CAS CS 655
Introduction to Computer Networks
Fall 2015

Assigned: September 16 Due: Thursday, October 15, 1:00pm

Simulation of FIFO and Fair Queuing (FQ) Algorithms

Introduction

A router output port can be modeled as a packet queue that is serviced using a certain service discipline, the simplest being first-in-first-out (FIFO). However, the FIFO discipline does not provide any throughput or delay guarantees that may be required in high-performance networks. Significant research and development has been done to rectify this problem by devising alternate service disciplines, collectively termed "Fair Queuing Disciplines" (algorithms). In fact, many commercial routers, such as Cisco's, now implement FQ-type algorithms. The aim of this project is to simulate and study FIFO queuing and two fair queuing algorithms, namely packet-by-packet round robin (RR) and deficit round robin (DRR), and compare their performance under different traffic arrivals.

Read the DRR scheduling paper available online. In brief, RR serves in turn one packet for each active user (i.e. user whose queue of packets is non-empty). This is unfair if packets are of different length. DRR maintains a quantum (in bits) for each user (flow). In turn, a packet is transmitted from the next queue if the flow's current quantum value is greater than the packet size (the quantum is reduced by the size of the served packet). The quantum is accumulated at each round (typically by the maximum packet size), if the flow remains active.

Note that scheduling (multiplexing) is a function that recurs at all levels of network architecture, for example we could have considered the same algorithms for scheduling at the application level, such as a web server.

You may work in teams of two for this assignment.

Overview

You should at least be familiar with discrete-event simulation concepts (covered in our undergraduate CS 350 class or equivalent). On Piazza, you may find helpful notes on basic concepts such as simulation clock, events, and event queue. A sample C code is also provided to show the basic simulator engine, the routines to initialize the simulation and process events, as well as FIFO. You are free to use this code as a guide (at your own risk) or write your own (possibly in a different language).

There should be two main types of concurrent events in the simulation. (You may have to define more event types.)

Packet Arrival Events: The sources generate packets of different lengths, and at different arrival times. Each packet generation is an event (say type PKT_ARV). Every time this type of event is pulled from the event queue and processed, a packet should be put in an appropriate queue. The queue chosen for the outgoing packet depends on the fair queuing algorithm. Note that FIFO has only one packet queue, so the packet is placed at the tail of a linked list that represents the queue of packets waiting to be transmitted. In case of RR and DRR, there is a separate queue for each source and the current packet is placed at the tail of the queue of the source that generated it.

Note that when the simulation is initialized, the first packet arrival events for each source must be scheduled. As the simulation progresses, current arrival events would automatically schedule the future arrival events. To do this, allocate an event structure and fill in the fields such as the length of the packet and its arrival time that are computed according to the source parameters. This event structure has to be enqueued in the event queue so that it can be dequeued in the future. You will generate arrival events according to an exponential distribution.

Link or Transmission Events: When this event (say type PKT_TXED) is processed as current event, the current packet being transmitted on the link is completely transmitted. The next transmission is then scheduled by inserting the new event. Its time is initialized using the packet length and the link speed parameters. The packet is pulled out of the appropriate packet queue for transmission depending on the queuing discipline. You may have to maintain a flag that indicates if the link is IDLE or BUSY at any given instance. The link's state IDLE indicates that there are no packets to transmit.

Simulation Engine

The simulation engine should implement the various functions for creating and managing the event queue such as creating an event queue, getting the "next" (earliest) event from the event queue, getting a free event structure, inserting an event in the event queue, freeing (pulling out) an event from the event queue, and freeing the event queue. Note that there is always only one event queue and its operation is independent of the scheduling discipline being simulated. Please define the fields in your event structures appropriately as required for various algorithms. Fields such as packet ID, length of packet, arrival time and more will be required. You may have to add fields for statistics collection.

Description of Traffic Sources

You will create 10 sources, of which 4 are of type "telnet" and have an average packet size of 512 bits and the other 6 are of type "ftp" and have an average packet size of 8192 bits.

In addition, you will create an ill-behaved "rogue" source with any packet length you like (say, 5000 bits). The total data rate of the output link R is 1 bps (this makes your life simpler!). Each "telnet" and "ftp" source generates data at the same rate r. The source "rogue" generates data at rate $R/2$. The offered load of a source is given by r/R and the total offered load M is the sum of the offered load of all sources *excluding "rogue"*. In your experiments, you must vary the total offered load M from 0.4 to 2.0 in increments of 0.2. The offered load of the "rogue" source is constant and is always 0.5 throughout all the experiments. For each source, for a given rate r and a mean packet length L, the mean interarrival time I is given by L/r. Knowing L and I, you can generate events (packets) with the length distributed exponentially with mean L, and the interarrival time also distributed exponentially with mean $I = L/r$.

Experiments

For each queuing algorithm that you will simulate (i.e., FIFO, RR, DRR), do the following.

At each value of load M, run the simulation until (say) 100,000 packets have been generated in all. For each source i, let the total number of bits transmitted be $B(i)$, $A(i,0)$ the arrival time of the first packet, and $F(i)$ the time at which the last bit of its last packet was sent out (transmitted). Then $B(i)/[F(i)-A(i,0)]$ is the measured rate (throughput) that the source obtained. Calculate the difference between this measured rate and the actual normalized generation rate or offered load of the source. For each source, plot this difference against different values of M. Also measure for each source the average packet latency (queuing delay). The latency (waiting time) of a packet is the duration from its arrival time to the time it is considered for transmission. The average latency

is the mean of these measured latencies. Plot for each source its average latency for each value of *M*.

Feel free to run other experiments as well and convince yourself that FIFO is not a good scheme, RR does better, and DRR improves over this. In your report give reasons for the above based on your experiments. Also compare the time and space complexities of the three scheduling algorithms. From your results, do you think it is justifiable to "pay" for a more complex algorithm for improvements in fairness that you obtained?

You will submit a printed copy of your report that contains the following:

- Program documentation that contains a description of the program design.
- The plots along with your description, observations, and inferences, and answers to the questions raised. Your plots must show 90% confidence intervals for the performance metrics. (You may refer to these online notes on confidence intervals.)

Drop your hard copy in the slotted inbox labeled "CS 455/655" in the hall outside MCS-137.

Also submit an **electronic copy** of your code (ASCII files only!) using gsubmit – name your submission directory "pa0".

If you choose to have a partner, please put both names on the printed report, and indicate which one of you submitted the electronic copy. Make sure each file in your electronic copy contains both names as well. Also, clearly describe the role of each team member in your documentation.

The following grading policy will be used for the purpose of this project:

- correct operation 45%
- documentation 15%
- results and report 40%