Name: Alvi Akbar
Student No: 11118887
NSID alv273

Assignment #03

Part B Solutions :

① Longed average rate = $R = 200$ Mbps $= 200 \times 10^6$ bits/second.

$= 200 \times 10^6 \frac{bits}{second} \times \frac{1\ byte}{8\ bits}$

$= 2.5 \times 10^7$ B/s

duration of max burst

$S = 1\ ms = 10^{-3}\ s$

Link rate $= M = 2$ Gbps $= 2 \times 10^9$ b/s

$= 2.5 \times 10^6$ B/s

Max No of tokens found in bucket $= B = M \times S$ bytes in Burst

$B = 2.5 \times 10^6$ B/s $\times 10^{-3}$ s

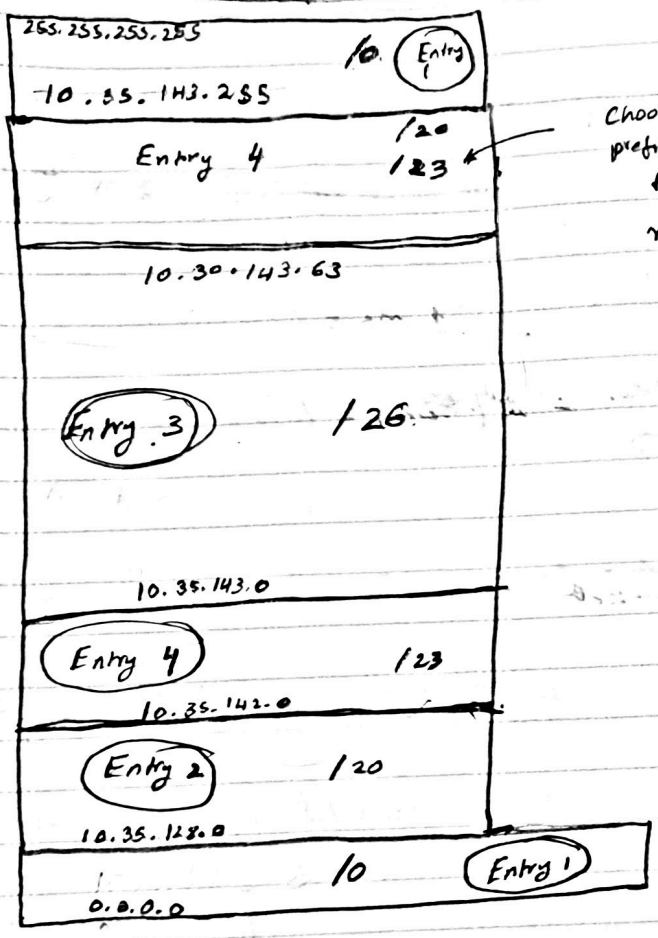
$B = 2.5 \times 10^5$ Bytes in Burst.

② - Decoding the given IP Addresses Range.

- Then we will use the **Longest Matching Prefix Algorithm**

Solution → Next Page

Finding IP Range using Longest prefix algorithm



For 10.35.128.0/20 & 32-20 = 12 bits which will be added to low range i.e 10.35.128.0

10000000.0000 0000
+ 1111.1111 1111
10001111 .1111 1111 i.e 10.35.143.255

10.35.142.0/23 32-23 = 9 bits

1000 1110
+ 1
143

∴ Higher Range : 10.35.143.255

Similarly for 10.35.143.0/26

Range is 10.35.143.0 to 10.35.143.63

Based on the diagram of IP ranges in previous page

(a) 10.35.143.128 forwarded to Entry 4 via destination link layer
Address of next hop router is 10.35.128.1
interface: A

(b) 10.35.140.28 forwarded to Entry 2 via IP datagram's destination.
interface: A

(c) 10.35.144.6 forwarded to Entry 1 via dest link layer
Address of next hop router is 10.0.81.28
interface: B

(d) 10.11.143.6 forwarded to Entry 1 via dest link layer
Address of next hop router is 10.0.81.28
interface: B.

(3) SRTT = 150 ms old = 0.875.

RTT = 75 ms

new sample = 0.125

until SRTT less than 100 ms

$$SRTT_n = \alpha\, SRTT + (1-\alpha)\, RTT$$

$\alpha = 0.875$

$$\therefore \quad f(n) = (0.875)^n (150) + \sum_{i=1}^{n} \left[(0.125)(0.875)^{(n-i)} (75) \right]$$

Question 3

$$SRTT_n = \alpha\,(SRTT) + (1-\alpha)(RTT)$$

$\alpha = 0.875$
$SRTT = 150\,ms$
$old = 0.875$
$new = 0.125$

$$f(n) = (0.875)^n(150) + \sum_{i=1}^{n}[(0.125)(0.875)^{n-i}(75)]$$

Wrote the following python script to generate the results below

```
def question3():
    for n in range(1, 10):
        sum_iter = sum([0.125 * (0.875 ** (n - i)) * 75 for i in range(1, n + 1)])
        f = (0.875 ** n) * 150 + sum_iter
        print(f'f({n}) = {f} ms')


if __name__ == "__main__":
    question3()
```

Output:

```
f(1) = 140.625 ms
f(2) = 132.421875 ms
f(3) = 125.244140625 ms
f(4) = 118.963623046875 ms
f(5) = 113.46817016601562 ms
f(6) = 108.65964889526367 ms
f(7) = 104.45219278335571 ms
f(8) = 100.77066868543625 ms
f(9) = 97.54933509975672 ms
```

Question 4

RTT = 250 ms except every Nth RTT

For Nth RTT = 500 ms

SRTT weight = 0.875, 0.125
RTT weight = 0.75, 0.25

If N is very large -> at 500 ms RTT -> TIMEOUT
If N is small -> no timeout

Timeout occurs if RTT > RTO

To calculate STRR, we use:

$$SRTT = \alpha SRTT + (1-\alpha) R$$

To calculate RTTVAR, we use:

$$RTTVAR = \beta(RTTVAR) + (1-\beta)(|SRTT - R|)$$

To calculate retransmission time RTO, we use

$$RTO = max(RTT_{min}, SRTT + 4(RTTVAR))$$

```
from tabulate import tabulate

ALPHA = 0.875
BETA = 0.75
RTT = 250
RTT_NTH = 500

# Assuming
RTO_MIN = 499  # as RTO_MIN < 500

WARM_UPDATE = 100
TRIALS = 1000


def question4():
```

```
    headers = ["N", "RTT", "SRTT", "RTT_VAR", "RTO"]

    get_new_srtt = lambda srtt_old, alpha, r: alpha * srtt_old + (1 -
alpha) * r
    get_new_rttvar = lambda old_rttvar, srtt, beta, r: beta * old_rttvar +
(1 - beta) * abs(srtt - r)

    get_new_rto = lambda srtt, rttvar: max(RTO_MIN, (srtt + (4 * rttvar)))
    get_rtt = lambda i, n, rtt, n_rtt: n_rtt if i % n == 0 else rtt

    # initially
    rtt = RTT
    srtt = rtt
    rttvar = RTT / 2  # initially
    rto = get_new_rto(srtt, rttvar)

    for n in range(1, TRIALS):
        done = False
        results = []

        print(f"{'-' * 25} N = {n} {'-' * 25}")

        for i in range(WARM_UPDATE + TRIALS + 1):
            rtt = get_rtt(i, n, RTT, RTT_NTH)

            if i >= WARM_UPDATE and rtt > rto:
                print(f' {"*" * 40}  TIMEOUT at N = {n} {"*" * 40}')
                done = True
                break

            srtt = get_new_srtt(srtt, ALPHA, rtt)
            rtt_var = get_new_rttvar(rttvar, srtt, BETA, rtt)
            rto = get_new_rto(srtt, rtt_var)

            results.append((i, rtt, srtt, rtt_var, rto))

        if done:
            break

        print(tabulate(results, headers=headers))
```

(5) (a) Slow start : from 1 to 6 && 23 to 26

(b) Triple Duplicate ACK

(c) Timeout

(d) ssthresh = 32. Congestion Avoidance starts.

(e) ssthresh = 21. CA starts at round 17 and cwnd = 21

(f) ssthresh = 21

(g) $\frac{8}{2} = 4$ ssthresh = 4.
round = 4