Aim :

To study the effect of scheduling algorithm (SCFQ /WFQ) in queue utilization, average packet delay, packet drop probability and per connection metrics like fraction of link bandwidth allotted, average packet delay etc. We have also implemented RED and studied it's effect on queue utilisation and average packet delay.

Main features of our implementation :

1. Program is implemented in C/Linux platform.

2. Program is implemented as a single thread execution and contains following modules : packet generator module, packet processor module and event scheduler module.

3. If more than one packet has the same lowest finish number, server randomly chooses one for processing.

Event scheduler module - is implemented in the main() function (scfq.c file). This function calls the respective event handlers (as function call) based on the schedule(event scheduled first will be processed first and events scheduled for the same time will be processed randomly).

The two events in the program, namely packet generation event and packet processing event will be handled by the following two modules.

Packet generator module – This is the event handler for packet generation event. This is implemented in packetgen( ) function. (pktgen.c file)

Packet processing module – This is the event handler for packet processing event. This is implemented in server( ) function. (pktproc.c file)

How to run the program :

Extract the tar file named CS09S006_CS08M015_lab7.tar.gz
Go to the concerned directory and type the following

```
# make clean
# make all
# ./scfq -in <input file name> -out <output file name> -wfq -red -wt
```

Input and output file names are compulsory. Other run time arguments, 'wfq', 'red' and 'wt' are optional.

Meaning of optional arguments:

wfq - Program will use WFQ scheduling algorithm instead of SCFQ(by default).
red - RED option gets enabled
wt – Sources are associated with weights (given in input file)

Input file format :

An input file will contain values in the following format:

```
N=4 T=10 C=100000 B=100
10 1000 1500
20 500 1200
20 750 1500
100 1000 1800
```

N – Number of sources
T – Simulation time
C - Output link's packet processing capacity (in packet length units per unit time)
B – Queue capacity in packet length units.

From second row onwards, per connection metrics are mentioned in the order : number of packets per unit time, minimum packet length and maximum packet length.

For simplicity, we assume each packet length unit as 1 byte and each time unit as 1 second. All results are written in to the output file mentioning these units. All comparisons and explanations use the above units.

If weight option is enabled, then input file should contain values in the following format:

```
N=4 T=10 C=100000 B=100
10 1000 1500   4
20 500 1200   10
20 750 1500   2
100 1000 1800   5
```

Weights are also mentioned along with other per connection metrics as the last value in each row(except first row).

Queue size samples are taken every 0.1 time units for calculating overall queue utilization.

RED Values:

Weight factor = 0.001
This is to reduce the effect of current queue size sample on the average value of queue size.

Minimum Threshold = Queue size/2
Maximum Threshold = Queue size

Max Probability = 1

This is to drop the packets uniformly when average queue size increases from minimum threshold to maximum threshold value. Optimally, queue utilisation has to be between minimum and maximum threshold values.

Sample Input file: input1, input2,.......input10
Sample Output files:output1, output2,.....output10

Refer later part of this document to see which sample files are used in each case for comparison study and analysis.

<u>Logical view of data structure which store scheduled events (an example):</u>

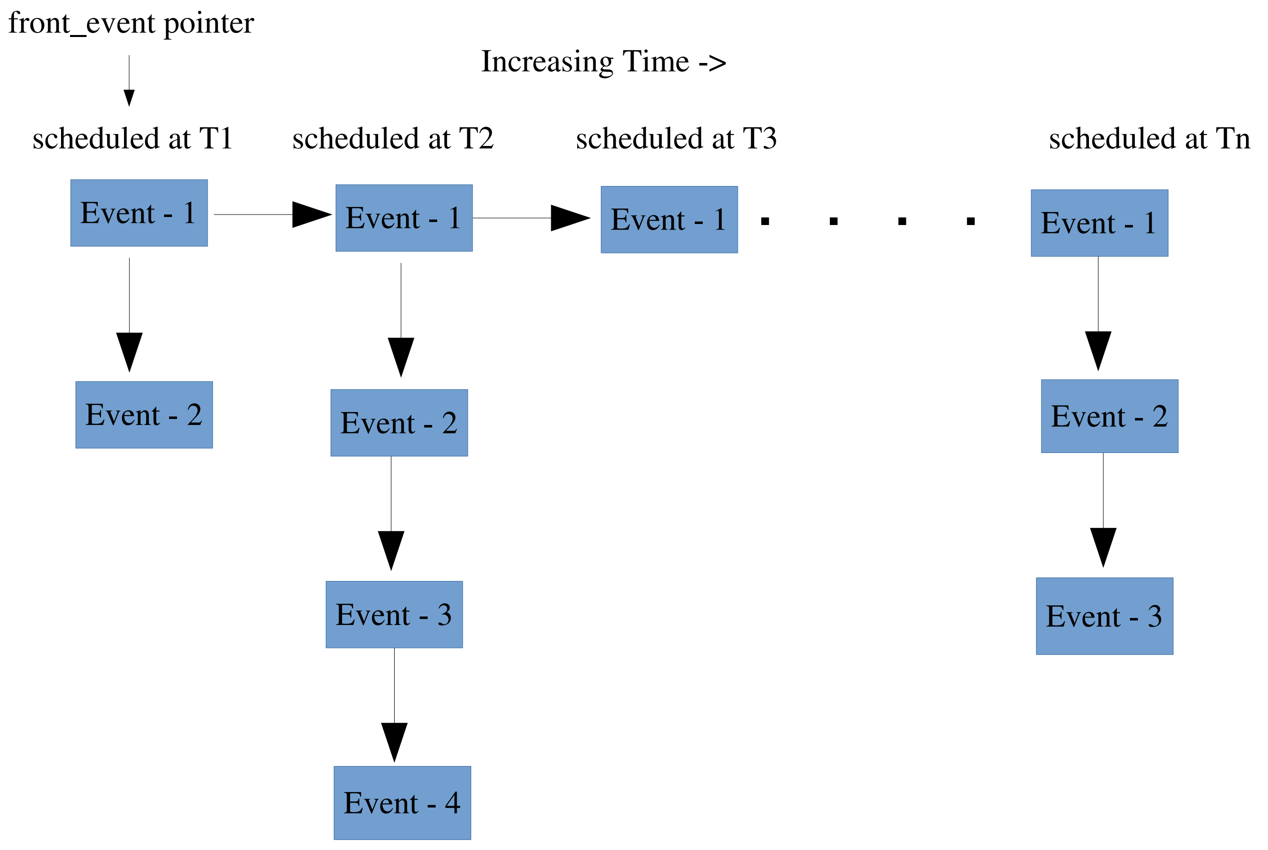

Figure 1: Event List

- Events are sorted in order of their event time.
- Events are executed in increasing order of time.
- Elements in the list are added and deleted throughout the execution of the program.
- Past events (already executed) are deleted from the list.
- Future events get added to the list (at the end or in between based on the event time).
- Events scheduled for the same time are executed in random fashion.

<u>Case - 1 (Effect of inter-arrival time on bandwidth allocation)</u>

A)

Input file name - input1
Output file name - output1

Input values -
N=4 T=100 C=10000 B=100000
10 1000 1500
10 900 1400
10 950 1500
10 1000 1500

Here we gave same inter-arrival time (of packets) for all sources and packet size range almost similar across all sources. We found that all sources were getting almost equal share of bandwidth of around 0.25.

B)

Input file name - input2
Output file name - output2

Input values -
N=4 T=1000 C=10000 B=100000
100 1000 1500
20 900 1400
100 950 1500
20 1000 1500

When we gave different inter-arrival time for sources, we observed that sources with less inter-arrival time seems to acquire more bandwidth. This is because, calculation of finish number depends on currently executing packet's finish number. As a result, packets of sources with smaller inter-arrival time get smaller finish numbers compared to others. Hence, more share of bandwidth.

<u>Case – III (Effect of rate on queue utilisation)</u>

A)

Input file name - input3
Output file name - output3

Input values -
N=4 T=50 C=10000 B=100000
10 1000 1500
20 900 1400
30 950 1500
10 1000 1500

B)

Input file name - input4
Output file name - output4

N=4 T=50 C=10000 B=100000
2 1000 1500
3 900 1400

5 950 1500
3 1000 1500

For A, we have given higher packet rate across all sources and for B, we have given lower packet rate across all sources. We have observed a substantial difference in average queue utilisation between these two input values. A has got larger queue utilisation (98352.367) compared to B (82761.969).

Case – IV (Effect of queue size and rate on average packet delay)

A)

Input file name - input5
Output file name - output5

Input values -
N=4 T=50 C=10000 B=10000
2 1000 1500
3 900 1400
5 950 1500
3 1000 1500

B)

Input file name - input6
Output file name - output6

Input values -
N=4 T=50 C=10000 B=100000
10 1000 1500
20 900 1400
30 950 1500
10 1000 1500

We have two inputs, one with lower rate and lower queue size(A) and one with higher rate and larger queue size(B). As expected, for A, we have observed lesser average packet delay(943.450ms) compared to B (8849.551ms).

Case – V (Effect of WEIGHTS on bandwidth allocation and average packet delay)

Input file name - input7
Output file name - output7

Input values:
N=4 T=50 C=10000 B=100000
2 1000 1500 4
3 900 1400 5
5 950 1500 15
3 1000 1500 1

Here, apart from usual metrics per source, we have given weights to each source. We have observerd that source with higher weight got higher bandwidth share and lower average packet delay. For example, source-2 (counting from 0), got bandwidth allocation of 0.469 compared to source-3 which got only 0.071. Also, average packet delays where 263.791ms and 28662.247ms respectively.

Case – VI (Effect of RED on queue utilisation and average packet delay)

Input file name - input8
Output file name - output8 (without RED)
Output file name - output8a (with RED)

Input values :
N=4 T=50 C=10000 B=100000
2 1000 1500
9 900 1400
5 950 1500
8 1000 1500

We ran the program for the same input with(output8a) and without enabling RED (output8). We have observed that by enabling RED, we are getting a better queue utilisation and average packet delay.

Queue utilisation (without RED) - 94627.305
Queue utilisation (with RED) - 78837.008

Average packet delay (without RED) – 7751.674ms
Average packet delay (with RED) – 7586.251ms

Case – VII (Comparison between SCFQ and WFQ)

A) Non-weighted sources

Input file name - input9
Output file name - output9 (with SCFQ)

Output file name - output9a (with WFQ)

Input values :
N=4 T=100 C=10000 B=100000
100 1000 1500
2 900 1400
100 950 1500
2 1000 1500

We observed that, over a large period of time, SCFQ and WFQ performance were very similar with comparable bandwidth allocation, average packet delay etc. **But, worst case delay per source were higher for SCFQ compared to WFQ.**

For example, for source - 0
Worst case delay (with SCFQ) - 13153.267ms
Worst case delay (with WFQ) - 11704.491ms

For example, for source - 1
Worst case delay (with SCFQ) - 439.600ms
Worst case delay (with WFQ) - 238.800ms

B) Weighted sources

Input file name - input10
Output file name - outpu10 (with SCFQ)
Output file name - output10a (with WFQ)

Input values:
N=4 T=100 C=10000 B=1000000
2 1000 1500 2
3 900 1400 50
4 950 1500 1
3 1000 1500 2

Explantion for A holds good for B as well.
For example, for source - 0
Worst case delay (with SCFQ) - 999.813ms
Worst case delay (with WFQ) - 405.101ms

For example, for source - 1
Worst case delay (with SCFQ) - 418.467ms
Worst case delay (with WFQ) - 275.284ms