

Introduction to Web Science

Assignment 2

PD Dr. Matthias Thimm

thimm@uni-koblenz.de

Ipek Baris Schlicht

ibaris@uni-koblenz.de

Kenneth Skiba

kennethskiba@uni-koblenz.de

Institute of Web Science and Technologies

Department of Computer Science

University of Koblenz-Landau

Submission until: 24.11.2020, CEST 23:59

Team: Bravo

Members:

Gaurav Kumar (220200656)

Pavithree Shetty (220200661)

Nisha Sharma (220202359)

1 Frame Check Sequence

24 points

1.1 Checksum

11 points

Calculate the *checksum* of the hexadecimal message `ab35 c3d1 5231 30ed` using the steps described in Chapters 3.3.2 and 5.2.2 in the textbook *Computer networking: A top-down approach* from Kurose and Ross¹. How does the send messages looks like?

Please write down every calculation step.

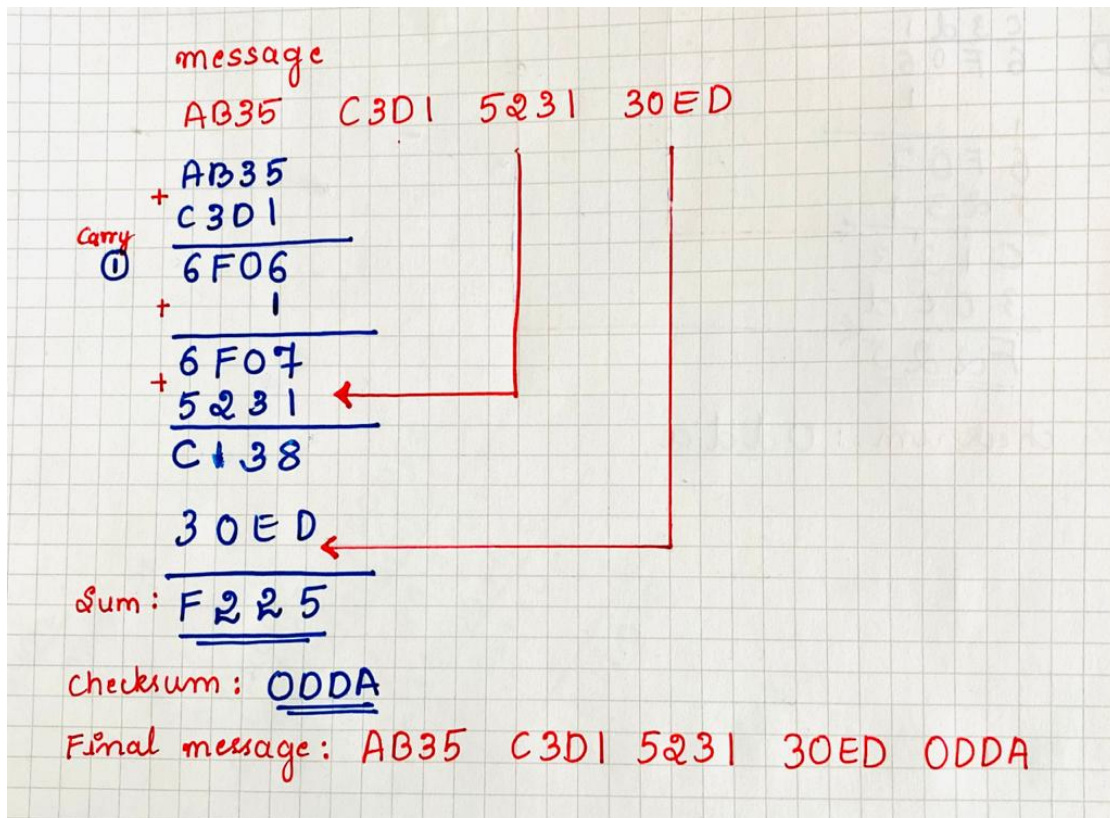


Figure 1: CheckSum

1.2 Cyclic Redundancy Check

13 points

Calculate the *CRC* bits using the *CRC* algorithm, as described in Chapter 5.2.3 in the textbook *Computer networking: A top-down approach* from Kurose and Ross¹ for the data $D = 10001110$ and the generator $g = 1010$. How does the send messages looks like?

Please write down every calculation step.

¹Computer networking: A top-down approach; James F. Kurose and Keith W. Ross; 6th Edition; Pearson Education India; 2013

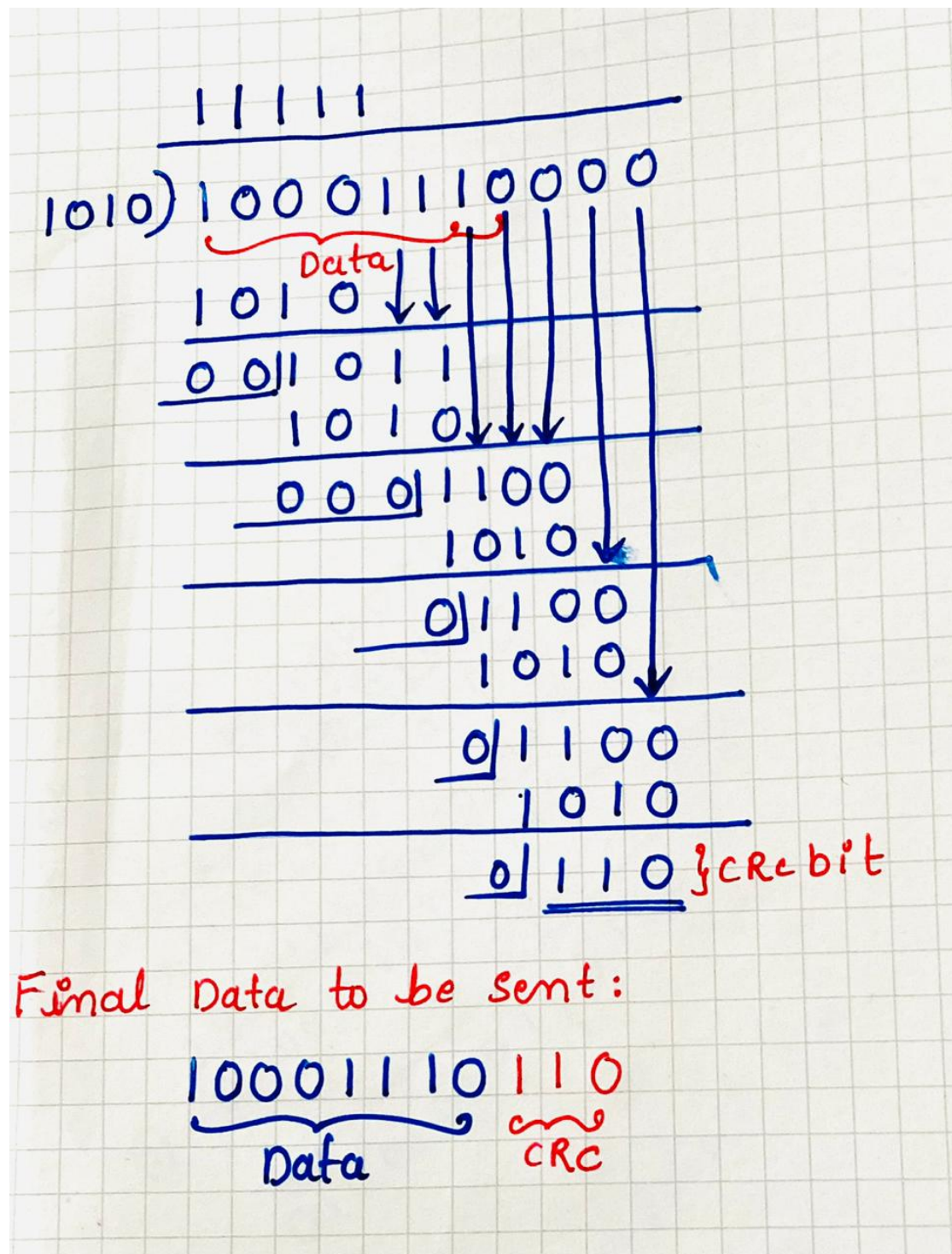


Figure 2: Cyclic Redundancy Check

2 Python Programming

25 points

2.1 Checksum

15 points

Write a method that computes and returns the checksum of a message. The input message contains four segments and each segment represents a 8-bit binary data.

1. In the method, make sure that each step is printed out in the console. For example:

```
1:      check_sum = calculate_checksum(message)
2:      console >> var 1 00000000
3:                      var 2 10011001
4:                      sum   10011001
5:                      ...
6:                      checksum 10001001
```

2. Fill out the following table with the checksum values that are computed by the method.

Messages	Checksum
[0b01101001,0b00100000,0b01101100,0b01101111]	
[0b01110110,0b01100101,0b00100000,0b01110111]	
[0b01100101,0b01100010,0b00100000,0b01110011]	

Table 1: Inputs for the checksum method

3. Write another method that validates or rejects given data and checksum. For the corrupted data, the method should output an error message.

```
1:      validate_data(message , check_sum)
2:      console >> Corrupted data
```

2.2 TCP Client Server

10 points

In this task you will simulate the checksum operation. You will write a sender (`sender.py`) and a receiver (`receiver.py`). The communication between them will be with IPv4 and TCP.

1. `sender.py` generates random messages as shown Table 1. When it gets four segments of messages, it calculates the checksum of a message with the method you will implement for Assignment 2.1 and send them to the receiver.
2. The `receiver.py` performs checksum validation with the method you will implement for Assignment 2.1 and sends the result of the validation to the sender.
3. When you execute `python receiver.py` from the console, you should print out the flow of receiver similarly:

```
1:      Connection from ('$SERVER$: $PORT$').
2:      Got the following message [219, 118, 233, 178] and 17 as
3:      checksum from the sender
4:      var 1  00000000
5:      var 2  11011011
6:          sum 11011011
7:      var 1  11011011
8:      var 2  01110110
9:          sum 01010010
10:     var 1  01010010
11:     var 2  11101001
12:         sum 00111100
13:     var 1  00111100
14:     var 2  10110010
15:         sum 11101110
16:     Got the following sum 11111111 after the validation
17:     ...
```

4. When you execute `python sender.py` from the console, you should print out the flow of sender similarly:

```
1:      connected to tmps-$$$SERVER$: $PORT$
2:      var 1  00000000
3:      var 2  11011011
4:          sum 11011011
5:      var 1  11011011
6:      var 2  01110110
7:          sum 01010010
8:      var 1  01010010
9:      var 2  11101001
10:         sum 00111100
11:     var 1  00111100
12:     var 2  10110010
13:         sum 11101110
14:     checksum 00010001
15: Response from the receiver b'Receiver Message:
16: Data is correctly received and checksum is 00010001'
17: ...
```

For the programming tasks, you can use the following libraries: `pickle`, `random`, `socket`. Make sure that the messages between sender and receiver are properly printed.

Along with your code, send the print screens of the terminals.

3 IPV4 and TCP

6 points

1a) 0xFF24C630 :

- Not a valid IPv4 address.
- IPv4 address are divided into following classes and the network id follows a certain range mentioned below:
Class A : Range(0-127)
Class B : Range(128-191)
Class C : Range(192 - 223)
Class D : Range(224 - 239)
Class E is reserved .
- As the above address does not belong to any class it is an invalid IPv4 address.

1b) 198.33.0.255 :

- Not a valid IPv4 address as it is the Direct Broadcast address
- The above address belongs to Class c, where Network id is 27 bits and Host id is 8 bits.
- The network id would be 198.33.0.0
The Broadcast address would be 198.33.0.255 both are not used as computer IP addresses

1c) 48.96.258.27 :

- Not a valid address
- IPv4 is a 32-bit address where each octet is 8 bits.
- The minimum value is 0 (00000000) and the maximum value is 255 (11111111)
- In the above IP address the value of third octet is 258 which is out of the range 0-255. Therefore it is not a valid IP address.

2a) TCP is preferred for video streaming if the speed of transmission is important.

- False Statement
- In video streaming if few packets are lost during transmission the devices using TCP protocol will then have to wait for the retransmission of those packets before receiving any new data. This is achieved by three-way handshake using SYN and ACK bits.
- Therefore buffering of unacknowledged packets causes delay and speed of transmission becomes an issue.
- Therefore TCP is not preferred for video streaming if the speed of transmission is important

2b) TCP is a reliable and stateless protocol which splits application data into equal size of the packets.

- False Statement
- TCP is not stateless protocol rather it is a stateful protocol.
- Because a connection is established between the host and receiver through three-way handshake before transmission of data. Adding to this both the host and the receiver maintain session information till the connection termination.
- TCP is reliable as it ensures data delivers in order not damaged and duplicated

2c) TCP controls the flow control of the networks having different speed of transmission

- True statement

4 Tree way handshake**12 Points**

Explain with a few sentences the three steps of the process of a TCP three-way handshake.

5 DNS

13 Points

For the DNS structure shown in Figure 3 answer the following questions:

1. What is the root?
2. What are the TLD?
3. What is the FQDN for the point *west*.
4. What is the domain of *google*.
5. What is the zone of *com*.

5.1 Solution:

1. In the given DNS structure root is represented as : **(.) Root**
2. In the given DNS structure the TLD(Top level Domains) are - **"com" and "de"**
3. In the given DNS structure the FQDN(Fully qualified Domain name) for the point *west* is : **west.uni-koblenz.de**
4. In the given DNS structure the domain of *google* is: **com**
5. In the given DNS structure the zone of *com* covers: **google and yahoo** in it.

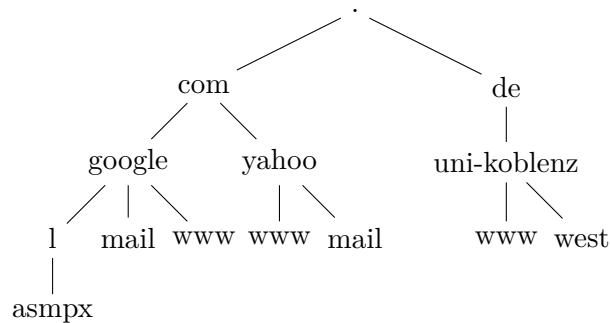


Figure 3: DNS structure