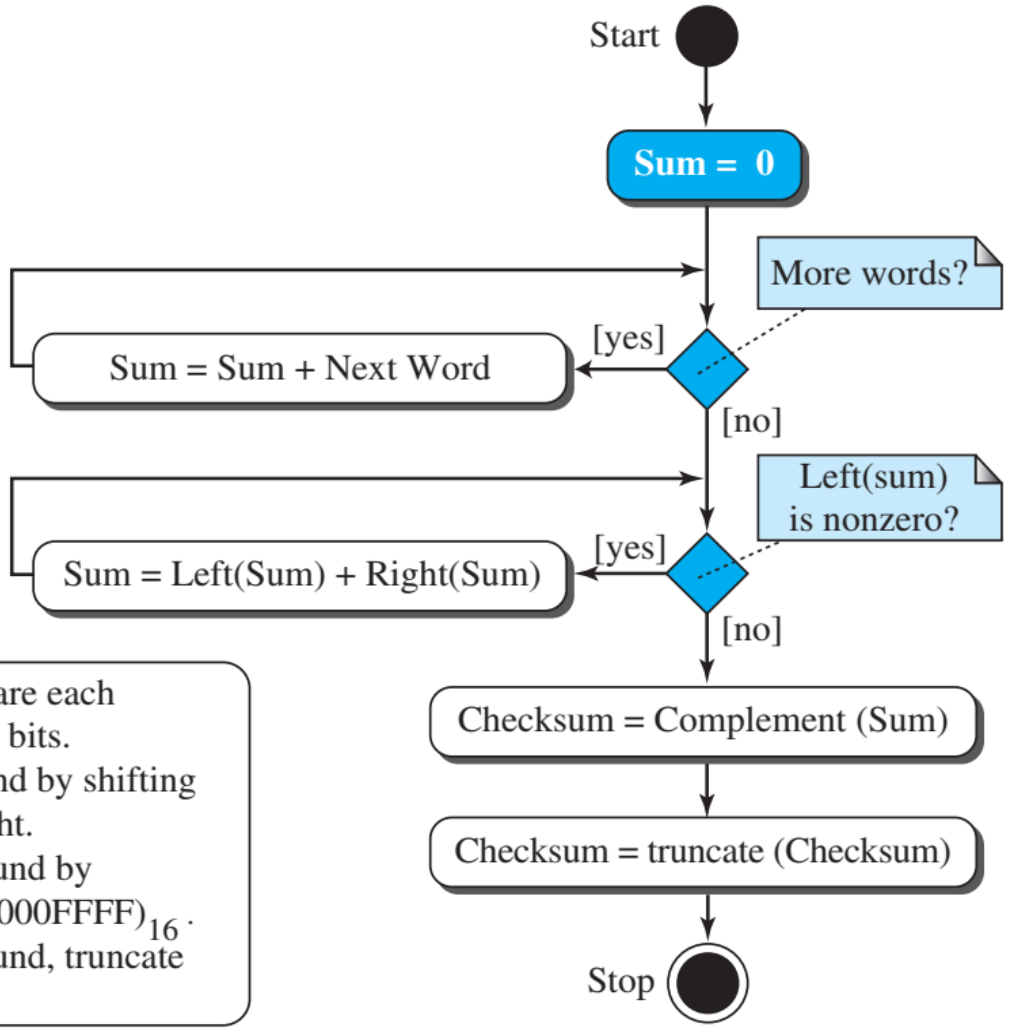
Example (Contd.)

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12, 0, 6, −36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

Complementary Checksum (Procedure)

Sender	*Receiver*
1. The message is divided into 16-bit words. 2. The value of the checksum word is initially set to zero. 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.	1. The message and the checksum are received. 2. The message is divided into 16-bit words. 3. All words are added using one's complement addition. 4. The sum is complemented and becomes the new checksum. 5. If the value of the checksum is 0, the message is accepted; otherwise, it is rejected.

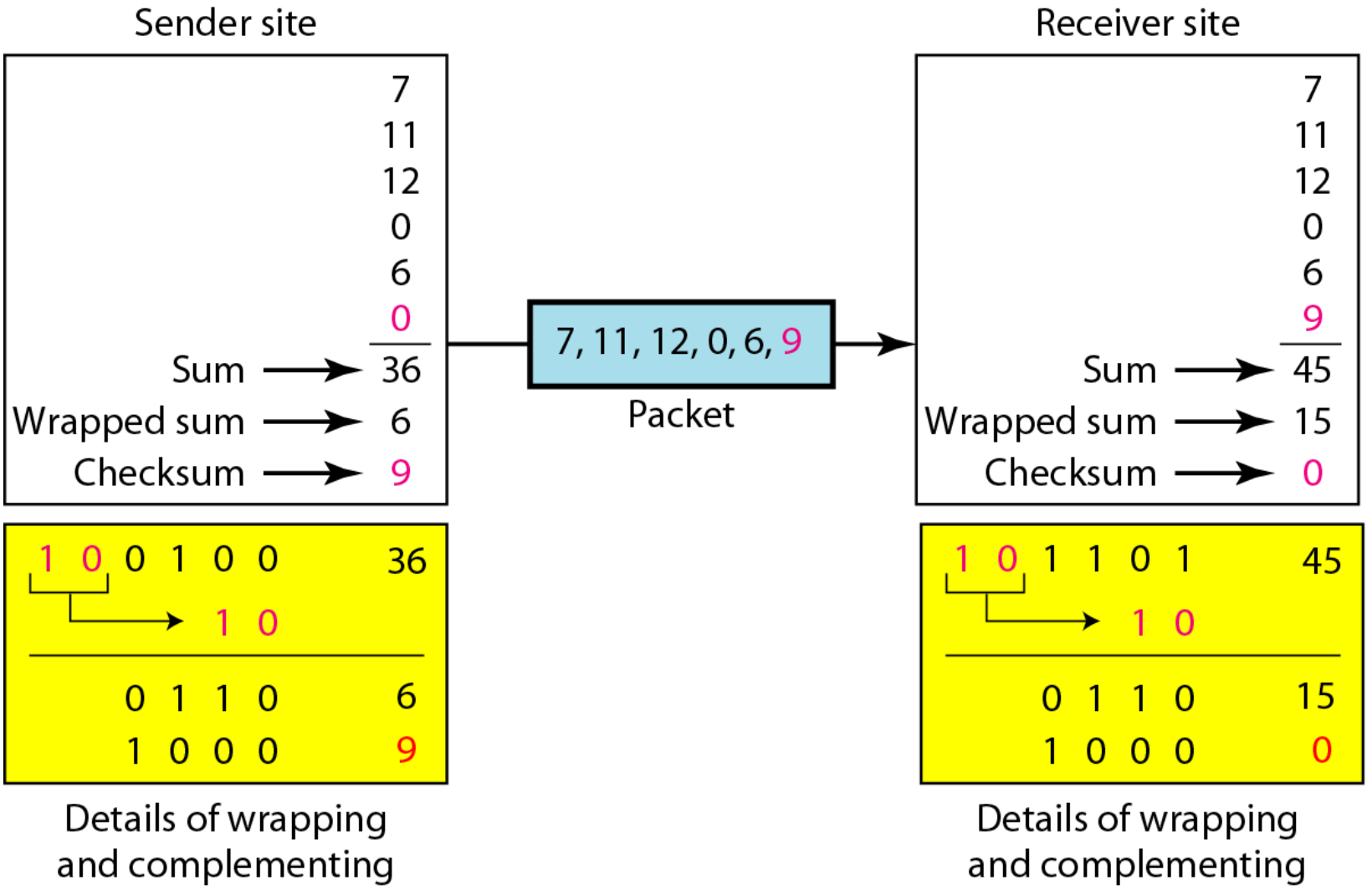
Complementary Checksum (Algorithm)



Notes:

a. Word and Checksum are each 16 bits, but Sum is 32 bits.
b. Left(Sum) can be found by shifting Sum 16 bits to the right.
c. Right(Sum) can be found by ANDing Sum with $(0000FFFF)_{16}$.
d. After Checksum is found, truncate it to 16 bits.

Example - Complementary Checksum




Thank you!