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.

Delta Group Members

Oana Dumitrasc
odumitrasc@uni-koblenz.de

Alisa Becker
alisabecker@uni-koblenz.de

Omar Aly
oaly@uni-koblenz.de

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: 4500 = 0100 0101 0000 0000
062A = 0000 0110 0010 1010

1st Res = 0100 1011 0010 1010
42A1 = 0100 0010 1010 0001

2nd Res = 1000 1101 1100 1011
8001 = 1000 0000 0000 0001

10000 1101 1100 1100//1 odd bit (carry),add to the result to keep checksum in 16 bits.
so:

0000 1101 1100 1100
0000 0000 0000 0001

3rd Res = 0000 1101 1100 1101
4210 = 0100 0010 0001 0000

4th Res = 0100 1111 1101 1101
C0A8 = 1100 0000 1010 1000

10001 0000 1000 0101//1 odd bit (carry),add to the result to keep checksum in 16 bits.
so:

0001 0000 1000 0101
0000 0000 0000 0001

5th Res = 0001 0000 1000 0110
0001 = 0000 0000 0000 0001

6th Res = 0001 0000 1000 0111
C0A8 = 1100 0000 1010 1000

7th Res = 1101 0001 0010 1111
0003 = 0000 0000 0000 0011

---

final Res = 1101 0001 0011 0010//As a last step we need to do a one's compliment of it to obtain the checksum.

Checksum= 0010 1110 1100 1101
And in Hex decimal the Checksum (`XXXX`) = 2ECD

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

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
141.71.0.0	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

Answer: As the start computer has ip 67.68.2.10 so its in 67.0.0.0 network which is connected to "Router 1" and so its send there using the computer routing table(default route), then router 1 analyze the package to figure out its for 25.0.0.0 network so from routing table it sees that next hop is 88.6.32.1 which is the ip of another network card at "Router 3", then router 3 analyze the package to figure out its for 25.0.0.0 network so from routing table it sees that next hop is 25.30.1.2 which is the ip for the destination network but still not the destination ip so inside the network and from that computer which received the package it will route it through the network using its own routing table to finally arrive at 25.30.3.13

So it will look like this:
67.68.2.10 –> 67.68.3.1 –> 88.6.32.1 –> 25.30.1.2 –> 25.30.3.13

Figure 1 illustrates the network.

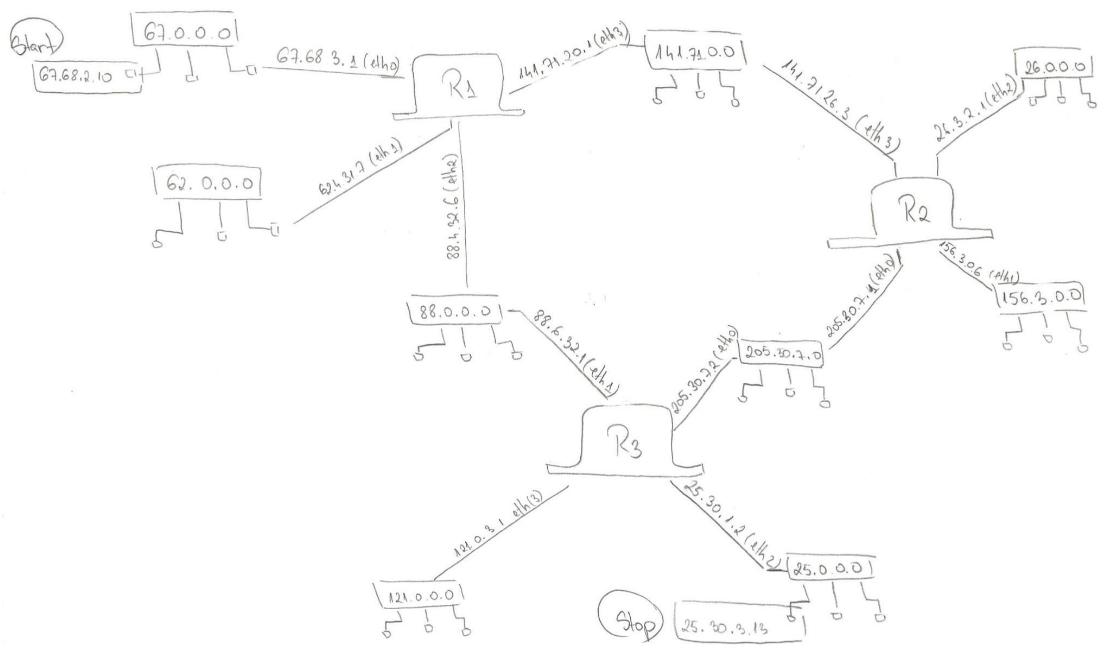


Figure 1: Network

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`]

Answer:
windows size = Bandwidth(bps) $\times$Round Trip delay Time
Round Trip delay Time = delay $\times 2$
So:
10 Mbps WAN:
BandWidth = 10×10^6
RTT = $20 \times 10^{-3} \times 2$
window size = $10000000 \times 0.04 = 400000$b
window size of number of packets = 400kb / 10kb (size of one packet) = 40 packet
1 Mbps WAN:
BandWidth = 1×10^6
RTT = $30 \times 10^{-3} \times 2$
window size = $1000000 \times 0.06 = 60000$b
window size of number of packets = 60kb / 10kb (size of one packet) = 6 packet

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:

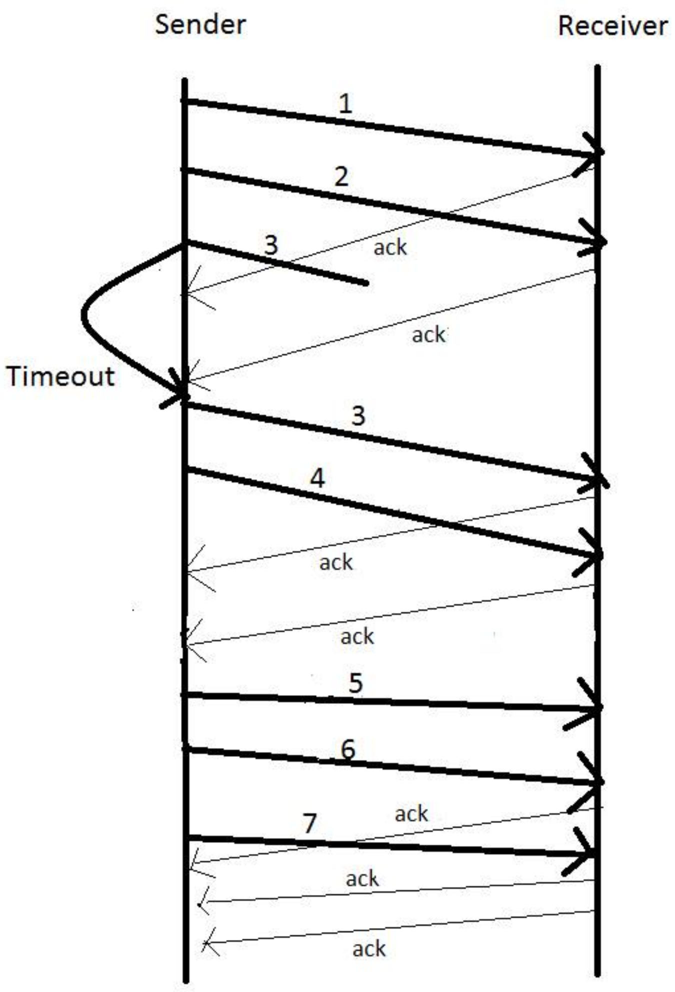


Figure 2: Sliding Window Protocol With Selective Repeat ARQ

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:

We chose to use JSON because it is easy to parse and to generate. Introducing our own delimiters or regex as our protocol would have been also a solution, but we found JSON to be the more elegant one. It is the most common data format used for (mostly asynchronous) browser/server communication so we found it proper to used for the task at hand.

Server script

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  6 18:49:04 2016
"""
import socket
import json

# Create a TCP/IP socket
sck = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```

# Bind the socket to the port
server_address = ('localhost', 8080)
print ('starting up on %s port %s' % server_address)
sck.bind(server_address)

# Listen for incoming connections,
#the passed value indicates the maximum number of connections
#to be created at once
sck.listen(1)

while True:
    # Wait for a connection
    connection, client_address = sck.accept()
    #try receiving data and with the finally option, its used for clearing action
    try:
        print ( 'connection from', client_address)
        while True:
            data = connection.recv(1024)
                        # if we got data process it
            #else break the loop of waiting for data.
            if data:
                final_data = json.loads(data.decode("utf-8"))
                print('Name:' + final_data['name']+ ';')
                print('Age:' + final_data['age']+ ';')
                print('Matrikelnummer:' + final_data['matrikelnummer'])

            else:
                break

    finally:
        # Clean up the connection
        connection.close()
```

Client script

```
# -*- coding: utf-8 -*-
"""
Created on Sun Nov  6 18:51:47 2016
"""

import socket
import json

# Create a TCP/IP socket
sck = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

```

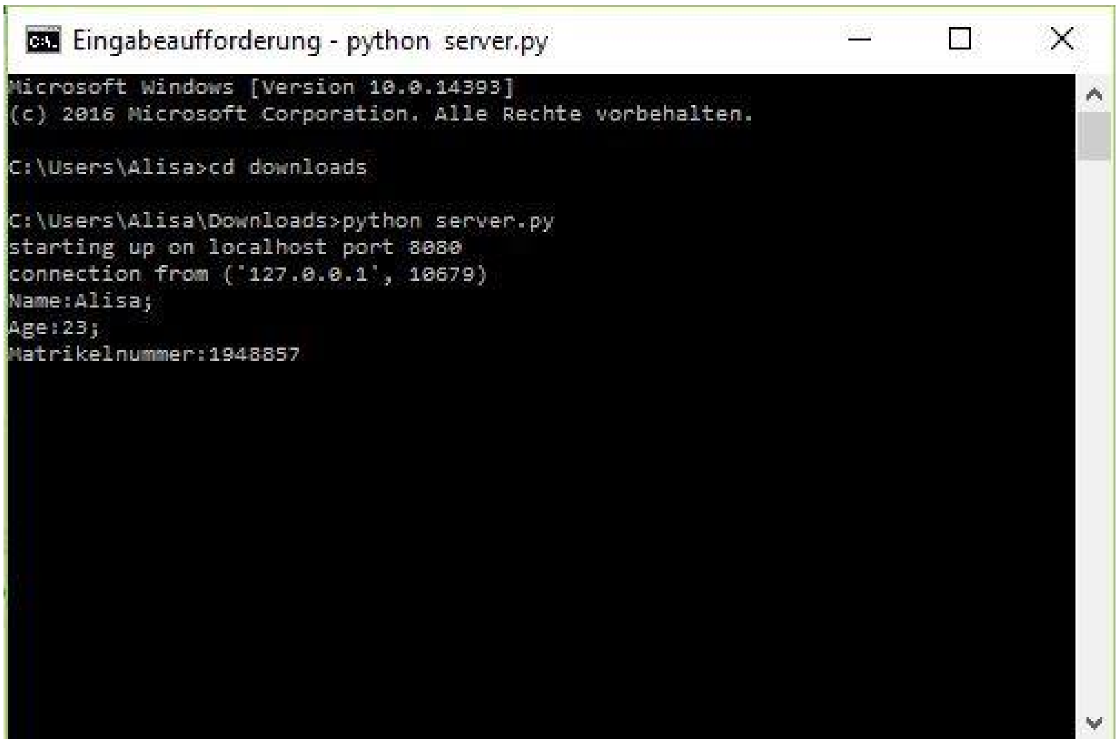
```
# Connect the socket to the port where the server is listening
server_address = ('localhost', 8080)
sck.connect(server_address)

#try sending data and with the finally option, its used for clearing action that
try:
    #Ask for input
    name = input("Name: ")
    age = input("Age: ")
    matrikelnummer = input("Matrikelnummer: ")

    # Build up JSON
    data = {}
    data['name'] = name
    data['age'] = age
    data['matrikelnummer'] = matrikelnummer
    json_data = json.dumps(data)

    # Send data
    sck.sendall(bytes(json_data,"UTF-8"))

finally:
        # Clean up the connection
    sck.close()
```



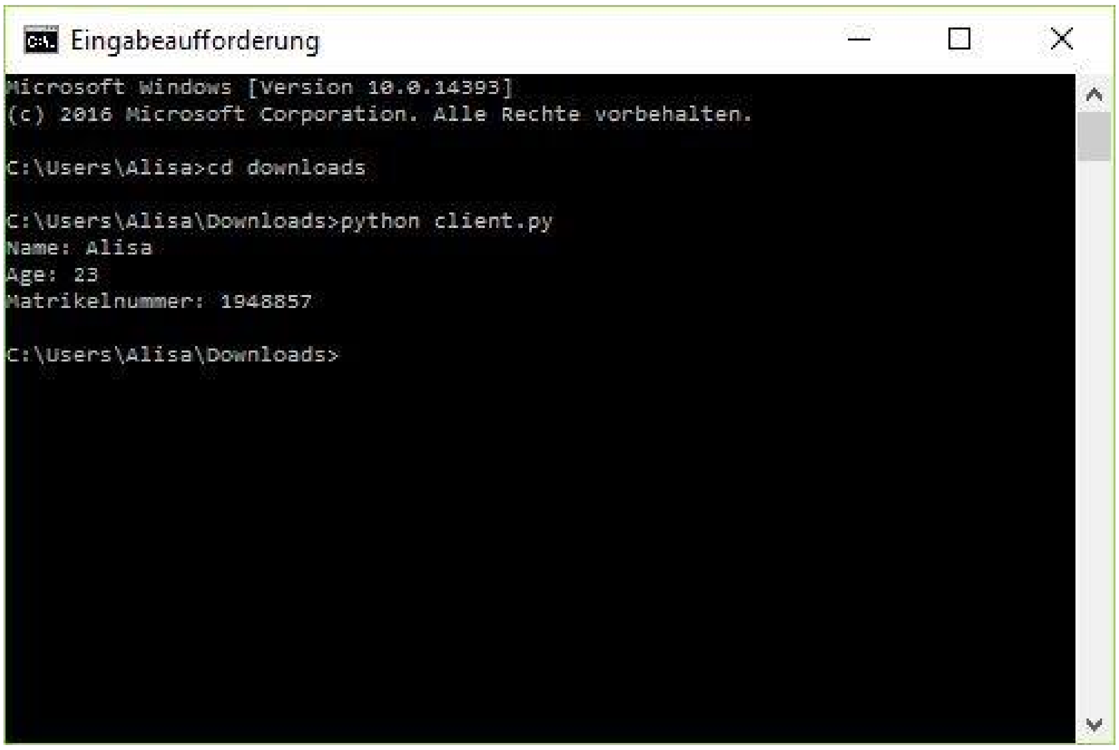
Eingabeaufforderung - python server.py

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Alisa>cd downloads

C:\Users\Alisa\Downloads>python server.py
starting up on localhost port 8080
connection from ('127.0.0.1', 10679)
Name:Alisa;
Age:23;
Matrikelnummer:1948857
```

Figure 3: Server Side



Eingabeaufforderung

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Alisa>cd downloads

C:\Users\Alisa\Downloads>python client.py
Name: Alisa
Age: 23
Matrikelnummer: 1948857

C:\Users\Alisa\Downloads>
```

Figure 4: Client Side

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".

LaTeX

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 LaTeXengine to `LuaLaTeX` in case you encounter any error