Homework 4

TCP

Assignment Overview

- HTTP over TCP
- TCP RTT estimation
- NS-3

Question 1

This assignment focuses on HTTP (not HTTPS) connections and data transfers based on TCP. For all exercises, assume the following parameters:

- Maximum Segment Size (MSS): 1600 Byte
- Initial Window Size (iW): 6

Question 1

In total, you need to draw 3 individual sequence diagrams including sequence and acknowledgment numbers as well as the amount of data transferred, where applicable.

As the assignments build on each other, use a continuous sequence and acknowledgment number space.

Question 1

In total, you need to draw 3 individual sequence diagrams including sequence and acknowledgment numbers as well as the amount of data transferred, where applicable.

As the assignments build on each other, use a continuous sequence and acknowledgment number space.

Question 1a)

Suppose you would like to access the Saarland University homepage. Draw a sequence diagram of the initial TCP connection setup.

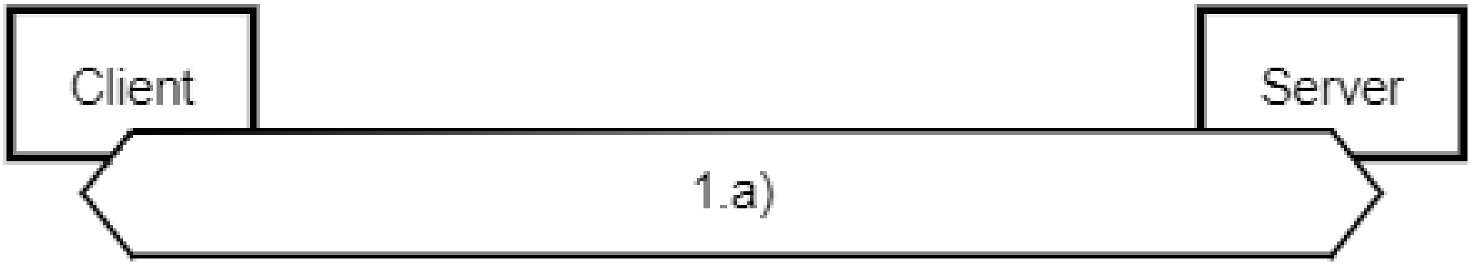
Assume your machine as the client chooses sequence number 4200 and the Saarland University web server chooses 5600.

Question 1a)

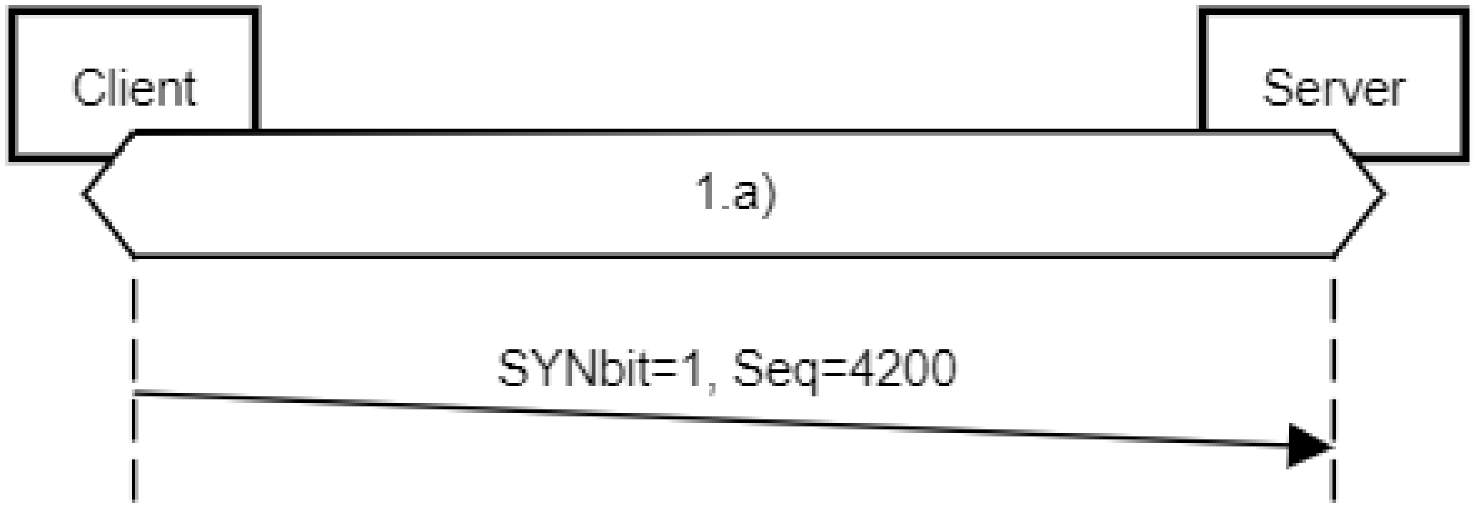
Suppose you would like to access the Saarland University homepage. Draw a sequence diagram of the initial TCP connection setup.

Assume your machine as the client chooses sequence number 4200 and the Saarland University web server chooses 5600.

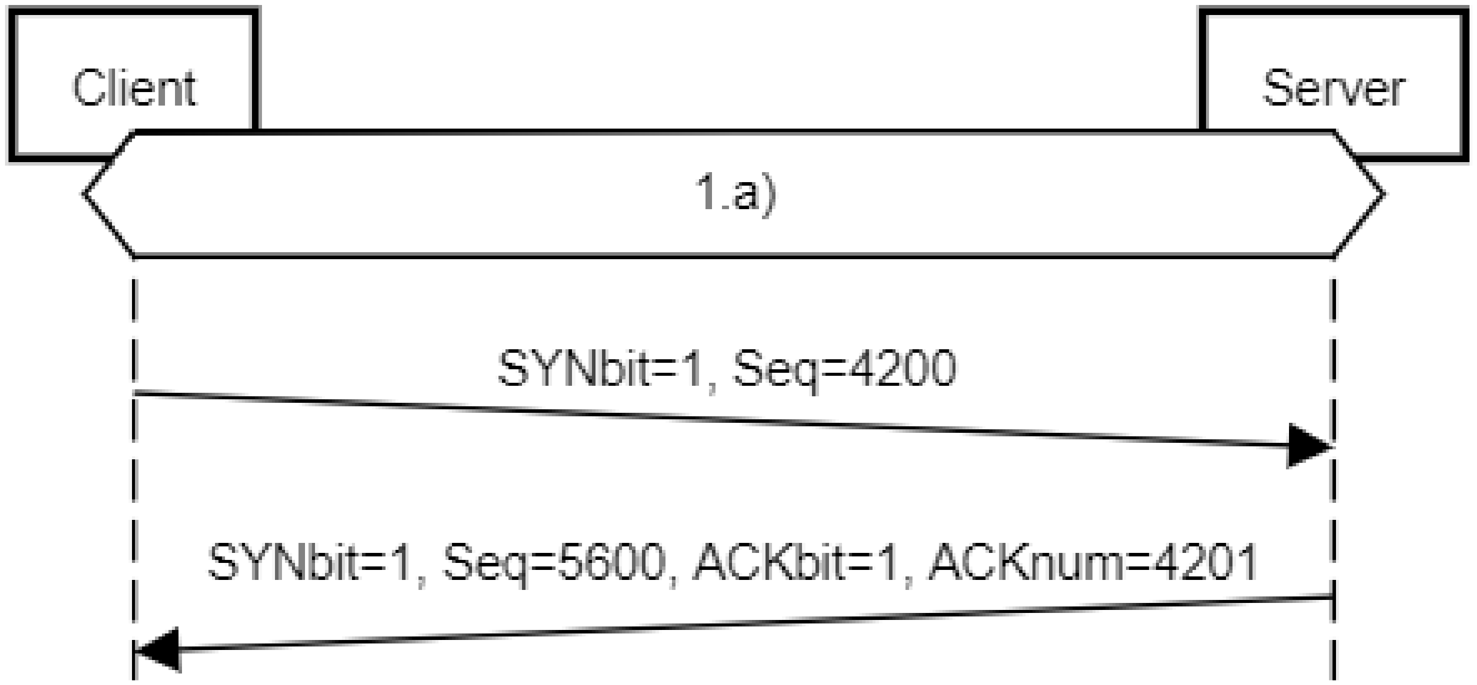
Question 1a)



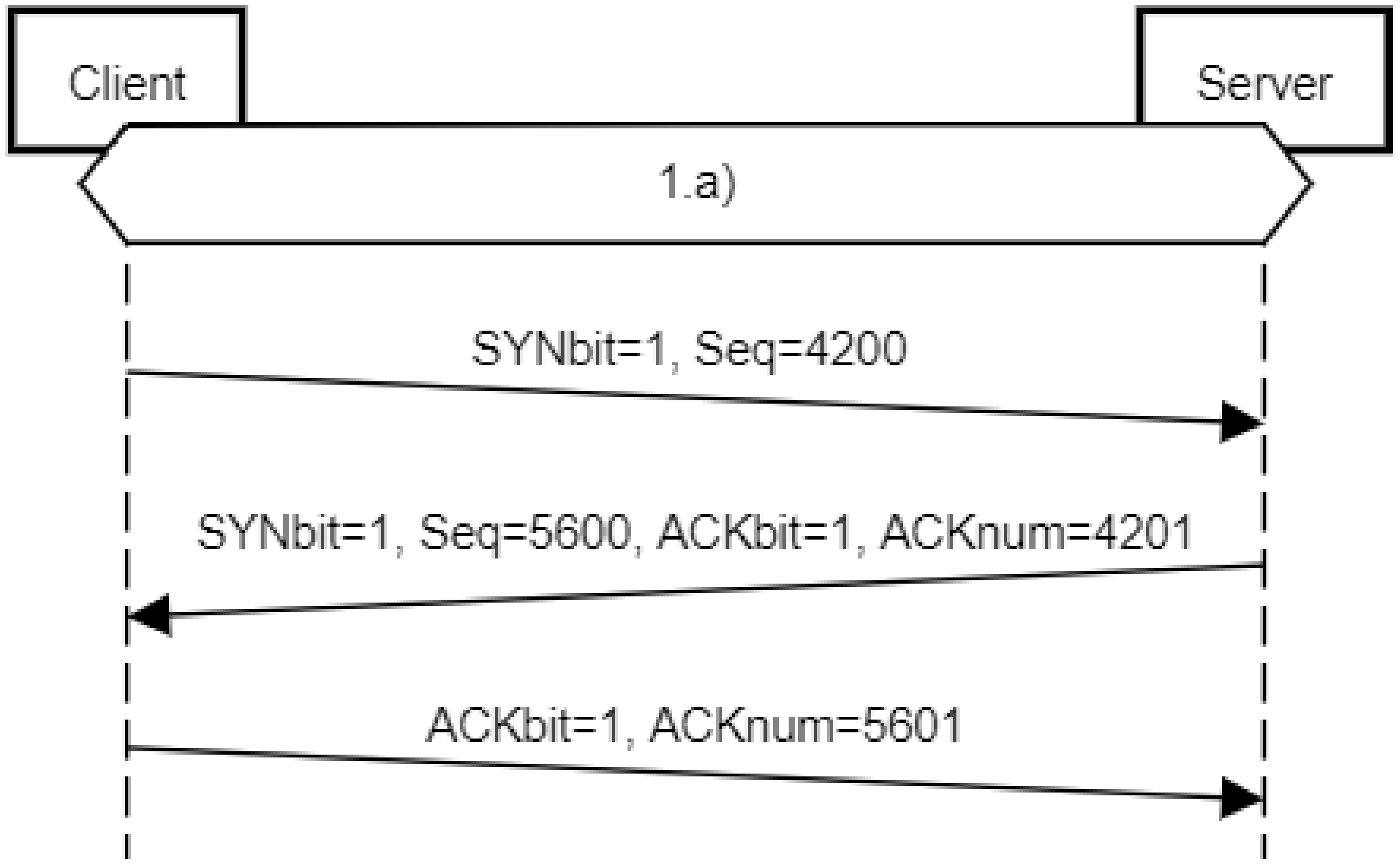
Question 1a)



Question 1a)


Client
Server
1.a)
SYNbit=1, Seq=4200
SYNbit=1, Seq=5600, ACKbit=1, ACKnum=4201

Question 1a)


Client
Server
1.a)
SYNbit=1, Seq=4200
SYNbit=1, Seq=5600, ACKbit=1, ACKnum=4201
ACKbit=1, ACKnum=5601

Question 1b)

Continuing on the diagram in a), draw another sequence diagram with the client sending a HTTP HEAD request to the web server, receiving the answer and closing the connection.

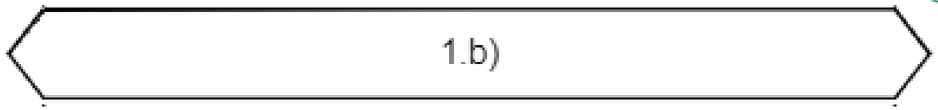
Assume the HTTP request (header) size of 690 bytes and an HTTP response size of 330 bytes.

Question 1b)

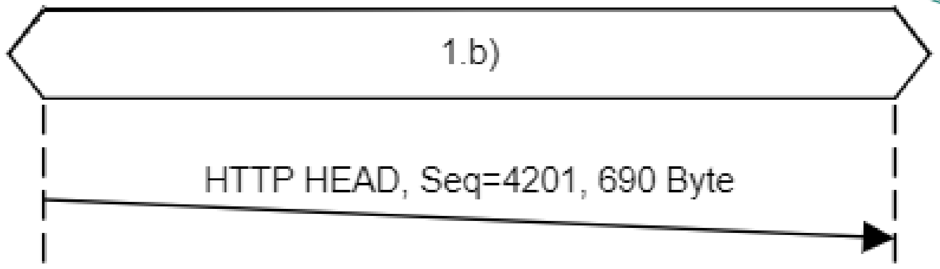
Continuing on the diagram in a), draw another sequence diagram with the client sending a HTTP HEAD request to the web server, receiving the answer and closing the connection.

Assume the HTTP request (header) size of 690 bytes and an HTTP response size of 330 bytes.

Question 1b)

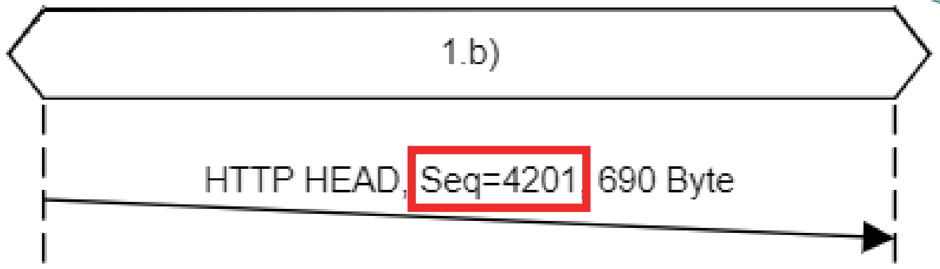


Question 1b)



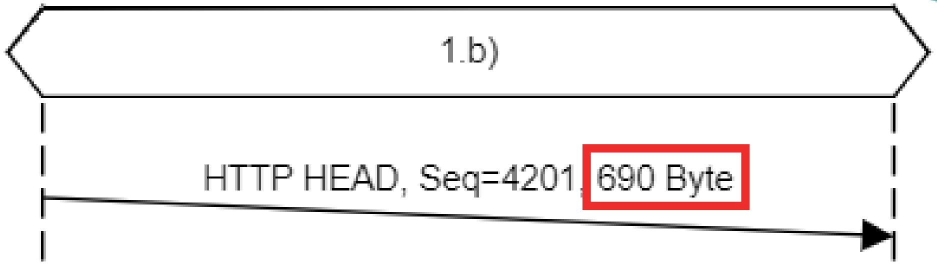
Question 1b)

- Seq from client

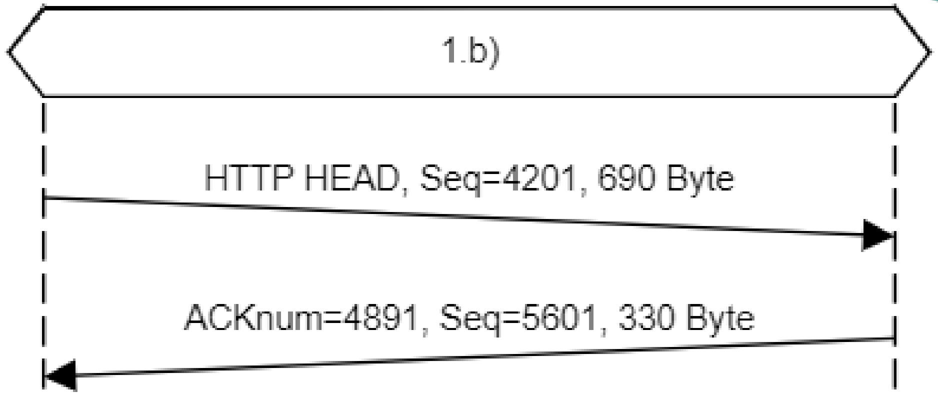


Question 1b)

- Seq from client
- HTTP request size

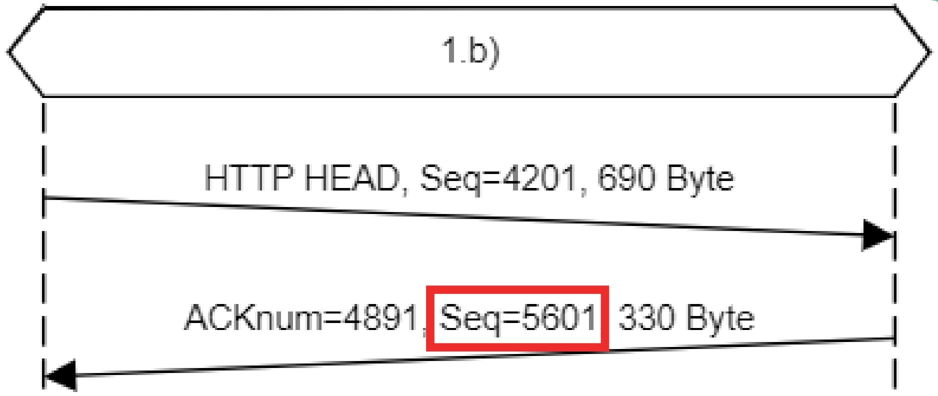


Question 1b)


1.b)
HTTP HEAD, Seq=4201, 690 Byte
ACKnum=4891, Seq=5601, 330 Byte

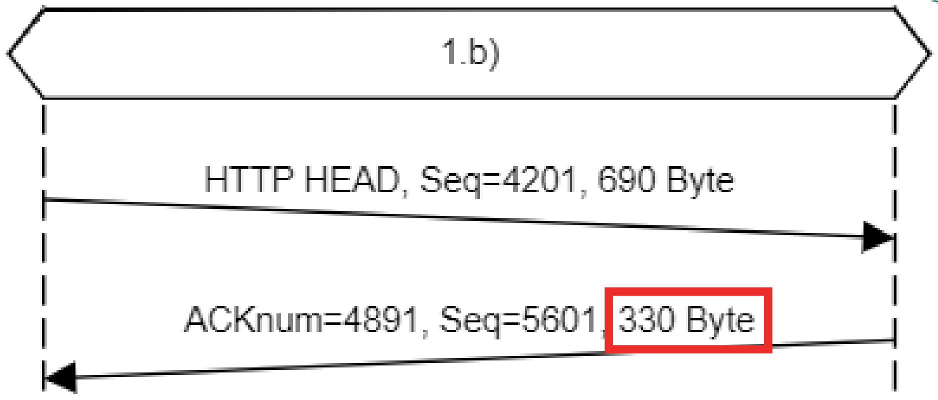
Question 1b)

- Seq from server

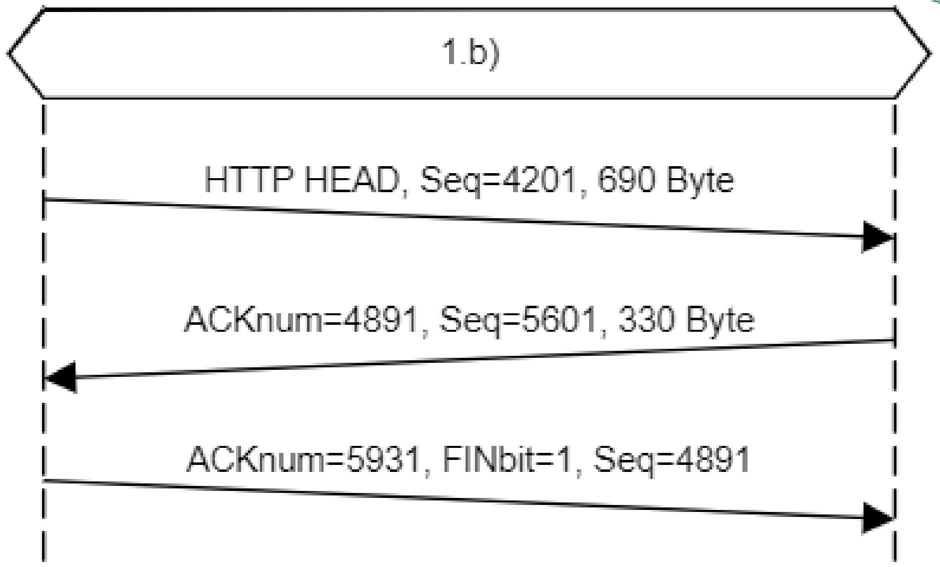

1.b)
HTTP HEAD, Seq=4201, 690 Byte
ACKnum=4891, Seq=5601, 330 Byte

Question 1b)

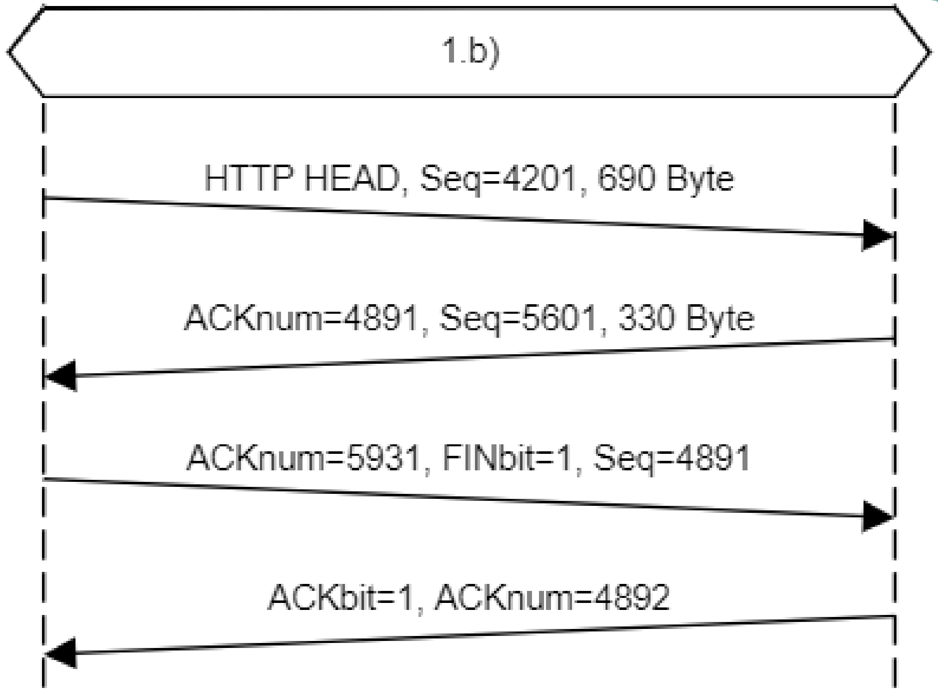
- Seq from server
- HTTP response size


1.b)
HTTP HEAD, Seq=4201, 690 Byte
ACKnum=4891, Seq=5601, 330 Byte

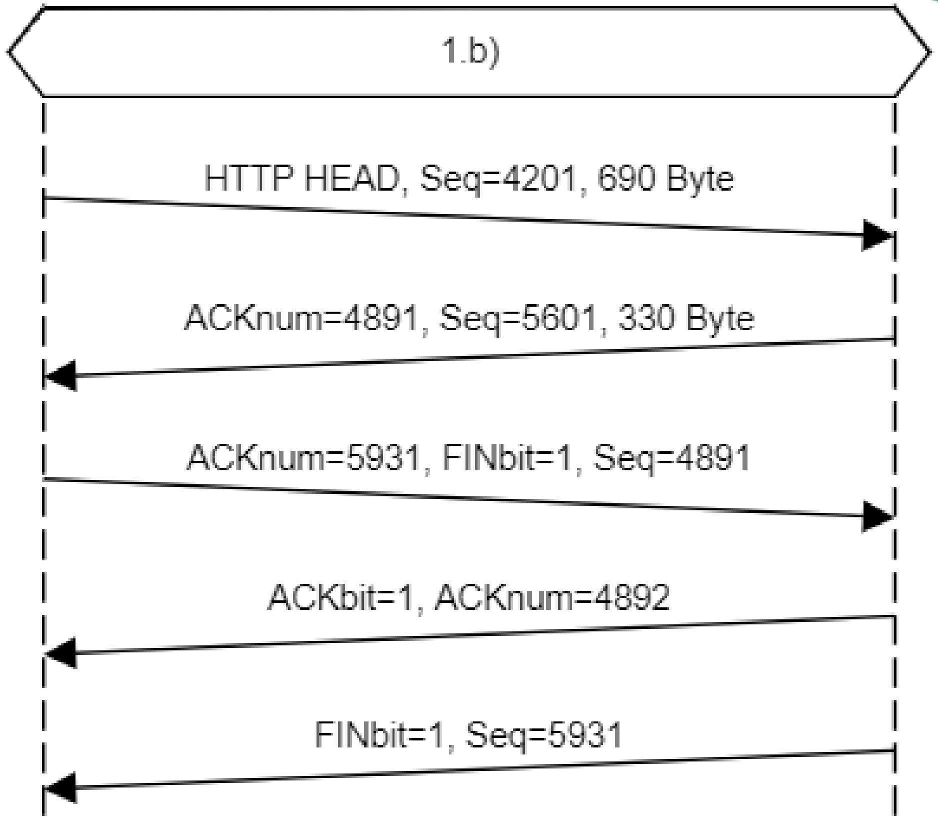
Question 1b)


1.b)
HTTP HEAD, Seq=4201, 690 Byte
ACKnum=4891, Seq=5601, 330 Byte
ACKnum=5931, FINbit=1, Seq=4891

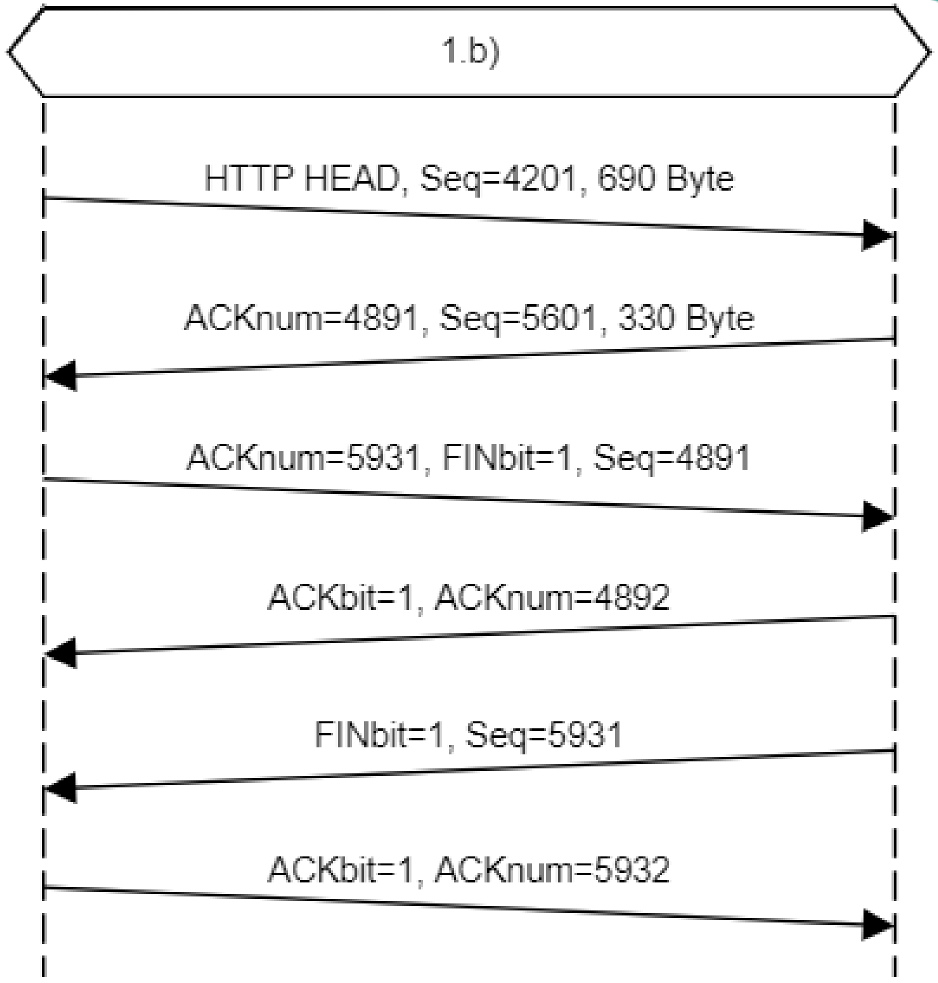
Question 1b)


1.b)
HTTP HEAD, Seq=4201, 690 Byte
ACKnum=4891, Seq=5601, 330 Byte
ACKnum=5931, FINbit=1, Seq=4891
ACKbit=1, ACKnum=4892

Question 1b)


1.b)
HTTP HEAD, Seq=4201, 690 Byte
ACKnum=4891, Seq=5601, 330 Byte
ACKnum=5931, FINbit=1, Seq=4891
ACKbit=1, ACKnum=4892
FINbit=1, Seq=5931

Question 1b)


1.b)
HTTP HEAD, Seq=4201, 690 Byte
ACKnum=4891, Seq=5601, 330 Byte
ACKnum=5931, FINbit=1, Seq=4891
ACKbit=1, ACKnum=4892
FINbit=1, Seq=5931
ACKbit=1, ACKnum=5932

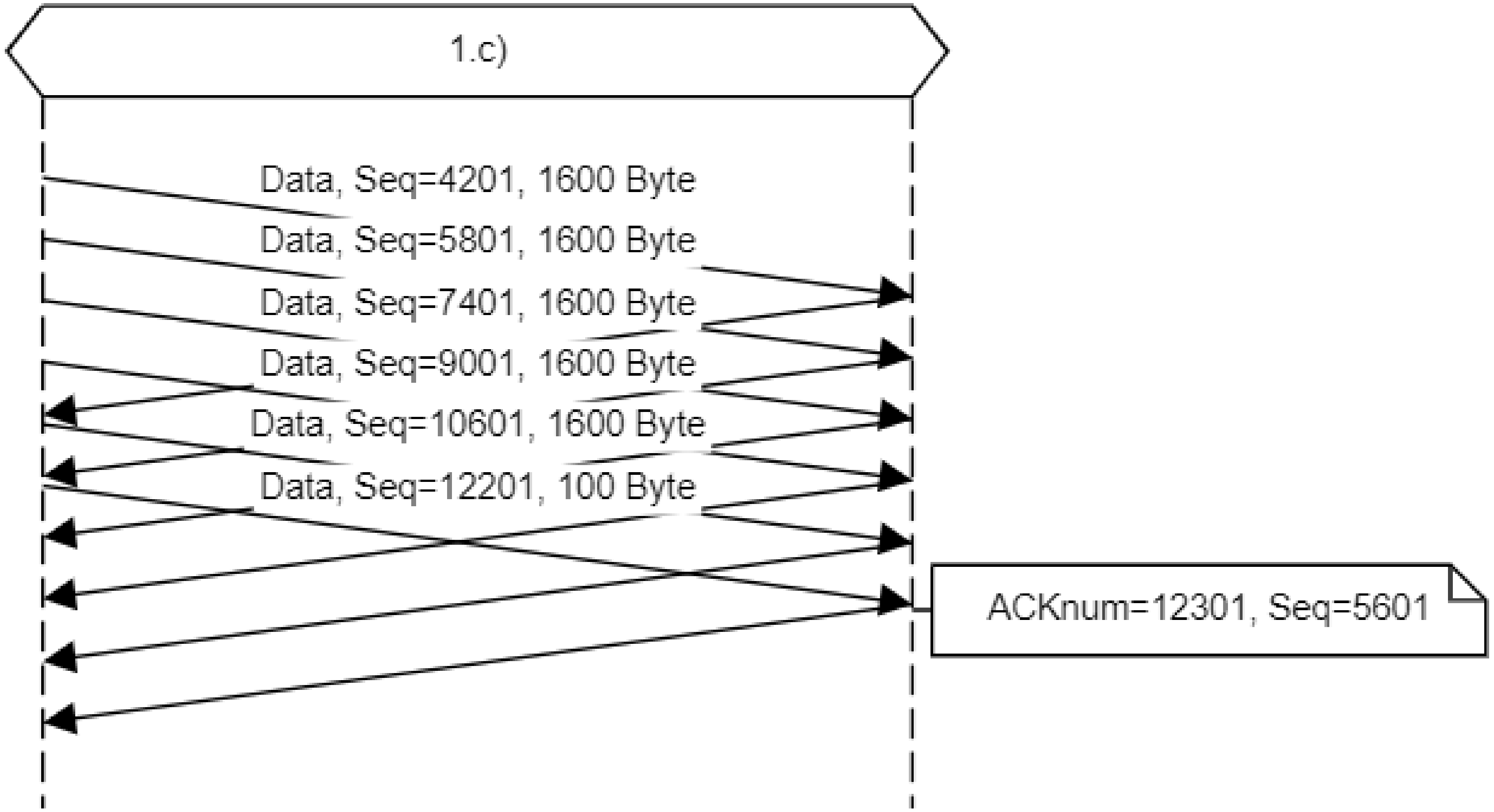
Question 1c)

Suppose you want to transmit 8100 bytes to the web server. Continuing with the sequence diagram you drew in a), draw another diagram to show the data transfer.

Question 1c)

Suppose you want to transmit 8100 bytes to the web server. Continuing with the sequence diagram you drew in a), draw another diagram to show the data transfer.

Question 1c)


1.c)
Data, Seq=4201, 1600 Byte
Data, Seq=5801, 1600 Byte
Data, Seq=7401, 1600 Byte
Data, Seq=9001, 1600 Byte
Data, Seq=10601, 1600 Byte
Data, Seq=12201, 100 Byte
ACKnum=12301, Seq=5601

Question 1d)

Suppose the client wants to send 7 segments to the server but the 3rd segment gets lost. Briefly describe how the server will react. What does the client do to make sure the 3rd segment is received by the server?

Answer:

Question 1d)

Suppose the client wants to send 7 segments to the server but the 3rd segment gets lost. Briefly describe how the server will react. What does the client do to make sure the 3rd segment is received by the server?

Answer:

Question 1d)

Suppose the client wants to send 7 segments to the server but the 3rd segment gets lost. Briefly describe how the server will react. What does the client do to make sure the 3rd segment is received by the server?

Answer:

The server will keep sending ACKs with the sequence number expected for the lost packet.

Question 1d)

Suppose the client wants to send 7 segments to the server but the 3rd segment gets lost. Briefly describe how the server will react. What does the client do to make sure the 3rd segment is received by the server?

Answer:

Question 1d)

Suppose the client wants to send 7 segments to the server but the 3rd segment gets lost. Briefly describe how the server will react. What does the client do to make sure the 3rd segment is received by the server?

Answer:

After triple duplicate ACKs, the client will retransmit the lost segment (fast retransmit).

Question 1e)

Older TCP versions used a Stop-And-Wait mechanism rather than the Go-Back-N mechanism. Suppose you would like to transmit an additional 2.5MB to the web server. Assume that no packets are lost, no nodal processing delay occurs, no queueing delay occurs, and the RTT is 17 ms. How long would the transmission take using each of the mechanisms (in seconds)? Are there any benefits in using a Stop-And-Wait mechanism? For this question, assume the TCP connection has already been established.

Question 1e)

Older TCP versions used a Stop-And-Wait mechanism rather than the Go-Back-N mechanism. Suppose you would like to transmit an additional 2.5MB to the web server. Assume that no packets are lost, no nodal processing delay occurs, no queueing delay occurs, and the RTT is 17 ms. How long would the transmission take using each of the mechanisms (in seconds)? Are there any benefits in using a Stop-And-Wait mechanism? For this question, assume the TCP connection has already been established.

Question 1e)

Stop-And-Wait:

Assuming the TCP connection is already established:

$$\frac{2.5\,MB}{1600\,B} \approx 1563\ segments$$

$$1563 \times 0.017 = 26.6\ seconds$$

Question 1e)

Go-Back-N:

Assuming the TCP connection is already established

$1^{st} wnd \mid 6 \mid 9600 \text{ Bytes}$

Question 1e)

Go-Back-N:

Assuming the TCP connection is already established

$1^{st} wnd$	6	9600 Bytes
$2^{nd} wnd$	12	19200 Bytes

Question 1e)

Go-Back-N:

Assuming the TCP connection is already established

$1^{st}wnd$	6	9600 Bytes
$2^{nd}wnd$	12	19200 Bytes
$3^{rd}wnd$	24	38400 Bytes

Question 1e)

Go-Back-N:

Assuming the TCP connection is already established

$1^{st} wnd$	6	9600 Bytes
$2^{nd} wnd$	12	19200 Bytes
$3^{rd} wnd$	24	38400 Bytes
$4^{th} wnd$	48	76800 Bytes
$5^{th} wnd$	96	153600 Bytes
$6^{th} wnd$	192	307200 Bytes
$7^{th} wnd$	384	614400 Bytes
$8^{th} wnd$	768	1228800 Bytes
$9^{th} wnd$	_	52000 Bytes

Question 1e)

Go-Back-N:

Assuming the TCP connection is already established

$1^{st} wnd$	6	9600 Bytes
$2^{nd} wnd$	12	19200 Bytes
$3^{rd} wnd$	24	38400 Bytes
$4^{th} wnd$	48	76800 Bytes
$5^{th} wnd$	96	153600 Bytes
$6^{th} wnd$	192	307200 Bytes
$7^{th} wnd$	384	614400 Bytes
$8^{th} wnd$	768	1228800 Bytes
$9^{th} wnd$	–	52000 Bytes

$$\times 0.017 = 0.15 seconds$$

Question 1e)

Older TCP versions used a Stop-And-Wait mechanism rather than the Go-Back-N mechanism. Suppose you would like to transmit an additional 2.5MB to the web server. Assume that no packets are lost, no nodal processing delay occurs, no queueing delay occurs, and the RTT is 17 ms. How long would the transmission take using each of the mechanisms (in seconds)? Are there any benefits in using a Stop-And-Wait mechanism? For this question, assume the TCP connection has already been established.

Question 1e)

Older TCP versions used a Stop-And-Wait mechanism rather than the Go-Back-N mechanism. Suppose you would like to transmit an additional 2.5MB to the web server. Assume that no packets are lost, no nodal processing delay occurs, no queueing delay occurs, and the RTT is 17 ms. How long would the transmission take using each of the mechanisms (in seconds)? Are there any benefits in using a Stop-And-Wait mechanism? For this question, assume the TCP connection has already been established.

Question 1e)

- Simple protocol
- Low chance of congesting the link
- No TCP re-ordering at the client

Question 1f)

Assume that the client has indicated the support of the TCP Selective Acknowledgements (SACK) option during the connection establishment and the server is capable of sending SACK options. Suppose a segment gets corrupted during transmission. How does the receiver react? What are advantages and disadvantages of this behavior?

* The Request For Comments by Mathis et al. might provide hints to solve this question.

Question 1f)

Assume that the client has indicated the support of the TCP Selective Acknowledgements (SACK) option during the connection establishment and the server is capable of sending SACK options. Suppose a segment gets corrupted during transmission. How does the receiver react? What are advantages and disadvantages of this behavior?

* The Request For Comments by Mathis et al. might provide hints to solve this question.

Question 1f)

When using SACK, the TCP receiver tells the sender which packets arrived successfully. The sender then retransmits only the segments that were lost.

Question 1f)

When using SACK, the TCP receiver tells the sender which packets arrived successfully. The sender then retransmits only the segments that were lost.

For cumulative ACKs, the sender retransmits all currently unacknowledged packets at a certain point in time

Question 1f)

Assume that the client has indicated the support of the TCP Selective Acknowledgements (SACK) option during the connection establishment and the server is capable of sending SACK options. Suppose a segment gets corrupted during transmission. How does the receiver react? What are advantages and disadvantages of this behavior?

* The Request For Comments by Mathis et al. might provide hints to solve this question.

Question 1f)

+ Fewer retransmissions
- Increased TCP header size

Question 2

Consider the TCP procedure for estimating RTT.

$SampleRTT_1$ = Most recent sample RTT

$SampleRTT_2$ = Next most recent sample RTT

…

Formula:

$$EstimatedRTT_{new} = (1-\alpha)\cdot EstimatedRTT_{old} + \alpha \cdot SampleRTT_{recent}$$

Question 2a)

Five measured SampleRTT values are 140ms, 120ms, 260ms, 220ms, and 410ms. Compute the $EstimatedRTT$ for each of these $SampleRTT$ values, using a value of $\alpha = 0.135$ and assuming the initial $EstimatedRTT$ was 130ms. Round off your answers to 3 decimal places.

Question 2a)

Five measured SampleRTT values are 140ms, 120ms, 260ms, 220ms, and 410ms. Compute the $EstimatedRTT$ for each of these $SampleRTT$ values, using a value of $\alpha = 0.135$ and assuming the initial $EstimatedRTT$ was 130ms. Round off your answers to 3 decimal places.

Question 2a)

$$140\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 130 + 0.135 \cdot 140 = 131.35 \text{ ms}$$

Question 2a)

$$140\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 130 + 0.135 \cdot 140 = 131.35 \text{ ms}$$

$$120\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 131.35 + 0.135 \cdot 120 = 129.818 \text{ ms}$$

Question 2a)

$$140\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 130 + 0.135 \cdot 140 = \boxed{131.35}\text{ ms}$$

$$120\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot \boxed{131.35} + 0.135 \cdot 120 = 129.818\text{ ms}$$

Question 2a)

$$140\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 130 + 0.135 \cdot 140 = 131.35 \text{ ms}$$

$$120\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 131.35 + 0.135 \cdot 120 = 129.818 \text{ ms}$$

$$260\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 129.818 + 0.135 \cdot 260 = 147.393 \text{ ms}$$

Question 2a)

$$140\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 130 + 0.135 \cdot 140 = 131.35 \text{ ms}$$

$$120\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 131.35 + 0.135 \cdot 120 = 129.818 \text{ ms}$$

$$260\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 129.818 + 0.135 \cdot 260 = 147.393 \text{ ms}$$

$$220\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 147.393 + 0.135 \cdot 220 = 157.195 \text{ ms}$$

Question 2a)

$$140\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 130 + 0.135 \cdot 140 = 131.35 \text{ ms}$$

$$120\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 131.35 + 0.135 \cdot 120 = 129.818 \text{ ms}$$

$$260\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 129.818 + 0.135 \cdot 260 = 147.393 \text{ ms}$$

$$220\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 147.393 + 0.135 \cdot 220 = 157.195 \text{ ms}$$

$$410\text{ms: } \texttt{EstimatedRTT} = 0.865 \cdot 157.195 + 0.135 \cdot 410 = 191.324 \text{ ms}$$

Question 2b)

Given n sample RTTs, the formula can be generalized to:

$$EstimatedRTT_{new} = \alpha \cdot \sum_{i=1}^{n-1} (1-\alpha)^{i-1} \cdot SampleRTT_i + (1-\alpha)^{n-1} \cdot SampleRTT_n$$

For formula (2), let n approach infinity. Comment on why this averaging procedure is called an exponential moving average and why no simple average is used.

Question 2b)

Given n sample RTTs, the formula can be generalized to:

$$EstimatedRTT_{new} = \alpha \cdot \sum_{i=1}^{n-1} (1-\alpha)^{i-1} \cdot SampleRTT_i + (1-\alpha)^{n-1} \cdot SampleRTT_n$$

For formula (2), let n approach infinity. Comment on why this averaging procedure is called an exponential moving average and why no simple average is used.

Question 2b)

The averaging procedure is called an exponential moving average because it uses a sliding window in which the weight of older SampleRTTs decreases exponentially fast. An exponential moving average, rather than a simple average, is used because RTTs might change over time and more recent SampleRTTs provide more information on the current network conditions than older SampleRTTs.

Question 2c)

Why do you think TCP avoids measuring the SampleRTT for retransmitted segments? Explain!

Question 2c)

Why do you think TCP avoids measuring the SampleRTT for retransmitted segments? Explain!

Question 2c)

Why do you think TCP avoids measuring the SampleRTT for retransmitted segments? Explain!

Answer:

Retransmissions may lead to "retransmission ambiguity", this may lead to significantly larger RTTs that distort the moving average.

Question 3

In this question we require you to work with the Network-Simulator-3 (NS-3) via a VM Image that we provide.

NS-3 is used primarily in networking research and development, and allows you to simulate complex network and traffic configurations.

→ We will not go into setup here. ←

Question 3a)

Using the pcap files generated by the provided NS-3 script, pick a data visualization tool of your choice (No Wireshark plots!) and provide us with a figure showing:

- The elapsed time since the start of the flows on the x-axis
- The sending node’s throughput in Megabits per second (Mbps) on the y-axis for each flow
- A legend that describes which flow is represented by which line

Question 3a)

Using the pcap files generated by the provided NS-3 script, pick a data visualization tool of your choice (No Wireshark plots!) and provide us with a figure showing:

- The elapsed time since the start of the flows on the x-axis
- The sending node's throughput in Megabits per second (Mbps) on the y-axis for each flow
- A legend that describes which flow is represented by which line

Question 3a)

The bytes from the sender for the TCP and UDP flows can be found by using the following tshark commands and redirected to a different file for further processing:

UDP data: tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e ip.src ip.src==<src IP> and ip.proto==17 |awk '{print $1, $2}'

TCP data: tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e ip.src ip.src==<src IP> and ip.proto==6 |awk '{print $1, $2}'

Question 3a)

The bytes from the sender for the TCP and UDP flows can be found by using the following tshark commands and redirected to a different file for further processing:

UDP data: tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e ip.src ip.src==**<src IP>** and ip.proto==17 |awk ‘{print $1, $2}’

TCP data: tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e ip.src ip.src==**<src IP>** and ip.proto==6 |awk ‘{print $1, $2}’

Question 3a)

The bytes from the sender for the TCP and UDP flows can be found by using the following tshark commands and redirected to a different file for further processing:

UDP data: tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e ip.src ip.src==10.1.1.1 and ip.proto==17 |awk '{print $1, $2}' > udp.plt

TCP data: tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e ip.src ip.src==10.1.1.1 and ip.proto==6 |awk '{print $1, $2}' > tcp.plt

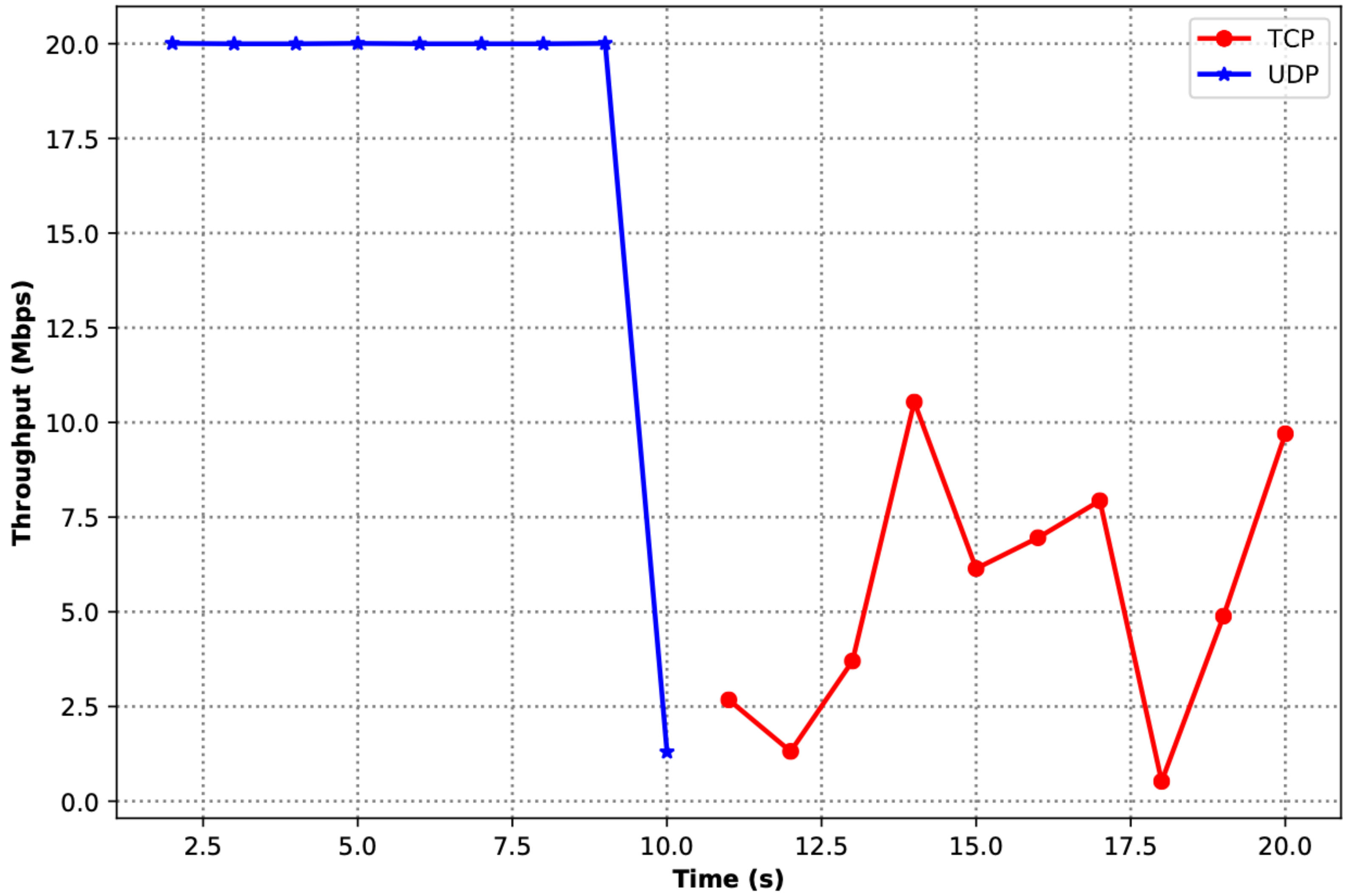
Question 3a)

The bytes from the sender for the TCP and UDP flows can be found by using the following tshark commands and redirected to a different file for further processing:

UDP data:
```
tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e
ip.src ip.src==10.1.1.1 and ip.proto==17 |awk ‘{print $1, $2}’ > udp.plt
```

TCP data:
```
tshark -r <pcap name> -T fields -e frame.time epoch -e frame.len -e ip.proto -e ip.src
ip.src==10.1.1.1 and ip.proto==6 |awk ‘{print $1, $2}’ > tcp.plt
```

Question 3a)


TCP
UDP
Throughput (Mbps)
Time (s)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Question 3b)

Based on the resulting figure, what differences do you notice between the TCP and UDP flows? Discuss and justify your observations.

Question 3b)

Based on the resulting figure, what differences do you notice between the TCP and UDP flows? Discuss and justify your observations.

Question 3b)

Based on the resulting figure, what differences do you notice between the TCP and UDP flows? Discuss and justify your observations.

- UDP sends at a steady rate, why?

Question 3b)

Based on the resulting figure, what differences do you notice between the TCP and UDP flows? Discuss and justify your observations.

- UDP sends at a steady rate, why?

  → No flow / congestion control.

Question 3b)

Based on the resulting figure, what differences do you notice between the TCP and UDP flows? Discuss and justify your observations.

- UDP sends at a steady rate, why?

  → No flow / congestion control.

- TCP sends at a varying rate, why?

Question 3b)

Based on the resulting figure, what differences do you notice between the TCP and UDP flows? Discuss and justify your observations.

- UDP sends at a steady rate, why?

  ➞ No flow / congestion control.

- TCP sends at a varying rate, why?

  ➞ Recognises and reacts to loss.

Feedback?