Data Networks WS 18/19

INTERNET ARCHITECTURE:

Assignment 4

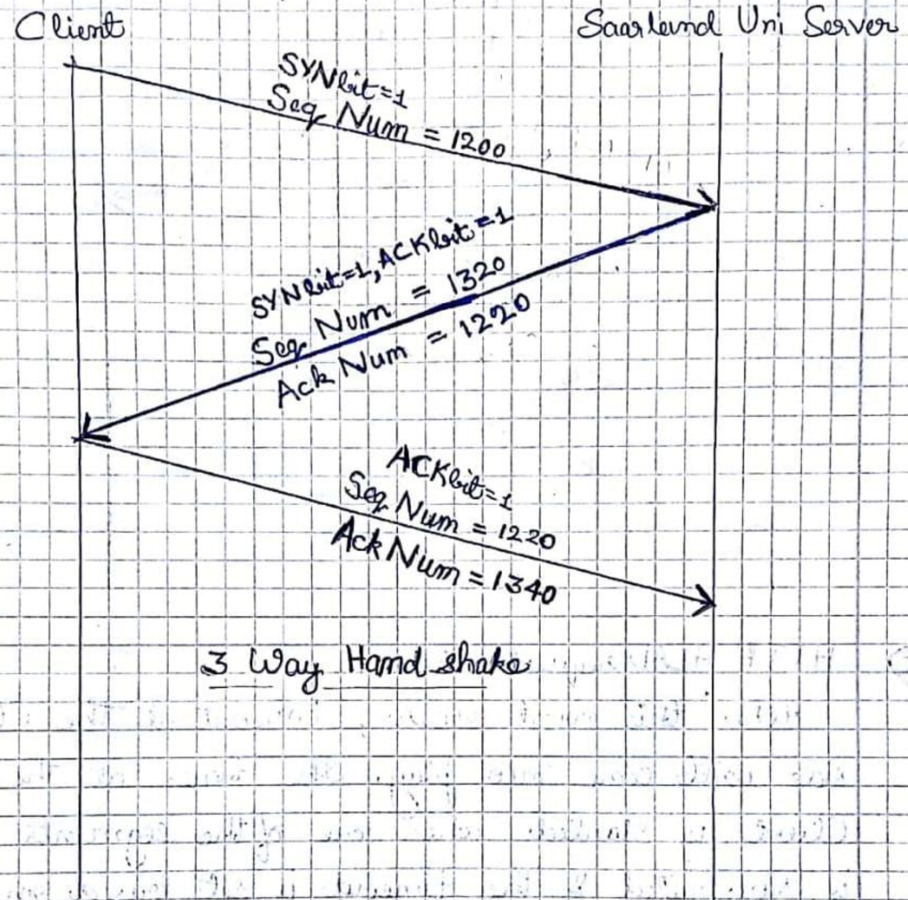
Chirag Bhuvaneshwara - 2571703

Florena Raja – 2566418

Question 1 :

MSS: 1460 Bytes ; iW = 10 ; HS = 20 Byte

a) TCP connection setup :

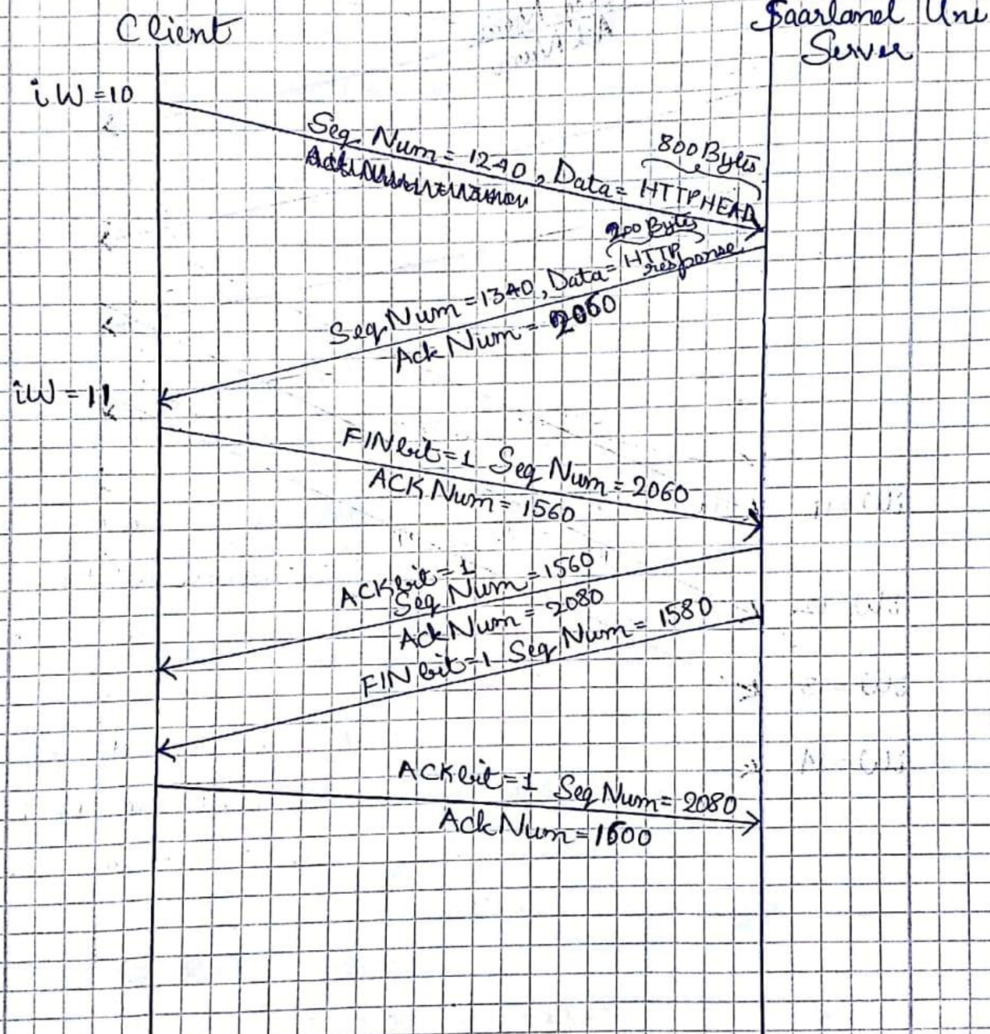


3 Way Hand shake

~~with complex background~~

HTTP Head:

b)



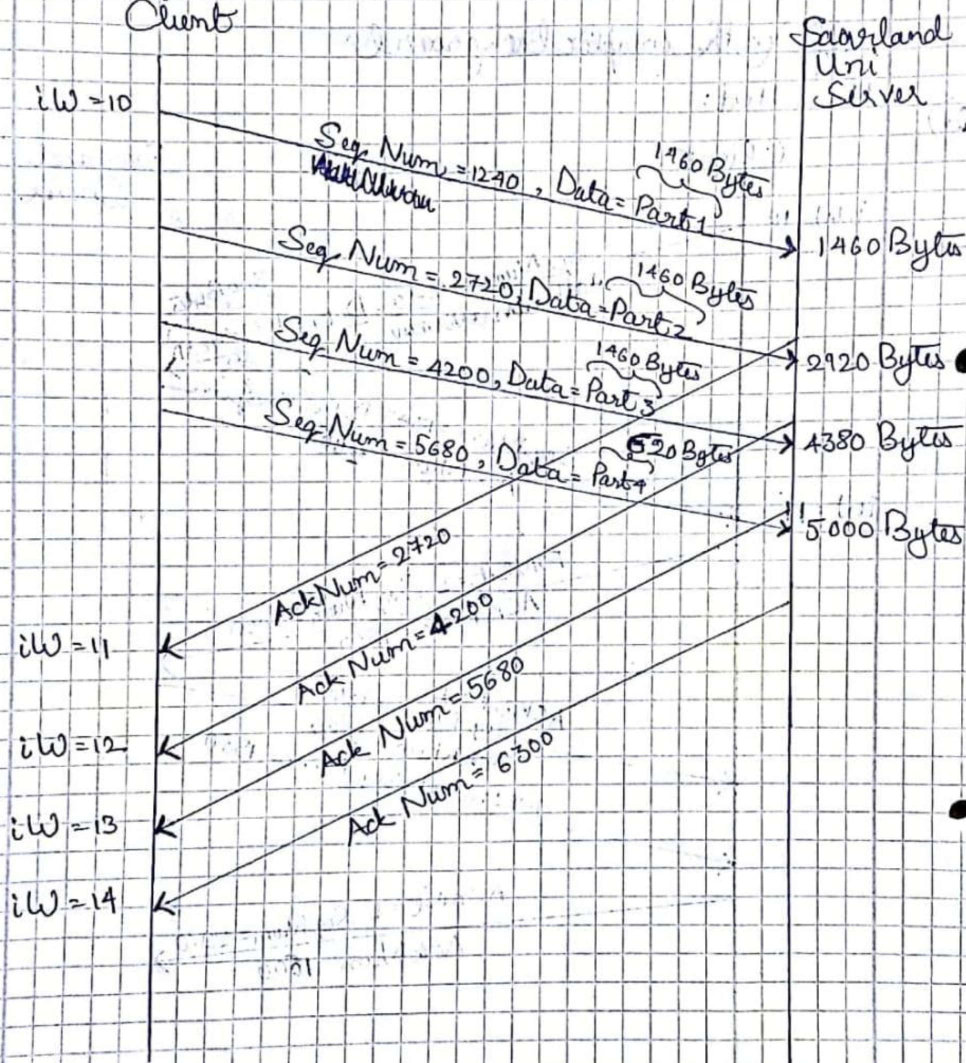
c) HTTP HEAD request lost:

In this case, the server takes no action as it has no idea that it was contacted. For the client, when the timer runs out, the request is retransmitted to the Uni Server. The timer runs out because the ACK for HTTP Head was not received.

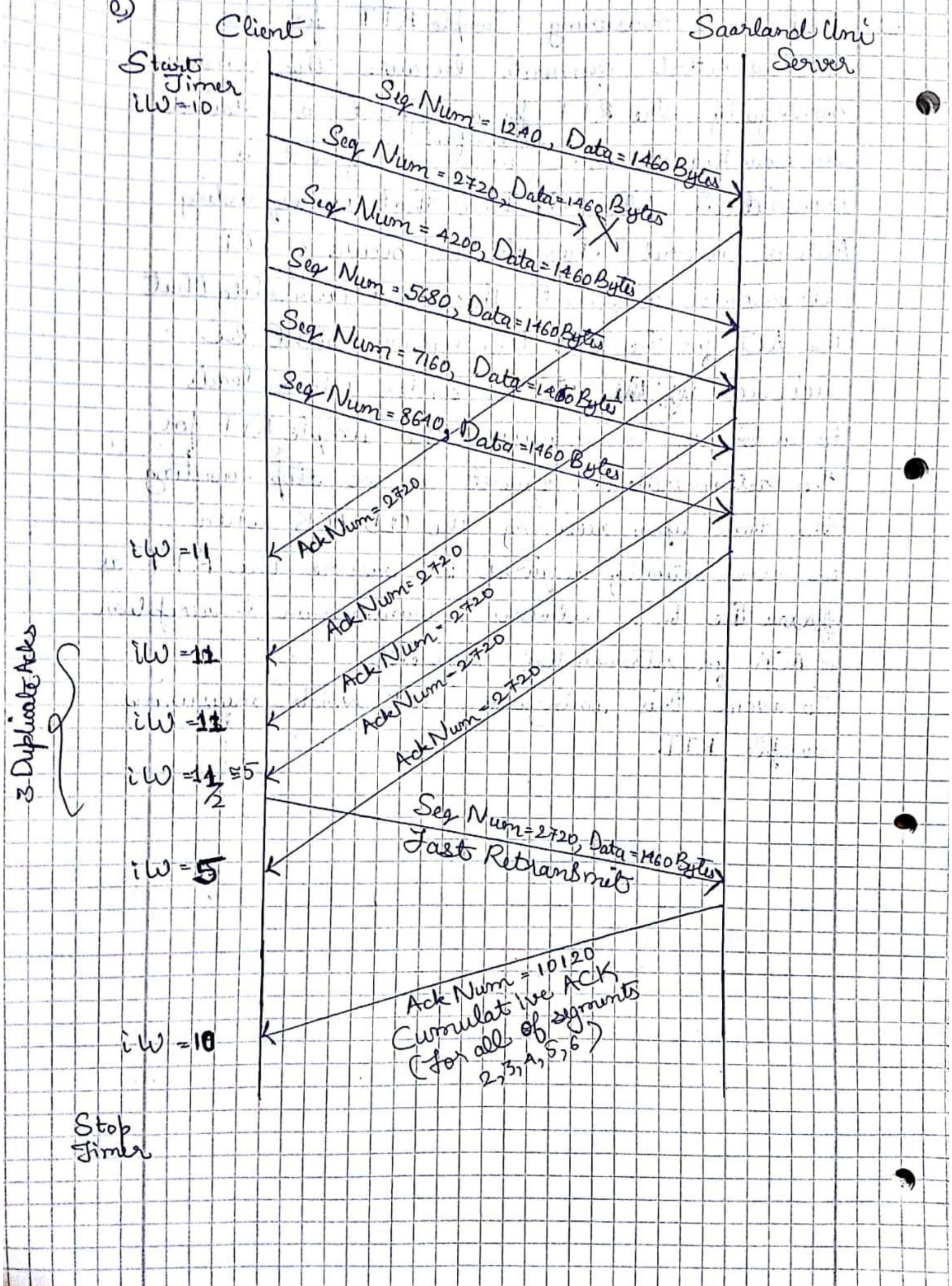
d) TCP Segment = $(20 + 1460)$ Bytes = 1480 Bytes

File size = 5000 Bytes = $(1460 + 1460 + 1460 + 620)$ Bytes

= Part1 + Part2 + Part3 + Part4



e)


Client
Saarland Uni Server
Start Timer
iW=10
Seq Num = 1240, Data = 1460 Bytes
Seq Num = 2720, Data = 1460 Bytes
Seq Num = 4200, Data = 1460 Bytes
Seq Num = 5680, Data = 1460 Bytes
Seq Num = 7160, Data = 1460 Bytes
Seq Num = 8640, Data = 1460 Bytes
Ack Num = 2720
iW = 11
Ack Num = 2720
iW = 11
Ack Num = 2720
iW = 11
Ack Num = 2720
iW = 11/2 ≅ 5
Ack Num = 2720
3 Duplicate Acks
Seq Num = 2720, Data = 1460 Bytes
Fast Retransmit
iW = 5
Ack Num = 10120
Cumulative ACK
(for all of segments 2, 3, 4, 5, 6)
iW = 10
Stop Timer

- Server Reaction:
When server receives out of order segment, it sends Ack for last in order segment successfully recieved. Then server repeats this action for every subsequent segment till the ~~original~~ lost segment is received.

- Client Action: 2 Possible actions:

1) Timer still running:
Within timeout, if client recieves 3 duplicate ACKs, it realises that ~~the~~ following segment is lost & retransmits it before timeout occurs as in diagram.

2) Timer ran out:
After timeout, client ~~re~~transmits the segment that follows the last segment that was ACK'd before the timeout.

The client also understands cumulative ACK meaning that once lost segment is retransmitted and if server stored previously delivered out of order segments then the server sends a cumulative ACK which acknowledges receipt of all 6 segments.

Question 2: TCP's RTT Estimation:

$EstRTT_{new} = (1-\alpha) . EstRTT_{old} + \alpha . Sample\ RTT$

Measured Sample RTTs : 130 ms , 150 ms , 320 ms , 220 ms , 300 ms

1 2 3 4 5

Most Recent ← Oldest

- $Sample\ RTT_5 = 300ms$

$Est\ RTT_{new} = ((1-0.125) . Est\ RTT_{initial}) + (0.125 \times 300)$

$Est\ RTT_{new} = (0.875 \times 115) + (0.125 \times 300)$

$Est\ RTT_{new} = 138.125\ ms$

- $Sample\ RTT_4 = 220\ ms$

$Est\ RTT_{new} = (0.875 \times 138.125) + (0.125 \times 220)$

$Est\ RTT_{new} = 148.3594\ ms$

- $Sample\ RTT_3 = 320\ ms$

$Est\ RTT_{new} = ((1-0.125) \times 148.3594) + (0.125 \times 320)$

$Est\ RTT_{new} = 169.8144\ ms$

- $Sample\ RTT_2 = 150\ ms$

$Est\ RTT_{new} = (0.875 \times 169.8144) + (0.125 \times 150)$

$Est\ RTT_{new} = 167.3377\ ms$

- $Sample\ RTT_1 = 130\ ms$

$Est\ RTT_{new} = (0.875 \times 167.3377) + (0.125 \times 130)$

$Est\ RTT_{new} = 162.6705\ ms$

(b) When n approaches infinity in (2), the SampleRTT$_i$ at small values of i such as 1, 2, ... are given a bigger weight in the summation term since $\alpha < 1$.
∴ SampleRTT$_i$ for small values of i are given more importance & for values of i close to $n-1$, we have very small weights (close to 0) which means SampleRTT$_i$ for i close to $n-1$ are given really less importance.
Also, this weighting decreases exponentially with increase in i from 1 to $n-1$.
For SampleRTT$_n$ we have a very small weight & its effect is not considered to a great extent because of the small weight.
In essence, we call (2) exponential moving average since the most recent sample RTT is given more importance & the previous sampleRTTs are given exponentially less importance as they do not accurately reflect the current state of the network. And this averaging scheme keeps moving as we obtain new SampleRTTs which are again given more importance.

c) TCP avoids measuring Sample RTT for retransmitted segments because this removes ambiguity. The RTT for a segment is obtained by counting the time starting from ~~when~~ a segment is transmitted till the time that corresponding ACK is received. When timeout occurs & TCP retransmits a segment, there is a possibility that the ACK for the 1st transmission might be recieved ~~by the~~ at the sender which leads to ambiguity in measuring the sample RTT for the retransmitted segment as we stop counting the time upon receiving the OLD ACK when we were actually required to measure the RTT as ~~from~~ the time between retransmission & reception of ACK for retransmitted segment.

To avoid this ambiguity, TCP avoids measuring sample RTT.