Prepared by Cem KARACA, cannot be copied, or used without prior permission of author.

INTRODUCTION

To improve network performance, multi queuing systems are started to use and the queues must be organized in a way that all the queues should be fairly used. By this approach scheduling algorithms are developed. In this research Random and other three fair queuing systems Round Robin, Shortest Queue, periodically are implemented and their respective simulation results are gathered. In order to gather information a packet source and for queues and four servers are developed and packet size and packet sending time intervals server capacities and server packet sizes are made variables, by the way the simulation can be implemented in all conditions such as some algorithms are more fair when servers are faster and some queues are reliable in slow servers. As we all know after overflow in queues we don't have much to do to fix that situation, thus overflows must be prevented.

SIMULATION STRUCTURE

In this simulation a packet server with variable packet sending time and variable packet length is used, also for the sake of real world type networks I also added Poisson packet sending times and Poisson packet lengths. Queues are FIFO (First In First Out) type and fixed to max 80 packets buffer. Also queue packet sizes and capacities are made variable in order to make simulation for all types of queues, such that if the servers operate faster than source then there is no meaning to make simulation because all the packets coming from the source will be directly routed to respective sinks and we never face a problem in order to make multiple queues.

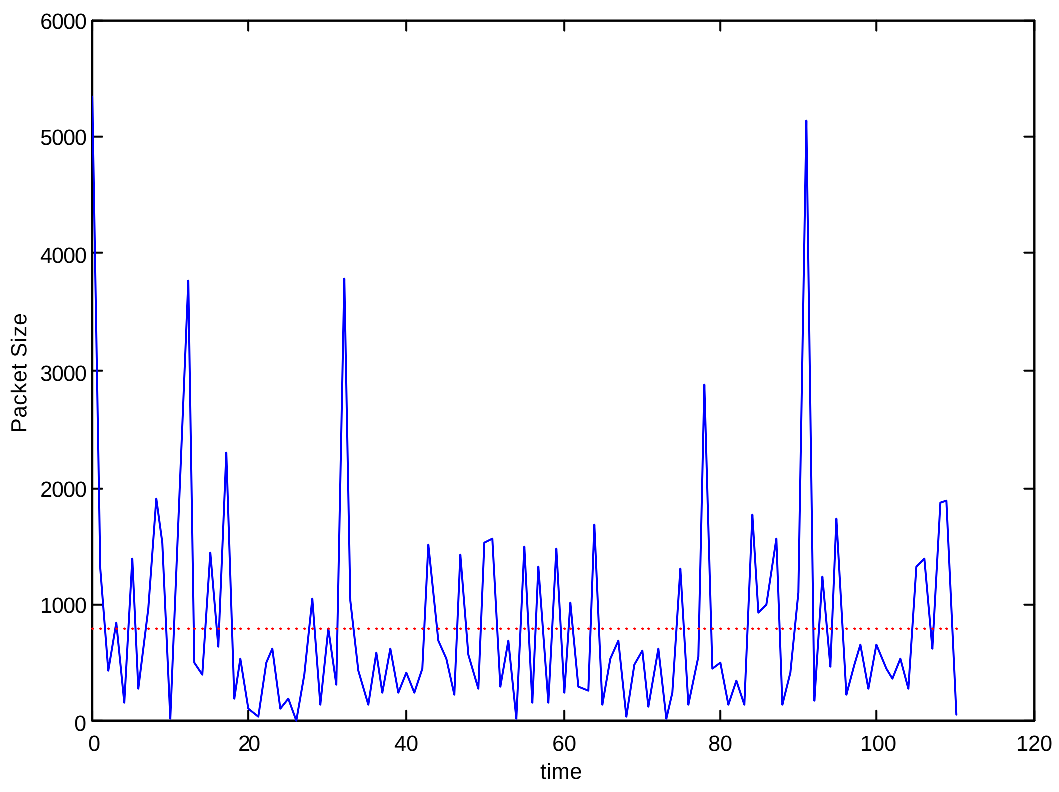
POISSON RANDOM VARIABLES

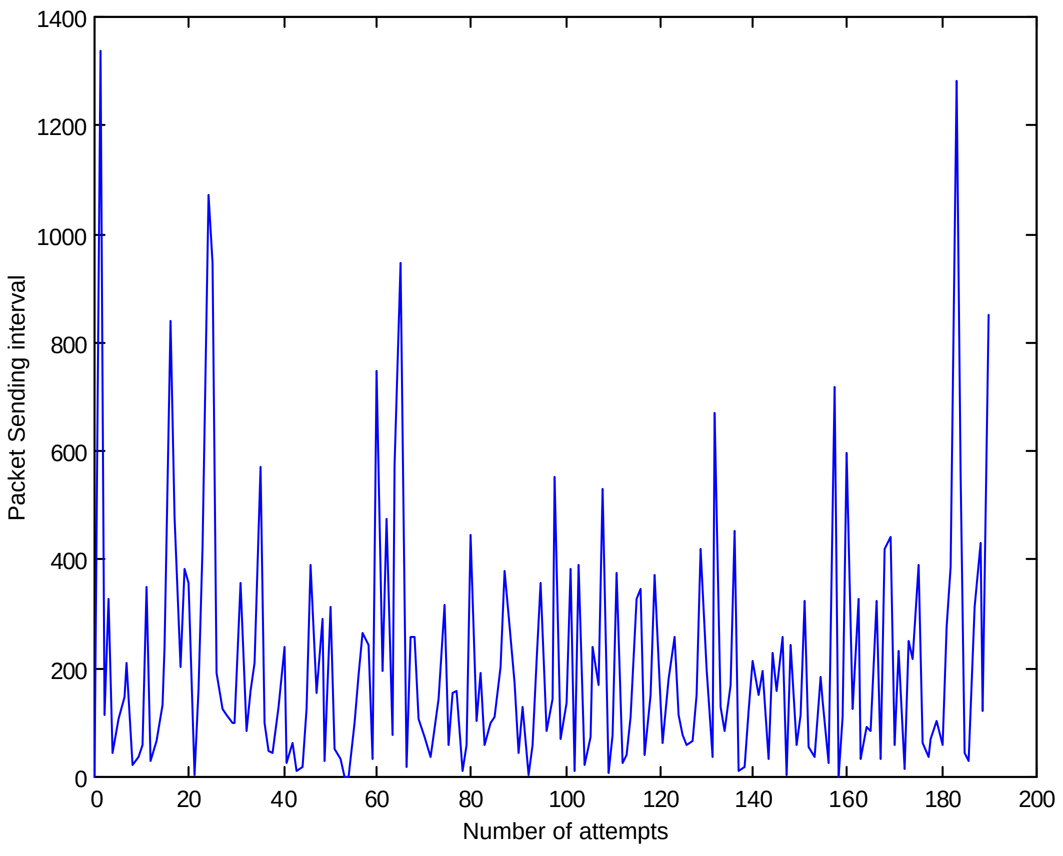
Queuing systems are random processes, this means we cannot predict at which time a packet will arrive and the size of the packet is also a random variable in a boundary.
The matching random system to real world queuing systems is negative exponential random numbers. The mathematical definition of negative exponential is:

$$f(x) = \begin{cases} l e^{-lx} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

In this formula x denotes the uniform random numbers generated between 0-1 and ? is the mean value, this means f(x) will give results near ?.
Such that for ?= { 5}:

$$f(l, x) = P_{NE}(l, x) = l e^{-lx}$$


6000
5000
4000
3000
2000
1000
0
Packet Size
0
20
40
60
80
100
120
time


1400
1200
1000
800
600
400
200
0
Packet Sending interval
0
20
40
60
80
100
120
140
160
180
200
Number of attempts

SCHEDULING ALGORITHMS

The primary objective of scheduling algorithms is to get messages transferred across network from the source to destination in a correct, simple and optimal way. In some applications such as video conferencing or streaming servers may require more than one buffer in order to reduce hard disk fragmentation fatalities. By the way a scheduling algorithm must be simple in order to make scheduling faster.

Round Robin Scheduling

It is one of the oldest, simplest, and fairest and most widely used scheduling algorithms, designed especially for time-sharing systems. A small unit of time, called time slices or **quantum** is defined. All run able processes are kept in a circular queue. The packet scheduler goes around this queue, allocating the packets to each queue for a time interval of one quantum. New processes are added to the tail of the queue.

If the process is still running at the end of the quantum, the packet is preempted and the process is added to the tail of the queue. If the process finishes before the end of the quantum, the process itself releases the packet voluntarily.

Shortest Queue

Shortest queue is the fairest scheduling algorithm because the algorithm searches for the shortest work remaining in the queues and puts the work to that queue. Especially shortest queue algorithm is better when Poisson arrival time and Poisson packet sizes are used. Here is the algorithm for shortest queue:

```
if(q1<=q2 && q1<=q3 && q1<=q4) min =0;
if(q2<=q1 && q2<=q3 && q2<=q4) min =1;
if(q3<=q1 && q3<=q2 && q3<=q4) min =2;
if(q4<=q1 && q4<=q2 && q4<=q3) min =3;
```

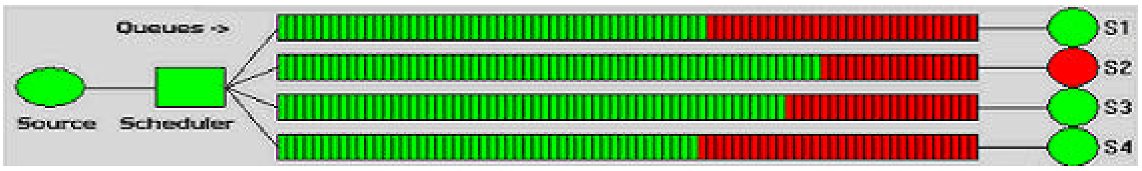
Random Queue

Random queue algorithm simply selects a queue randomly, this kind of algorithms may be efficient on high load large queue servers in order to reduce calculation time.

Periodically Queuing

Periodically Queuing is similar to Round Robin but it uses minimum quanta or time division, in Round Robin queues are switched by time but in periodically queuing instead of time packets are switching the queues.

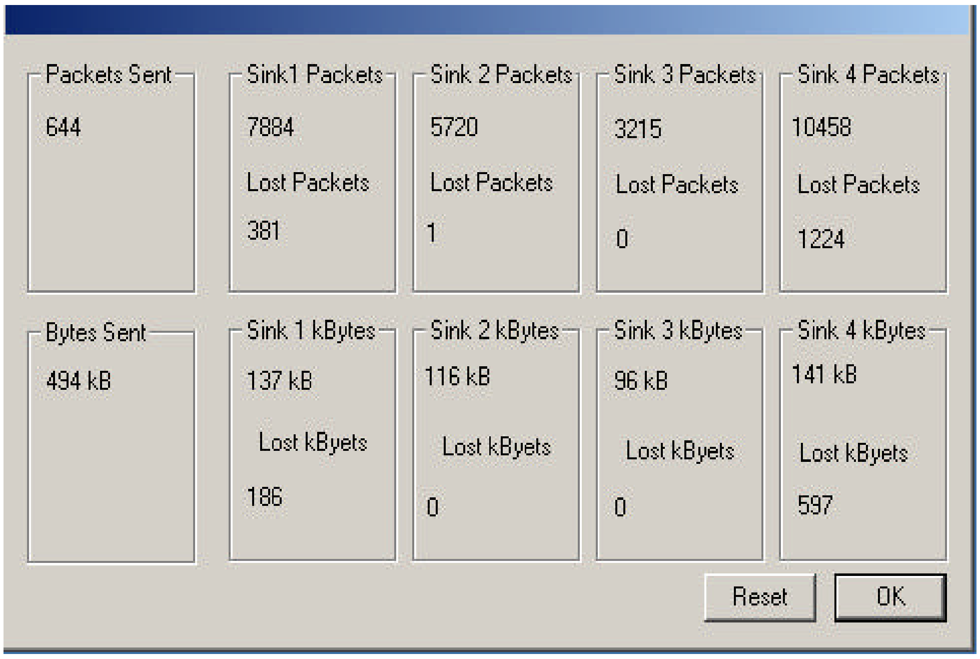
SIMULATION



Simulation is made using Visual C++ and in order to make simulation more flexible and efficient **multi-thread** programming is used. *Threads*, sometimes called *lightweight processes* are indepedendently scheduled parts of a single program. We say that a task is *multithreaded* if it is composed of several independent sub processes which do work on common data, and if each of those pieces could (at least in principle) run in parallel.
Threads allow a programmer to switch between lightweight processes when it is best for the program. (The programmer has control.) A process which uses threads does not get more CPU time than an ordinary process - but the CPU time it gets is used to do work on the threads. It is possible to write a more efficient program by making use of threads.
Inside a heavyweight process, threads are scheduled on a FCFS basis, unless the program decides to force certain threads to wait for other threads. If there is only one CPU, then only one thread can be running at a time. Threads context switch without any need to involve the kernel - the switching is performed by a user level library, so time is saved because the kernel doesn't need to know about the threads.
In simulation there are six threads, one main thread which handles the window drawings and system messages, one thread for the packet source, four threads for queues (independently) and one thread for all sinks.

Statistics

Statistical data is taken from a dialog box as shown in the figure below:



Packets Sent	Sink1 Packets	Sink 2 Packets	Sink 3 Packets	Sink 4 Packets
644	7884	5720	3215	10458
	Lost Packets	Lost Packets	Lost Packets	Lost Packets
	381	1	0	1224
Bytes Sent	Sink 1 kBytes	Sink 2 kBytes	Sink 3 kBytes	Sink 4 kBytes
494 kB	137 kB	116 kB	96 kB	141 kB
	Lost kByets	Lost kByets	Lost kByets	Lost kByets
	186	0	0	597

The main purpose of this project is to gather data from different types of scheduling algorithms by making a simulation. Simulation itself must be efficient in a way so the simulation is making deal with time.

Conclusion

By the help of simulation I observe that some algorithms are better in some conditions, such as random scheduling is the cheapest algorithm but we cannot guarantee the packets to be lost or not, also random scheduling is ideal for large queued slow servers, Round Robin algorithm is more flexible than others, that means, it can be adapted to all type of servers by configuring the quanta amount, but requires stochastically gathered server legend in order to find suitable quanta value. Shortest queue needs more calculation power because this algorithm makes more comparisition than the others but very fair kind of algorithm. Especially gives better results when Poisson packet sizes and arrival times are used. Periodically scheduling is a little control less than the others, this algorithm simply selects next queue but when Poisson packet sizes are used, after a while some queues works are getting larger and some queues sits idle. In my opinion the best method is to use random scheduling when servers are in light work and if work amount gets bigger the servers should switch to shortest queue algorithm, by this method the cost will be less and the packet arrivals should be guaranteed.

Cem KARACA
cem.karaca@emu.edu.tr