

Introduction to Web Science

Assignment 2

Prof. Dr. Steffen Staab

staab@uni-koblenz.de

René Pickhardt

rpickhardt@uni-koblenz.de

Korok Sengupta

koroksengupta@uni-koblenz.de

Institute of Web Science and Technologies

Department of Computer Science

University of Koblenz-Landau

Submission until: November 9, 2016, 10:00 a.m.

Tutorial on: November 11th, 2016, 12:00 p.m.

The main objective of this assignment is for you to use different tools with which you can understand the network that you are connected to or you are connecting to in a better sense. These tasks are not always specific to “Introduction to Web Science”. For all the assignment questions that require you to write a code, make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.

Group name: mike

Group members: Anish Girijashivaraj, Shohel Ahamad, Slobodan Kocevski

1 IP Packet (5 Points)

Consider the IPv4 packet that is received as:

4500 062A 42A1 8001 4210 XXXX C0A8 0001 C0A8 0003

Consider XXXX to be the check sum field that needs to be sent with the packet.

Please provide a step-by-step process for calculating the "Check Sum".

Answer:

1. Calculating the sum of all the bits in the IPv4 package, except for the Check Sum field.
 $4500 + 062A + 42A1 + 8001 + 4210 + C0A8 + 0001 + C0A8 + 0003 = 2\ D130$
2. Convert the result (2 D130) to binary: 0010 1101 0001 0011 0000
3. In order to get 16 bits value, we need the first 4 bits which are the carry, to add them to the rest of the value:
 $0010 + 1101\ 0001\ 0011\ 0000 = 1101\ 0001\ 0011\ 0010$
4. Next step is to find a compliment of the 16bit value. This can be done by flipping all zeros to ones and all ones to zeros.
1101 0001 0011 0010
0010 1110 1100 1101 - Compliment
5. Final step is to find the hex equivalent of the binary number 0010 1110 1100 1101 which is 2ECD

After calculating the "Check Sum", the IPv4 packet is:

4500 062A 42A1 8001 4210 2ECD C0A8 0001 C0A8 0003

2 Routing Algorithm (10 Points)

UPDATE. The bold fonted numbers have been updated on Monday Nov. 7th. (If you already have done so feel free to use the old numbers. But the solution with the old version will be more complex than the solution with the updated numbers.)

You have seen how routing tables can be used to see how the packets are transferred across different networks. Using the routing tables below of Router 1, 2 and 3:

1. Draw the network [6 points]
2. Find the shortest path of sending information from 67.68.2.10 network to 25.30.3.13 network [4 points]

Table 1: Router 1

Destination	Next Hop	Interface
67.0.0.0	67.68.3.1	eth 0
62.0.0.0	62.4.31.7	eth 1
88.0.0.0	88.4.32.6	eth 2
141. 71 .0.0	141. 71 .20.1	eth 3
26.0.0.0	141.71.26.3	eth 3
156.3 .0.0	141.71.26.3	eth 3
205. 30.7 .0	141.71.26.3	eth 3
25.0.0.0	88.6.32.1	eth 2
121.0.0.0	88.6.32.1	eth 2

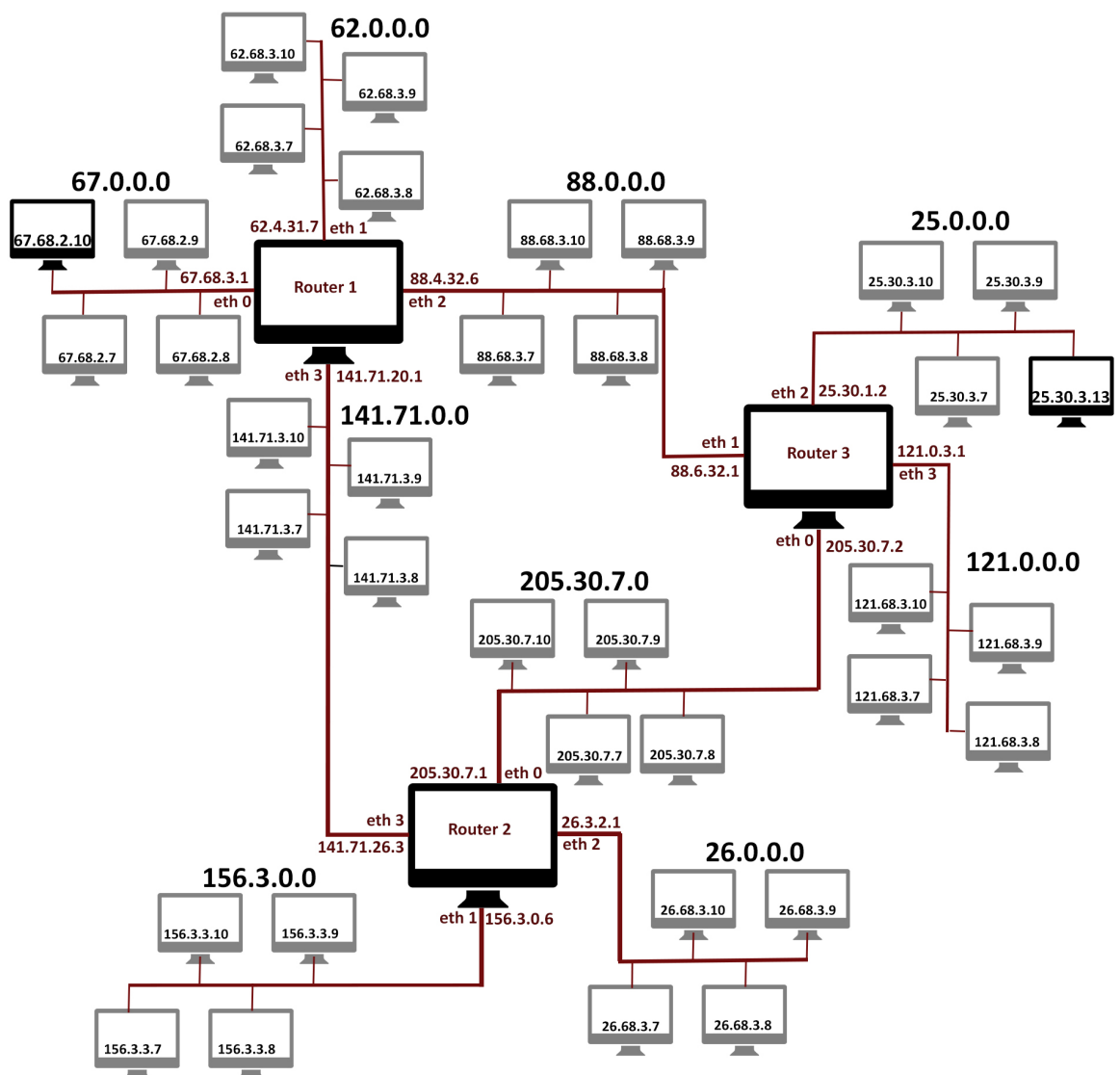
Table 2: Router 2

Destination	Next Hop	Interface
141. 71 .0.0	141.71.26.3	eth 3
205. 30.7 .0	205. 30.7 .1	eth 0
26.0.0.0	26.3.2.1	eth 2
156. 3 .0.0	156.3.0.6	eth 1
67.0.0.0	141. 71 .20.1	eth 3
62.0.0.0	141. 71 .20.1	eth 3
88.0.0.0	141. 71 .20.1	eth 3
25.0.0.0	205.30.7.2	eth 0
121.0.0.0	205.30.7.2	eth 0

Answer:

Table 3: Router 3

Destination	Next Hop	Interface
205.30.7.0	205.30.7.2	eth 0
88.0.0.0	88.6.32.1	eth 1
25.0.0.0	25.30.1.2	eth 2
121.0.0.0	121.0.3.1	eth 3
156.3.0.0	205.30.7.1	eth 0
26.0.0.0	205.30.7.1	eth 0
141.0.0.0	205.30.7.1	eth 0
67.0.0.0	88.4.32.6	eth 1
62.0.0.0	88.4.32.6	eth 1



2. The shortest way of sending information from 67.68.2.10 to 25.30.3.13 is going through eth2 of Router 1.

From the routing table of Router 1 we can see that there is only one data on how to get from Router 1 to the network 25.0.0.0 where is located computer with ip address 25.30.3.13 and that is through ethernet interface 2. The next hop will be 88.6.32.1.

The packet goes from 67.68.2.10 to Router 1, then using eth2, the packet is sent to 88.6.32.1 to Router 3, where Router 3 knows from his Routing table that he is responsible for network 25.30.3.13.

3 Sliding Window Protocol (10 Points)

Sliding window algorithm, which allows a sender to have more than one unacknowledged packet "in flight" at a time, improves network throughput.

Let us consider you have 2 Wide Area Networks. One with a bandwidth of 10 Mbps (Delay of 20 ms) and the other with 1 Mbps (Delay of 30 ms) . If a packet is considered to be of size 10kb. Calculate the window size of number of packets necessary for Sliding Window Protocol. [5 points]

Since you now understand the concept of Window Size for Sliding Window Protocol and how to calculate it, consider a window size of 3 packets and you have 7 packets to send. Draw the process of **Selective Repeat Sliding Window Protocol** where in the 3rd packet from the sender is lost while transmission. Show diagrammatically how the system reacts when a packet is not received and how it recuperates from that scenario. [5 points]

Answer:

In order to find the window size of number of packets we multiply the bandwidth with the delay and we divide the result with the size of the packet.

N number of packets = (Bandwidth*delay)/size of the packet.

1. 10Mbps bandwidth, 20ms delay, 10kb packet size

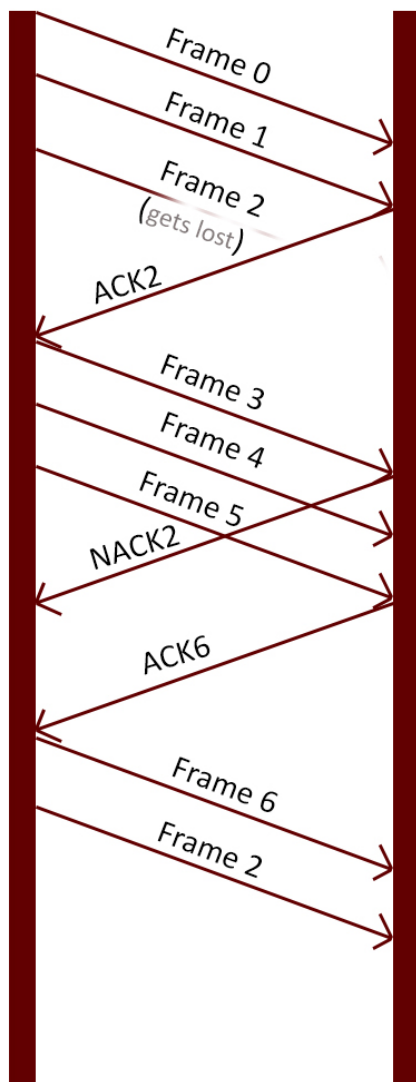
$$N = (10\text{Mbps} * 20\text{ms}) / 10\text{kb}$$

$$N = (10 * 1000\text{kbps} * 0.02\text{s}) / 10\text{kb} = 20$$

2. 1Mbps with bandwidth, 30ms delay, 10kb packet size

$$N = (1\text{Mbps} * 30\text{ms}) / 10\text{kb}$$

$$N = (1 * 1000\text{kbps} * 0.03\text{s}) / 10\text{kb} = 3$$

SENDER**RECEIVER**

When three frames are sent, the receiver sends acknowledgment about the received frames. When he noticed that frame number 2 is missing, he sends acknowledgment to the sender about the missing frame. After the sender sends all of the remaining frames, then he starts sending the missing frame.

4 TCP Client Server (10 Points)

Use the information from the [socket](#) documentation and create: [4 points]

1. a simple TCP Server that listens to a
2. Client

Note: Please use port 8080 for communication on `localhost` for client server communication.

Given below are the following points that your client and server must perform: [6 points]

1. The *Client* side asks the user to input their name, age & *matrikelnummer* which is then sent to the server all together.
2. Develop a protocol for sending these three information and subsequently receiving each of the information in three different lines as mentioned in the below format. Provide reasons for the protocol you implemented.
3. Format the output in a readable format as:
Name: Korok Sengupta;
Age: 29;
Matrikelnummer: 21223ert56

Provide a snapshot of the results along with the code.

Answer:

Server.py

```
1: import socket
2: import json
3:
4: #creating socket
5: PORT = 8080
6: s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7: s.bind(('localhost', PORT))    #connecting the port with the localhost machine
8: s.listen(1) # server listens to one client
9:
10: conn, addr = s.accept() # accepting the connection
11: data = conn.recv(1024) # receives data
12:
13: #converting the type of 'data' from byte to string, and then to dictionary
14: ss = data.decode()
```

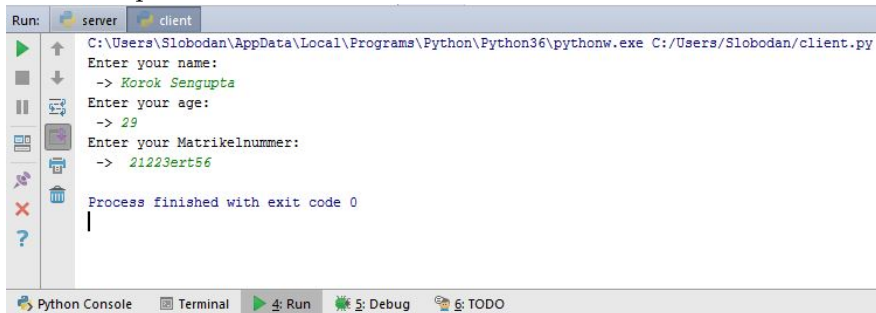


```
15: user = json.loads(ss)
16:
17: print("Name: " + user['name'])          #printing user's data while accessing specif
18: print("Age: " + user['age'])
19: print("Matrikelnummer: " + user['Matnummer'])
20:
21: conn.close()                          #closing the connection of the socket
```

Client.py

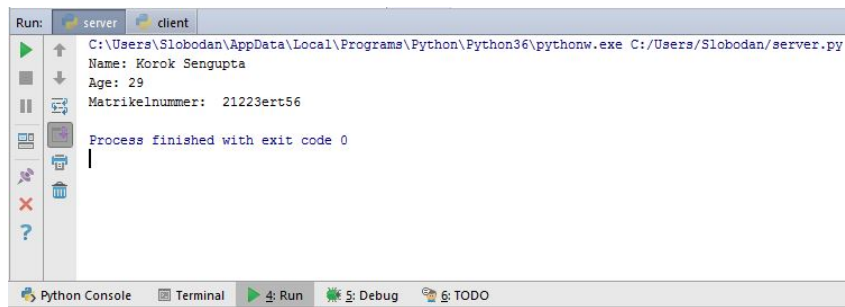
```
1: import socket
2:
3: #creating the socket
4: PORT = 8080
5: s = socket.socket()
6: s.connect(('localhost', PORT))
7:
8: #User enters data
9: print("Enter your name: ")
10: name = input(" -> ")
11: print("Enter your age: ")
12: age = input(" -> ")
13: print("Enter your Matrikelnummer: ")
14: matnum = input(" -> ")
15:
16: #creating json format string and sending the data
17: dataaaa = '{"name":"' + name + '","age":"' + age + '","Matnummer":"' + matnum +
18: s.sendall((dataaaa).encode('utf-8'))
19: s.close()                          #closing the socket connection with the server
```

1. Client output



```
Run: server client
C:\Users\Slobodan\AppData\Local\Programs\Python\Python36\pythonw.exe C:/Users/Slobodan/client.py
Enter your name:
-> Korok Sengupta
Enter your age:
-> 29
Enter your Matrikelnummer:
-> 21223ert56
Process finished with exit code 0
```

2. Server output



The screenshot shows a Python IDE window with a 'Run' tab active. The command bar shows the execution of `C:\Users\Slobodan\AppData\Local\Programs\Python\Python36\pythonw.exe C:/Users/Slobodan/server.py`. The output area displays the following text: `Name: Korok Sengupta`, `Age: 29`, `Matrikelnummer: 21223ert56`, and `Process finished with exit code 0`. The IDE interface includes a toolbar with icons for running, debugging, and other actions, and a bottom status bar with tabs for 'Python Console', 'Terminal', 'Run', 'Debug', and 'TODO'.

3. In this exercise we use JSON format of storing data and sending it from the client to the server side. User's data is sent as a string in a json format including the key parameter (name, age, matrikelnummer).

In this way we can be sure that the data is going to be shown as required. That is possible because we store each input of the user in three variable (key), so we have three separate variable, and it is easy later to show them in the required outcome. Other thing is distinguishing different data in one string. JSON allows us easy to split the string that we send over the socket without concerning about special cases, for example If the user has several names, or he has second name with 2 letter, or even if he puts a special sign in his name, everything is going to be categorized using indexes in the dictionary, which will assure us that the outcome will be accurate.

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment2/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use UTF-8 as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
 - Make sure you code has consistent [indentation](#).
 - Make sure you comment and document your code adequately in English.
 - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

L^AT_EX

Currently the code can only be build using [LuaLaTeX](#), so make sure you have that installed. If on Overleaf, go to settings and change the L^AT_EX engine to LuaLaTeX in case you encounter any error