Week 13: Internet Checksum

EE3017/IM2003 Computer Communications

Prof. Cheng Tee Hiang
Room: S1-B1a-29
Email: ethcheng@ntu.edu.sg
Phone: 6790-4534

Topic Outline
(Updated in January 2020)

Topic Outline

Wireless Local Area Network (WLAN)

- Checksum
- Internet Checksum Implementation
- Modular Maths
- Congruence Modulo
- Modular Arithmetic
- Checksum Calculation
- Internet Checksum: Undetectable Errors

Recommended reading:
Section 3.3, Pages 236 to 240 and Section 3.5, Pages 268 to 297 of the recommended textbook
(Page numbers are based on 5^{th} or 2010 Edition.)

Learning Objectives

Learning Objectives

By the end of this topic, you should be able to:

- Explain internet checksum and its implementation.
- Perform modular arithmetic.
- Perform checksum calculation using binary number, hexadecimal number and modular arithmetic.
- Explain errors that cannot be detected by checksum.

Checksum

Checksum

- Error-detecting technique that can be applied to a message of any length.
- Mostly used at the **network** and **transport** layer; e.g., IPv4, TCP, and UDP etc.
- IP header Checksum is recalculated at every router, so the algorithm is selected for ease of implementation in software.
- TCP/UDP checksum provides end-to-end error detection for both the **transport header** (including network and transport layer information) and the **transport layer data payload**.

Internet Checksum Implementation

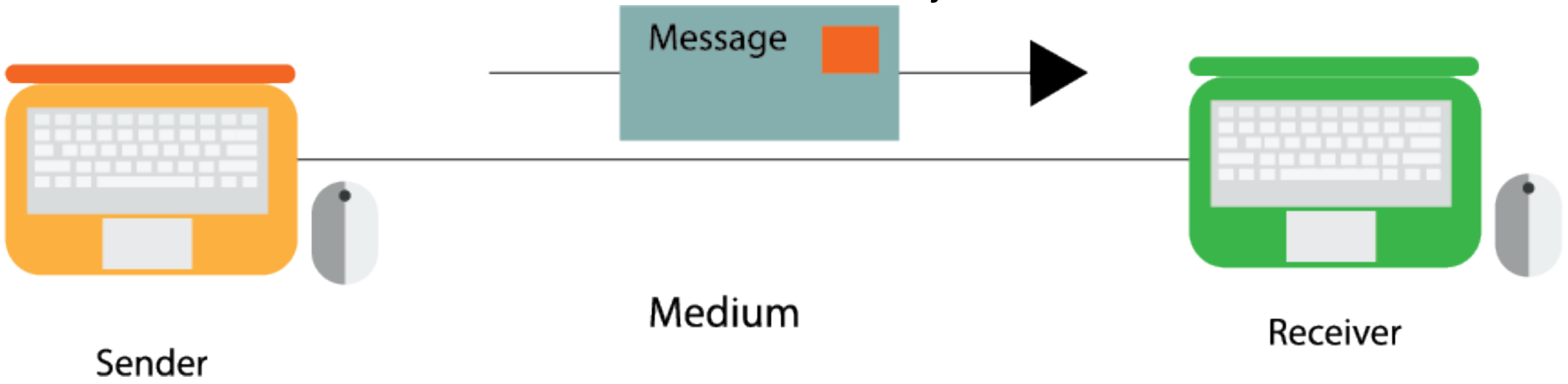
Internet Checksum Implementation

Sender

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.

Receiver

1. The message and the checksum is 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.



Modular Maths

Modular Maths

When we divide two integers we will have an equation that looks like the following:

A / B = Q remainder R

A is the dividend
B is the divisor
Q is the quotient
R is the remainder

Sometimes, we are only interested in what the remainder is when we divide A by B.

For these cases there is an operator called the modulo operator (abbreviated as mod).

Modular Arithmetic

Using the same A, B, Q, and R as above, we would have: A mod B = R.

We would say this as A modulo B is equal to R. Where B is referred to as the modulus.

For example:

13 / 5 = 2 remainder 3

Can be written as:

13 mod 5 = 3

Congruence Modulo

Congruence Modulo

You may see an expression like:

A ≡ B (mod C)

This says that A is congruent to B modulo C.

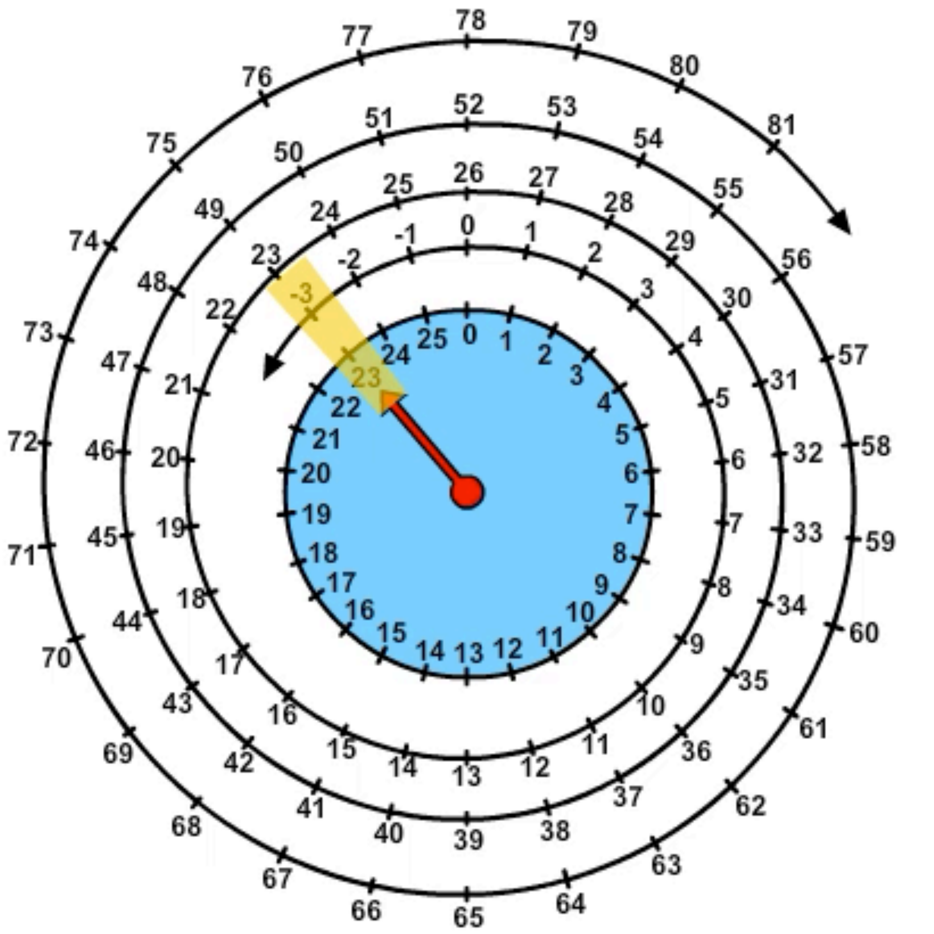
- ≡ is the symbol for congruence, which means the values A and B are in the same equivalence class.
- (mod C) tells us what operation we applied to A and B.
- when we have both of these, we call "≡" congruence modulo C.

e.g. 26 ≡ 21 ≡ 11 (mod 5)

(Note that for convenience we will use '=' in place of '≡' in this course.)

26 mod 5 = 1 so it is in the equivalence class for 1,
21 mod 5 = 1 so it is in the equivalence class for 1,
11 mod 5 = 1 so it is in the equivalence class for 1, as well.

Example: Modulo 26 clock



Counter: -3

Modular Arithmetic

Modular Arithmetic

When a, b, c and d are integers and m is a positive integer such that:

$$a \equiv c \pmod{m}$$
$$b \equiv d \pmod{m}$$

Addition rule:

$$a + b \equiv c + d \pmod{m}$$

Subtraction rule:

$$a - b \equiv c - d \pmod{m}$$

Note also:

$-a = m.n - a \pmod{m}$ where n is an integer.

Examples

Assume m = 5, a = 8, c = 13, b = 6, and d = 11, n = 2

$$a \equiv c \pmod{m} \qquad 8 \equiv 13 \pmod{5}$$
$$b \equiv d \pmod{m} \qquad 6 \equiv 11 \pmod{5}$$

Addition rule:

$$a + b \equiv c + d \pmod{m} \qquad 8 + 6 \equiv 13 + 11 \pmod{5}$$

Subtraction rule:

$$a - b \equiv c - d \pmod{m} \qquad 8 - 6 \equiv 13 - 11 \pmod{5}$$

Note also:

$$-a = m.n - a \pmod{m} \qquad -8 \equiv 5 \times 2 - 8 \pmod{5}$$

Checksum Calculation

Checksum Calculation: Method 1

Here we use **IP header Checksum** as an example:

Let the IP header consist of L, 16-bit words,

$$b_0, b_1, b_2, \ldots, b_{L-1}$$

The algorithm appends a 16-bit checksum b_L.

The checksum b_L is calculated as follows:

1 Treating each 16-bit word as an integer,
find $x = (b_0 + b_1 + b_2 + \ldots + b_{L-1})$ modulo $(2^{16} - 1)$

2 The checksum is then given by:

$b_L = -x$ modulo $(2^{16} - 1) = 2^{16} - 1 - x$ modulo $(2^{16} - 1)$

$b_L = 2^{16} - 1 - [(b_0 + b_1 + \ldots + b_{L-1})$ modulo $(2^{16} - 1)$

Thus, the headers must satisfy the following pattern:

$0 = (b_0 + b_1 + b_2 + \ldots + b_{L-1} + b_L)$ modulo $(2^{16} - 1)$

3 The checksum calculation is carried out in software using **one's complement arithmetic**.

Example

Internet Checksum can be calculated using **decimal arithmetic** or **binary arithmetic**. Here we show the decimal approach:

If b_0 = 1000 0001 1011 1100

and b_1 = 0100 1000 0101 1010

What will be their Internet Checksum b_2?

Using mod $2^{16} - 1 = 65535$ arithmetic,

- b_0 = 1000 0001 1011 11002 = 33212
- b_1 = 0100 1000 0101 10102 = 18522
- $b_0 + b_1 = (33212 + 18522) \bmod 65535 = 51734$
- $b_2 = - 51734 \bmod 65535 = 65535-51734$

  = 13801 = 0011 0101 1110 10012

Checksum Calculation: Method 2

Here we use **IP header Checksum** as an example:

Let the IP header consist of L, 16-bit words,

$$b_0, b_1, b_2, \ldots, b_{L-1}$$

The algorithm appends a 16-bit checksum b_L.

The checksum b_L is calculated as follows:

1. Add the 16-bit words using binary addition. Note that if an overflow occurs, a carryout from the most significant bit needs to be added to the result, which is known as wraparound.
2. The checksum is then given by the one's complement of the sum.

Internet Checksum Example

Note

When adding numbers, a carryout from the most significant bit needs to be added to the result.

Example: Add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Example

Method 2

b_0: 0000000000000001

b_1: 0000001000000011

b_2: 0000010000000101

b_3: 0000011000000111

b_4: 0000100000001001

0001010000011001

(Note that no wraparound has occurred.)

checksum = the one's complement of the sum

= 11101011 11100110

Method 1

1

515

1029

1543

2057

5145

checksum = -5145 (mod 65535)

= 60390 (mod 65535)

= 11101011 11100110

Internet Checksum: Undetectable errors

Internet Checksum: Undetectable errors

The simplest form of checksum, which adds up the bytes in the data to form a sum value, cannot detect a number of types of errors. In particular, such checksum value is not changed by:

- Reordering of the 16-bit word in the message block.
- Inserting or deleting zero-valued 16-bit word.
- Multiple errors that cancel the effect.

Reason?

Value is considered but order is not considered.

Exercise

Compute the checksum b_3 for the following data that consists three 8-bit words b_0, b_1 and b_2. The sender sends b_0, b_1 and b_2, and b_3 (checksum) to a remote receiver. If the receiver receives the four 8-bit words b_0, b_1, b_2 and b_3, verify that the new checksum for these four 8-bit words is zero. Compute first using the binary approach and then repeat the computation using the modular approach.

b_0: 00001111

b_1: 10001000

b_2: 11110000

Solution (Binary Approach)

Transmitting side

b_0:	00001111	
b_1:	10001000	

$b_0 + b_1$:	10010111	- no wraparound
b_2:	11110000	

	110000111	- wraparound
	1	

$b_0 + b_1 + b_2$:	10001000	- no wraparound
Cheksum b_3:	01110111	- one's complement

Solution (Binary Approach)

Receiver side

b_0:	00001111
b_1:	10001000
$b_0 + b_1$:	10010111
b_2:	11110000
	110000111
	1
$b_0 + b_1 + b_2$:	10001000
Cheksum b_3:	01110111

$b_0 + b_1 + b_2$:	10001000
b_3:	01110111
$b_0 + b_1 + b_2 + b_3$:	11111111
CS (1's com):	00000000

Solution (Modulo Approach)

Transmitting side

b_0:	00001111	=	15
b_1:	10001000	=	136
b_2:	11110000	=	240

			391

Checksum = b_3 = -391 (mod 255) = 119 (mod 255)

(Note that 119 is equal to 01110111 in binary.)

Receiver side

Sum = 15 + 136 + 240 + 119 = 510

Checksum = -510 (mod 255) = 0 → No error!

Summary

Summary

Key points discussed in this topic:

- Checksum is an error-detecting technique that can be applied to a message of any length. It is mostly used at the network and transport layer.
- In modular mathematics, we are only interested in the remainder of a division. This is called the modulo operator.
- Internet Checksum can be calculated using decimal arithmetic or binary arithmetic.
- The simplest form of checksum, which adds up the bytes in the data to form a sum value, cannot detect a number of types of errors. In particular, such checksum value is not changed by:
    - Reordering of the 16-bit word in the message block
    - Inserting or deleting zero-valued 16-bit word
    - Multiple errors that cancel the effect

○ The checksum for an IP packet only covers the IP header; however, the checksum for a TCP or UDP segment covers the entire segment; i.e., the header as well as the payload.

○ The checksum for IP, TCP and UDP operates on 16-bit words.