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.

This assignment was made by group quebec:

1. Daniel Kostic
2. Stefan Vujovic
3. Igor Fedotov

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:

The checksum is calculated by forming the ones' complement of the ones' complement sum of the header's 16-bit words. The result of summing the entire IP header, including checksum, should be zero if there is no corruption.

To calculate the checksum, we can first calculate the sum of each 16 bit value within the header, skipping only the checksum field itself:

`4500 + 062A + 42A1 + 8001 + 4210 + C0A8 + 0001 + C0A8 + 0003 = 2D130`

Then, we convert the 2D130 value from the hexadecimal to the binary system:

`2D130 = 0010 1101 0001 0011 0000`

The first 4 bits are the carry and will be added to the rest of the value:

`0010 + 1101 0001 0011 0000 = 1101 0001 0011 0010`

In this example the addition of the carry didn't itself generate a carry. If it had it would have been necessary to add that new carry back in as well.
Next, we flip every bit in that value, to obtain the checksum:

0010 1110 1100 1101

This is the "check sum" which equals **2ECD** when converted into the hexadecimal system.

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

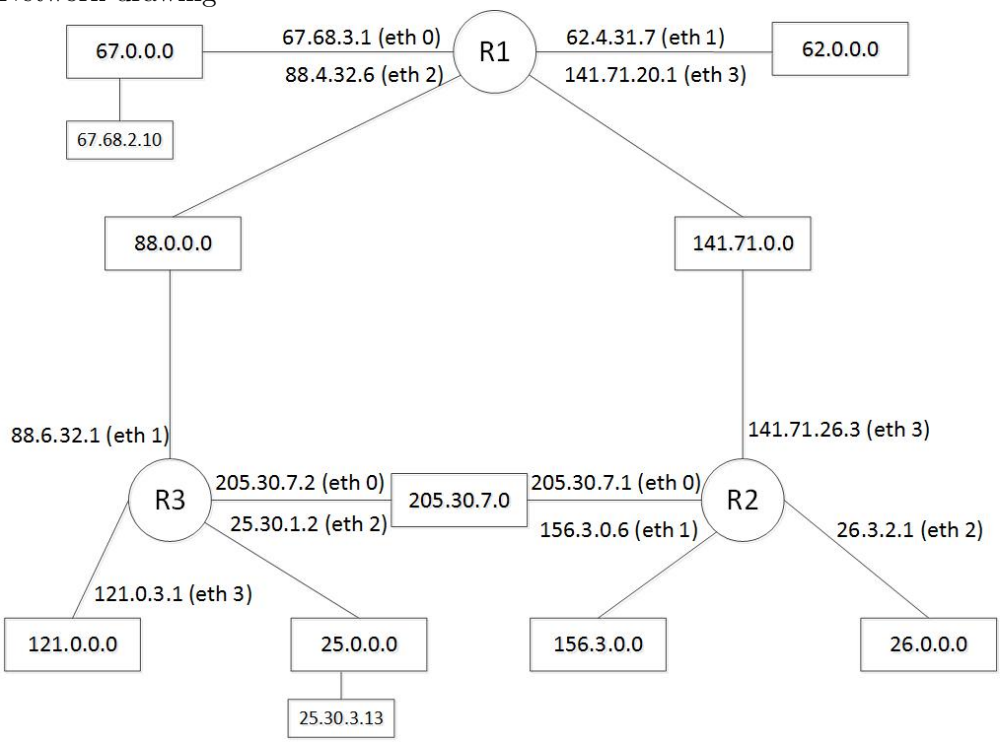
Table 3: Router 3

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

2.1 Answer:

Important! Table 3 has been updated to include routing for network 141.71, and not 141.0 (bug indicated in mailing list).

1. Network drawing

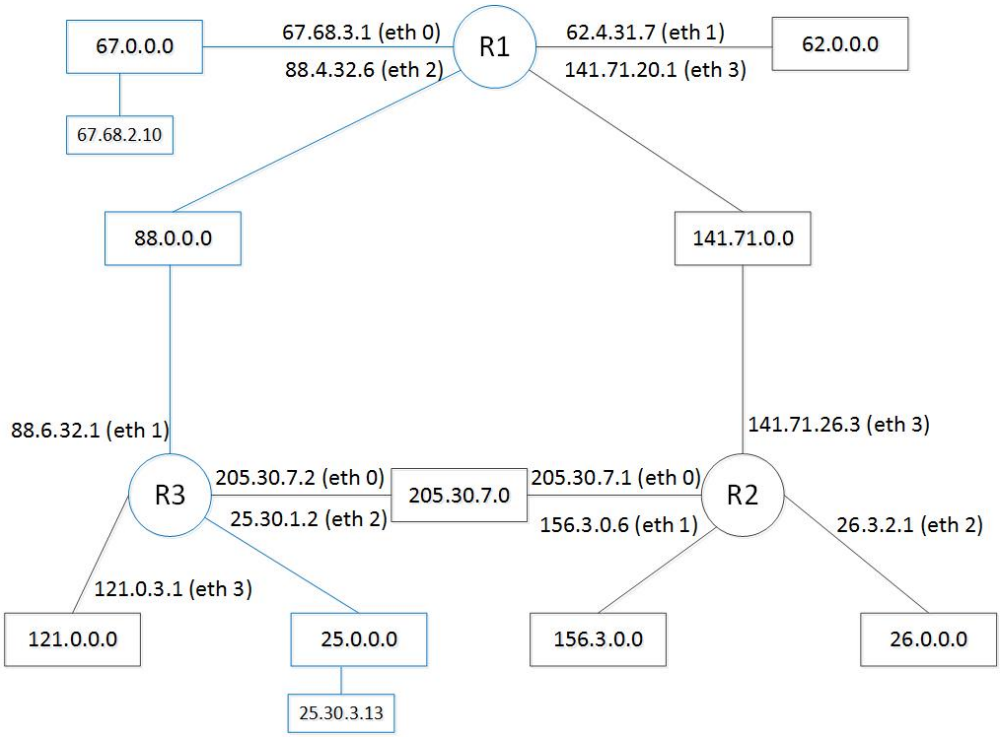


2. Shortest path

   The shortest path is highlighted in blue color on the following drawing. The path starts at the machine with an IP address 67.68.2.10 (from now on: PC1),

which is a part of the 67.0.0.0 network and it ends at the machine with the IP address 25.30.3.13 (from now on: PC2) which is part of the 25.0.0.0 network. In order for to communicate with machines outside the 67.0.0.0 network, PC1 needs to send the data to the router connected to this network. PC1 has the routers IP address 67.68.3.1, but it needs its MAC address in order to send data using the Ethernet protocol and therefore sends an ARP request to get it. When the data arrives to the router(R1), it looks up the routing table to find the next responsible machine (next hop) on the way to the final destination. In this case it is over the 88.0.0.0 network, and the responsible machine is the router (R3) with the IP address 88.6.32.1. Again, the router (R1) will send an ARP request and get the MAC address of the other router (R3), since both routers are connected to the same network 88.0.0.0. The router (R3) is connected to the network 25.0.0.0 which contains PC2 with the IP: 25.30.3.13. The router will send an ARP request, receive the MAC address for PC2 and send the data using Ethernet.

Path: 67.67.2.10 => 67.68.3.1 => 88.6.32.1 => 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`]

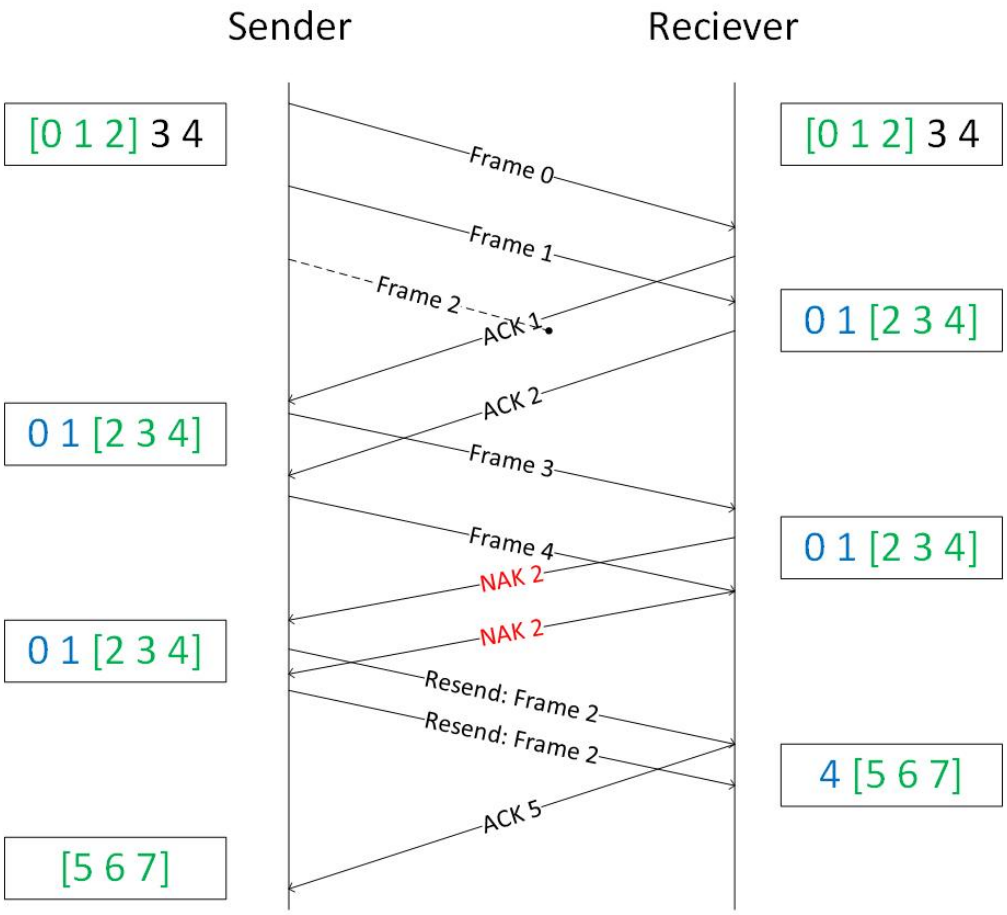
Since you now understand the concept of Window Size for Sliding Window Protocol and how to calculate it, consider a window size of 3 packets and you have 7 packets to send. Draw the process of `Selective Repeat Sliding Window Protocol` where in the 3rd packet from the sender is lost while transmission.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

Answer1: The window size should be large enough to fill the network path, so it is equal to Round trip time times bandwidth. The window size, expressed in number of packets, is window size divided by packet size.

$$
\begin{aligned}
B_1 &= 10Mbps \\
B_2 &= 1Mbps \\
D_1 &= 20ms \\
D_2 &= 30ms \\
packet &= 10Kb \\
W &= 2 * B * D \\
W_1 &= 2 * 10\frac{Mb}{s} * 20ms \\
W_1 &= 2 * 10000\frac{Kb}{s} * 0.02s \\
W_1 &= 400Kb \\
N_1 &= \frac{W_1}{packet} \\
N_1 &= 400Kb/10Kb = 40packets \\
W_2 &= 2 * 1\frac{Mb}{s} * 30ms \\
W_2 &= 2 * 1000\frac{Kb}{s} * 0.03s \\
W_2 &= 60Kb \\
N_2 &= \frac{W_1}{packet} \\
N_2 &= 60Kb/10Kb = 6packets
\end{aligned}
\tag{1}
$$

where:
W_i = window size for network i
B_i = Bandwidth for network i
D_i = Delay for network i
N_i = Window size in packets for network i
In conclusion, for the first network we need a window size of 40 packets, and for the second one a window size of 6 packets.

Answer2: If the third package is lost in transmission, when a forth package arrives, it will generate a NAK message, that will tell the sender to send the expected package again.The sender will send the missing package again, and wait for it to be ACKed, before proceeding further.


Sender
Reciever
[0 1 2] 3 4
[0 1 2] 3 4
Frame 0
Frame 1
Frame 2
ACK 1
0 1 [2 3 4]
0 1 [2 3 4]
ACK 2
Frame 3
Frame 4
0 1 [2 3 4]
NAK 2
NAK 2
0 1 [2 3 4]
Resend: Frame 2
Resend: Frame 2
4 [5 6 7]
ACK 5
[5 6 7]

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.

4.1 Answer:

We decided to transfer data in the form of a JSON object. The object contains 3 fields, ”name”, ”age” and ”matrikelnummer”. We think this is a better solution than sending a string, with values separated by some character, for example (”,”, ”@”, ”#”) because names can contain comas, and if we decided to add emails as values the same problem would occur if we used the ”@” symbol etc. This approach is also easier to comprehend by other developers. It is also very easy to implement, since Python dictionaries are a compatible data structure with JSON.

Client side code:

```
1  import socket
2  import json
3
4  name = input("What is your name? \n")
5  age = input("How old are you? \n")
```

```
matrikelnummer = input("What is your Martikelnummer? \n")

pack = {}

pack["name"] = name
pack["age"] = age
pack["matrikelnummer"] = matrikelnummer

pack_json = json.dumps(pack)

TCP_IP = '127.0.0.1'
TCP_PORT = 8080
#BUFFER_SIZE = 1024
MESSAGE = pack_json

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
s.send(MESSAGE.encode())
#data = s.recv(BUFFER_SIZE)
s.close()

#print "received data:", data
```

Server side code:

```
import socket
import json

TCP_IP = '127.0.0.1'
TCP_PORT = 8080
BUFFER_SIZE = 1024


s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#print "Socket initialized"
s.bind((TCP_IP, TCP_PORT))
s.listen(5)


#print "waiting for clients"
conn, addr = s.accept()

#print 'Connection address:', addr
while True:
    data = conn.recv(BUFFER_SIZE).decode()
    if not data: break
    pack = json.loads(data)
    print ("""Name: {};
Age: {};
Matrikelnummer: {};""".format(pack["name"],pack["age"], pack["matrikelnummer"]))
    #conn.send(data)  # echo
    #conn.close()
```

Output:

```
Name: Igor;
Age: 19;
Matrikelnummer: 1602994657;

Process finished with exit code 0
```

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