Introduction to Web Science

Assignment By Group "Charlie"

Rinku Chowdhury 216101118
M.Rauf Qureshi 216100729
Shreya Chatterjee 216100848

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.

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

42A1 -> 0100 0010 1010 0001

8001 -> 1000 0000 0000 0001

4210 -> 0100 0010 0001 0000

0000 -> 0000000000000000 Note that the checksum is set to zero since we are computing checksum

C0A8 -> 1100 0000 1010 1000

00001 -> 0000 0000 0000 0001

C0A8 -> 1100 0000 1010 1000

C0A8 -> 1100 0000 1010 1000

0003 -> 0000 0000 0000 0011

Now lets add these binary values one by one :

4500 -> 0100 0101 0000 0000

062A -> 0000 0110 0010 1010

4B2A -> 0100 1011 0010 1010 First result

4B2A -> 0100 1011 0010 1010 First result plus next 16-bit word.

42A1 -> 0100 0010 1010 0001

8DCB -> 1000 1101 1100 1011 Second result.

8DCB -> 1000 1101 1100 1011 Second result plus next 16-bit word

8001 -> 1000 0000 0000 0001

10DCD -> 10000 1101 1100 1100 Third result.

0DCD -> 0000 1101 1100 1101 One odd bit (carry), add that odd bit to the result as we need to keep the checksum in 16 bits.

0DCD -> 0000 1101 1100 1101 Third result plus next 16-bit word.

4210 -> 0100 0010 0001 0000

4FDD -> 0100 1111 1101 1101 Fourth result.

4FDD -> 0100 1111 1101 1101 Fourth result plus next 16-bit word

C0A8 -> 1100 0000 1010 1000

11085 -> 0001 0001 0000 1000 0101 Fifth result.

1086 -> 0001 0000 1000 0110 One odd bit (carry), add that odd bit to the result as we need to keep the checksum in 16 bits.

1086 -> 0001 0000 1000 0110 Fifth result plus next 16-bit word

0001 -> 0000 0000 0000 0001

1087 -> 0001 0000 1000 0111 Sixth result.

1087 -> 0001 0000 1000 0111 Sixth result plus next 16-bit word.

C0A8 -> 1100 0000 1010 1000

D12F -> 1101 0001 0010 1111 Seventh result.

D12F -> 1101 0001 0010 1111 Seventh result plus next 16-bit word.

0003 -> 0000 0000 0000 0011

D132 -> 1101 0001 0011 0010 Final result.
So now 1101 0001 0011 0010 is our final result of summing up all the 16 bit words in the header. As a last step we just need to do a one's compliment of it to obtain the checksum.

D132 -> 1101 0001 0011 0010 Final result.
2ECD Checksum

2 Routing Algorithm (10 Points)

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

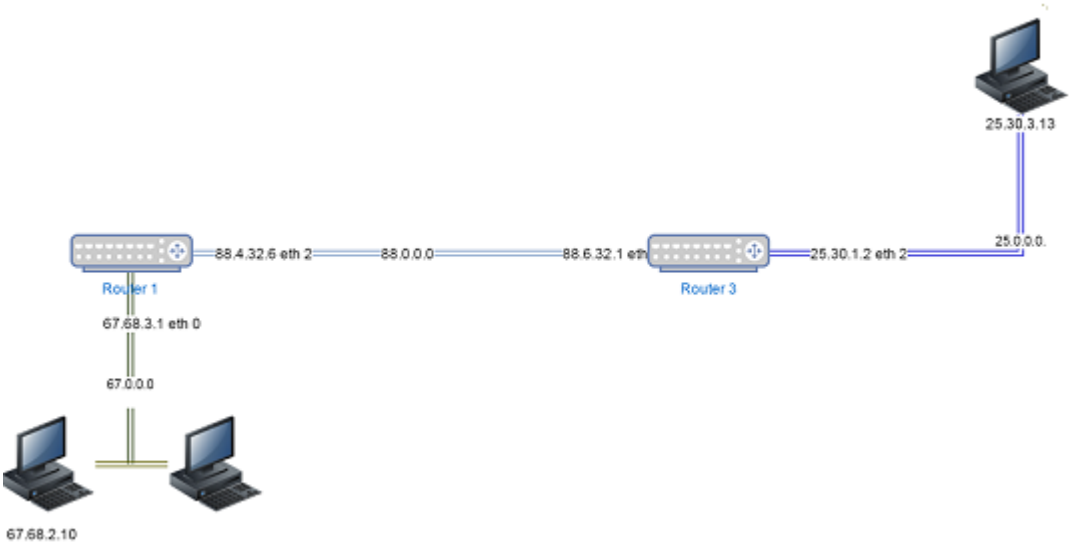
Destination	Next Hop	Interface
67.0.0.0	67.68.3.1	eth 0
62.0.0.0	62.4.31.7	eth 1
88.0.0.0	88.4.32.6	eth 2
141.0.0.0	141.30.20.1	eth 3
26.0.0.0	141.71.26.3	eth 3
150.0.0.0	141.71.26.3	eth 3
205.0.0.0	141.71.26.3	eth 3
25.0.0.0	88.6.32.1	eth 2
121.0.0.0	88.6.32.1	eth 2

Table 2: Router 2

Destination	Next Hop	Interface
141.0.0.0	141.71.26.3	eth 3
205.0.0.0	205.25.71.1	eth 0
26.0.0.0	26.3.2.1	eth 2
156.0.0.0	156.3.0.6	eth 1
67.0.0.0	141.30.20.1	eth 3
62.0.0.0	141.30.20.1	eth 3
88.0.0.0	141.30.20.1	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
205.0.0.0	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
156.0.0.0	205.25.7.1	eth 0
26.0.0.0	205.25.7.1	eth 0
141.0.0.0	205.25.7.1	eth 0
67.0.0.0	88.4.32.6	eth 1
62.0.0.0	88.4.32.6	eth 1



Finding the shortest path of sending information from 67.68.2.10 network to 25.30.3.13 network
Each router keeps track of its incident links:

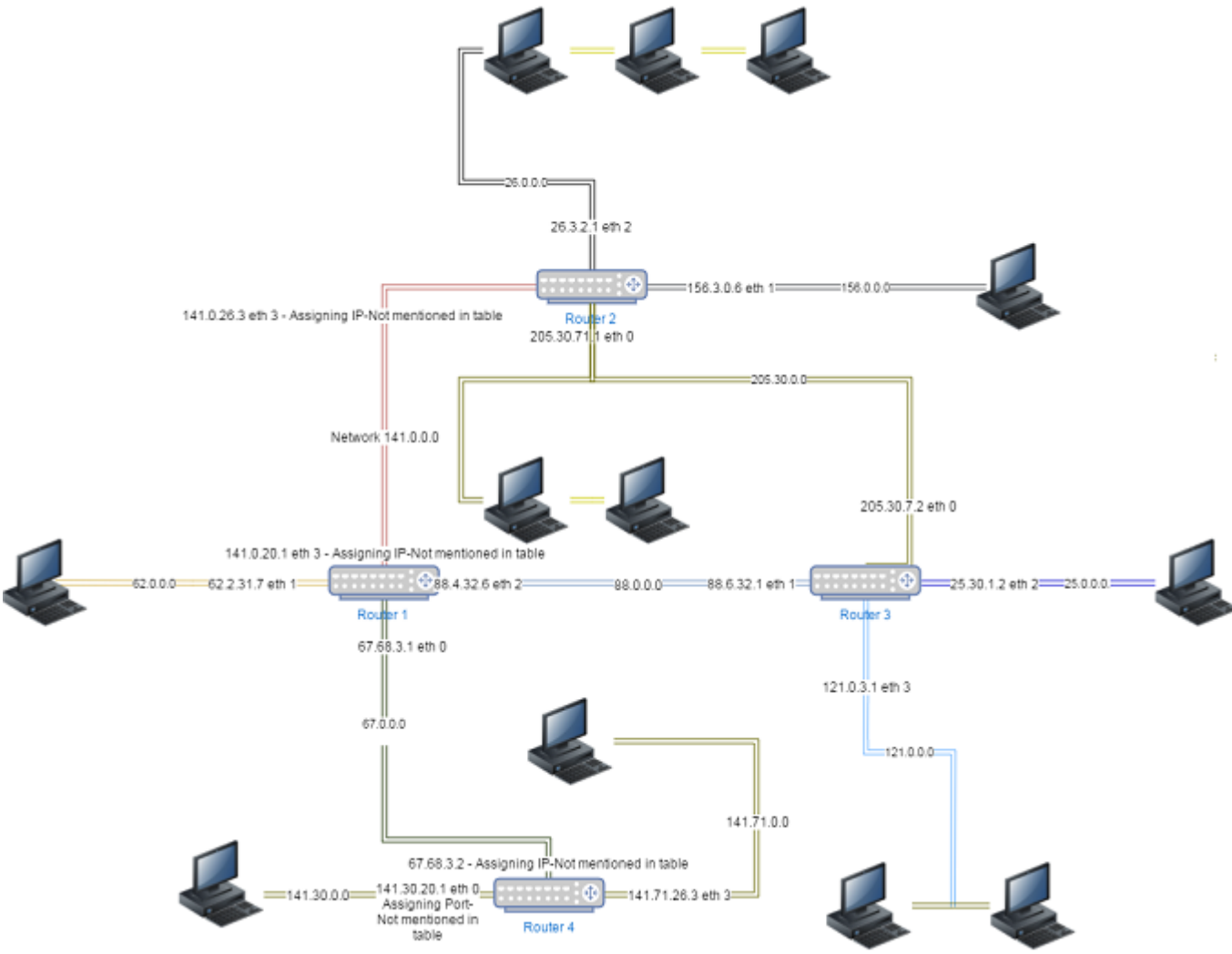
-Whether the link is up or down

-The cost on the link

Each router broadcasts the link state:

-To give every router a complete view of the graph

Each router runs Dijkstra's algorithm to compute the shortest paths and construct the forwarding table


26.0.0.0
26.3.2.1 eth 2
156.3.0.6 eth 1
156.0.0.0
141.0.26.3 eth 3 - Assigning IP-Not mentioned in table
Router 2
205.30.71.1 eth 0
205.30.0.0
Network 141.0.0.0
205.30.7.2 eth 0
141.0.20.1 eth 3 - Assigning IP-Not mentioned in table
62.0.0.0
62.2.31.7 eth 1
88.4.32.6 eth 2
88.0.0.0
88.6.32.1 eth 1
25.30.1.2 eth 2
25.0.0.0
Router 1
Router 3
67.68.3.1 eth 0
121.0.3.1 eth 3
67.0.0.0
121.0.0.0
141.71.0.0
67.68.3.2 - Assigning IP-Not mentioned in table
141.30.0.0
141.30.20.1 eth 0
Assigning Port-
Not mentioned in
table
141.71.26.3 eth 3
Router 4

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`]

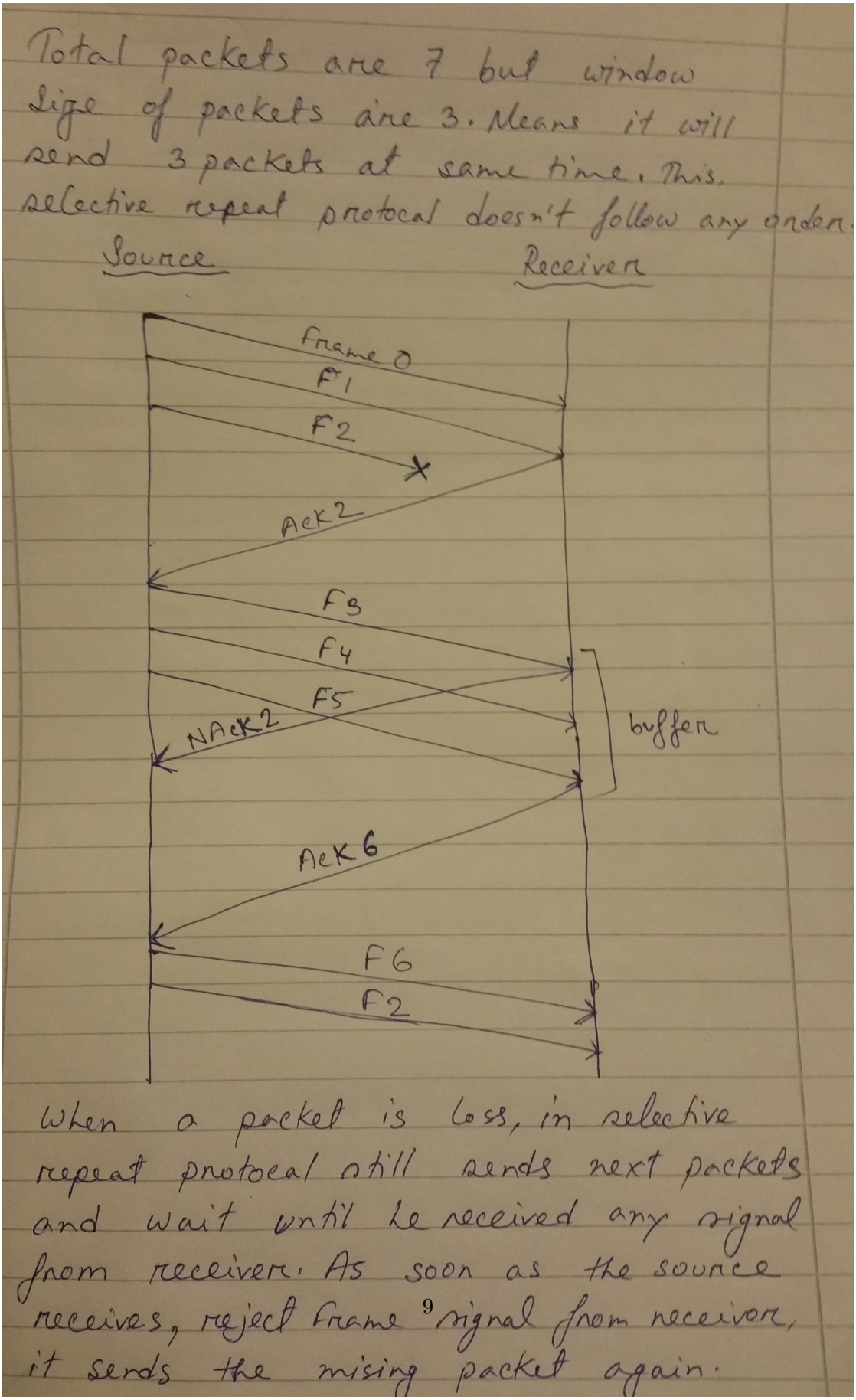
Since you now understand the concept of Window Size for Sliding Window Protocol and how to calculate it, consider a window size of 3 packets and you have 7 packets to send. Draw the process of `Selective Repeat Sliding Window Protocol` where in the 3rd packet from the sender is lost while transmission.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

Answer:
Given information for WAN 1 is followed:
Bandwidth = 10Mbps with delay 20ms
Packet size= 10kb
According to sliding window protocol,
N number of packets = (Bandwidth*delay)/size of the packet.
N=(10Mbps*20ms)/10kb
N=(10*1000kbps*0.02s)/10kb =20

Given information for WAN 2 is followed:
Bandwidth = 1Mbps with delay 30ms
Packet size= 10kb
According to sliding window protocol,
N number of packets = (Bandwidth*delay)/size of the packet.
N=(1Mbps*30ms)/10kb
N=(1*1000kbps*0.03s)/10kb =3

Selective Repeat Sliding Window Protocol

Total packets are 7 but window size of packets are 3. Means it will send 3 packets at same time. This selective repeat protocal doesn't follow any order.



When a packet is loss, in selective repeat protocal still sends next packets and wait until he received any signal from receiver. As soon as the source receives, reject frame 9 signal from receiver, it sends the mising packet again.

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 side :

```
import socket
import sys
import re
HOST = 'localhost'
PORT = 8080
_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
_socket.bind((HOST, PORT))
except socket.error as msg:
print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
sys.exit()

#Start listening
_socket.listen(10)

conn, addr = _socket.accept()
while 1:
reply= conn.recv(1024)
if len(reply)>1:
a=reply.split('::')
data= 'Name: '+str(a[0])+';\n' + 'Age: '+str(a[1])+';\n' + 'Matrikelnummer: '+str(a[2])+
conn.sendall(data)
print str(data)+'/n'

conn.close()

_socket.close()
```

Client Side:

```
import socket
import sys
import re
HOST = 'localhost'
PORT = 8080
_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
_socket.connect((HOST, PORT))
except socket.error as msg:
print 'Bind failed. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
sys.exit()

#Start listening



while 1:
#wait to accept a connection - blocking call
print 'Please enter that Name'
_Name=raw_input()
print 'Please enter that Age'
_Age=raw_input()
print 'Please enter that Name'
_Matrikelnummer=raw_input()
_socket.sendall(_Name+'::'+_Age+'::'+_Matrikelnummer)
reply= _socket.recv(1024)
lk = "\n".join(reply.split("\r\n"))
print lk


conn.close()

_socket.close()
```

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignmentBy Group "Charlie"/` 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