CS655 Pa0 Report

Documentation

Design of simulator and Statistics collection method:

According to the discrete-event simulation method, we designed three classes to model the simulation system, three classes to simulate FIFO, RR, DRR separately, and a class to deal with the outputs.

The core of the simulation is a random generator, using the following formula:

```
Exponential(s) =
        return (-s * ln(Random));
```

to generates a random number exponential distribution with a given number, which is produced by "Random()" method according to system's current time.

Our main() function is a simple function that defines a clock to note the simulation and includes an interface for users to determine M value and packets number to simulate. Besides, we also initial 11 sources (4 for telnet, 6 for ftp and 1 for a rogue), and set 512, 8192, 5000 bits of packet size separately. After determining M value, the main () cyclically allocates FIFO, RR, DRR simulation classes to simulate from M=0.4 to the set M value, and outputs the result throughputs, latencies and differences.

The flow of control is defined as professor said. While simulation is running, after initialing the clock, state, and performance counters, each time the simulator picks up (e, t) tuple with minimum t from EventQueue, and call routine (e, t). To do these, we defined a "compare to" function to sort Event with ascendant arrival time. Besides, in every simulation module, we included a Routine () function to routine an event according to the event's type (Arrival or Transmission/Departure). Accompanied with the routine (), we also defined a schedule () function to schedule an event, add either an arrival event or transmission event to the event list.

What' more is three simulation classes: FIFOSimulation{}, RRSimulation{}, DRRSimulation{}, their flows of control are similar to each other, but when it comes to schedule(), they use their own algorithms to schedule the events. Especially in DRR, we defined quantum, each time arrival, plus it to the remaining quantum, which can be used for the next round to pick packets from this source. During those simulations, by using the clock, we use getArrivalTime and get DequeueTime to calculate the latency, and use totalBits/(lastTransTime - firstArrivalTime) to calculate the throughputs.

At last, "WriteTxt" class uses FileWriter to write outputs.

How to run simulator:

Our simulator has only one ".java" file, when start to simulate, use "javac" command to compile this java file, and there will be several ".class", use" java" command to run the

“simulator” class, and input M value, packets number and Start to start the simulation.

```
➜  PHW0  javac simulator.java
➜  PHW0  java simulator
Input a max value of "M" to simulate:0.4/0.6/0.8/1.0/1.2/1.4/1.6/1.8/2.0
0.4
Input packets(if use default, please input 100000):
10
Input "Start" to start Simulation, Input"Exit" to quit
Start
M=0.4
```

How to run graph generator:

The program plot.R is a graph generator written by R language’s ggplot2 engine. It reads in the outputs of our simulator, rebuilds the data structure and produces difference against different values of M, average latency against different values of M for each source. What’s more, according to our several simulations, after selecting the normal quantiles, we plot 90% confidence interval in the graphs. To run the graph generator, use R and set work path and just run the plot.R script.

Correctness

1. FIFO under light load (M = 0.4, rogue rate = 0.5):
   total throughput close to 0.9

source	FIFO-Throughput
telnet0	0.03998
telnet1	0.03998
telnet2	0.03998
telnet3	0.03998
ftp4	0.04003
ftp5	0.04003
ftp6	0.04003
ftp7	0.04003
ftp8	0.04003
ftp9	0.04003
rogue10	0.49988
sum	0.89998

   As the result showed, total throughput = 0.89998 < 0.9.

2. FIFO under high load (M = 2):
   (1) Total throughput does not exceed 1

type	FIFO-Throughput
telnet0	0.08
telnet1	0.08
telnet2	0.08
telnet3	0.08
ftp4	0.08
ftp5	0.08
ftp6	0.08
ftp7	0.08
ftp8	0.08
ftp9	0.08
rogue10	0.2
sum	1

As the result showed, total throughput = 1.

(2) Each source should get throughput in proportion to its sending rate

type	FIFO-Throughput
telnet0	0.06667
telnet1	0.06667
telnet2	0.06667
telnet3	0.06667
ftp4	0.06672
ftp5	0.06672
ftp6	0.06671
ftp7	0.06671
ftp8	0.06671
ftp9	0.0667
rogue10	0.33341

At M = 1, rogue gets 0.33, a Telnet gets 0.067 and an FTP source gets 0.067.

(3) All sources experience similar delay, but compared to DRR, a Telnet source suffers most delay due to higher FTP volume and the interference with higher-rate rogue.

type	FIFO-Latency	DRR-Latency
telnet0	37067118.6	30859562.4
telnet1	37067576.6	31436152.4
telnet2	37068034.6	30883371
telnet3	37068492.6	30912386.6

As the result above, a Telnet source from FIFO is almost 37067000 > 31000000 which is Telnet source delay from DRR.

3. RR and DRR, light load (M=0.4):
   rogue source isolated, each source gets throughput close to its sending rate.

type	FIFO-Throug	FIFO-Differer	FIFO-Latency	RR-Throughp	RR-Differenc	RR-Latency	DRR-Throug	DRR-Differer	DRR-Latency
telnet0	0.03998	0.00002	16138.4	0.0025	0.0375	780967992	0.03999	0.00001	1034328.4
telnet1	0.03998	0.00002	16596.4	0.0025	0.0375	780968450	0.03991	0.00009	6812336.3
telnet2	0.03998	0.00002	17054.4	0.0025	0.0375	780968908	0.03848	0.00152	7404663.4
telnet3	0.03998	0.00002	17512.4	0.0025	0.0375	780969366	0.03974	0.00026	7860773.2
ftp4	0.04003	0.00003	1664.1	0.04	0	0	0.03966	0.00034	8405865.6
ftp5	0.04003	0.00003	8995.1	0.04	0	7331	0.03958	0.00042	8929864.8
ftp6	0.04003	0.00003	16326.1	0.04	0	14662	0.0395	0.0005	9453863.9
ftp7	0.04003	0.00003	23657.1	0.04	0	21993	0.03941	0.00059	9977863.1
ftp8	0.04003	0.00003	30988.1	0.04	0	29324	0.03933	0.00067	10501862.2
ftp9	0.04003	0.00003	38319.1	0.04	0	36655	0.03925	0.00075	11025861.4
rogue10	0.49988	0.00012	15068.8	0.4812	0.0188	792447365	0.4953	0.0047	126146106

in the above graph, for RR algorithm, FTP sources get 0.04, a rogue gets 0.4812, but, telnet sources get 0.0025, we know there is something wrong with our telnet sources, but we can't find out how to revise them, even if we run a lot of times to decrease single error's effect. For RR algorithm, both FTP and Telnet sources get normal results.

4. RR under high load (M=2):
   (1) each source gets throughput in proportion to its packet size, total throughput close to 1.

type	RR-Throughput
telnet0	0.00911
telnet1	0.00911
telnet2	0.00911
telnet3	0.00911
ftp4	0.14577
ftp5	0.14577
ftp6	0.14577
ftp7	0.14577
ftp8	0.14576
ftp9	0.14576
rogue10	0.08895
sum	0.99999

According to the result, each source indeed gets throughput in proportion to its packet size and the total throughput = 0.99999, close to 1.

(2) an FTP source gets higher throughput than a TELNET due to its bigger packet size.

From the result above, throughput of FTP source is almost 0.14578 > 0.00911, which is throughput of Telnet.

(3) delay improves for FTP compared to FIFO, since rogue source isolated. TELNET experiences higher delay due to its smaller packet size.

type	FIFO-Latency	RR-Latency
telnet0	37067118.6	218201619.4
telnet1	37067576.6	218202077.4
telnet2	37068034.6	218202535.4
telnet3	37068492.6	218202993.4
ftp4	37068137	61980165
ftp5	37075468	61987496
ftp6	37082799	61994827
ftp7	37090130	62002158
ftp8	37097461	62009489
ftp9	37104792	62016820

5. DRR under high load (M=2):

(1) each source should get close to 1/11=0.09 throughput, and unlike RR, a TELNET source gets throughput much closer to FTP since DRR tries to equalize "bits" served from each source.

type	DRR-Throughput
telnet0	0.09349
telnet1	0.09306
telnet2	0.08966
telnet3	0.09225
ftp4	0.09178
ftp5	0.09134
ftp6	0.09091
ftp7	0.09048
ftp8	0.09005
ftp9	0.08963
rogue10	0.08922

(2) improvement in delay for TELNET because of the better fairness (higher throughput) under DRR.

type	FIFO-Latency	DRR-Latency
telnet0	37067118.6	30859562.4
telnet1	37067576.6	31436152.4
telnet2	37068034.6	30883371
telnet3	37068492.6	30912386.6

The delay of DRR is much lower than FIFO for source of Telnet.

Results & Analysis

1. Correctness tests

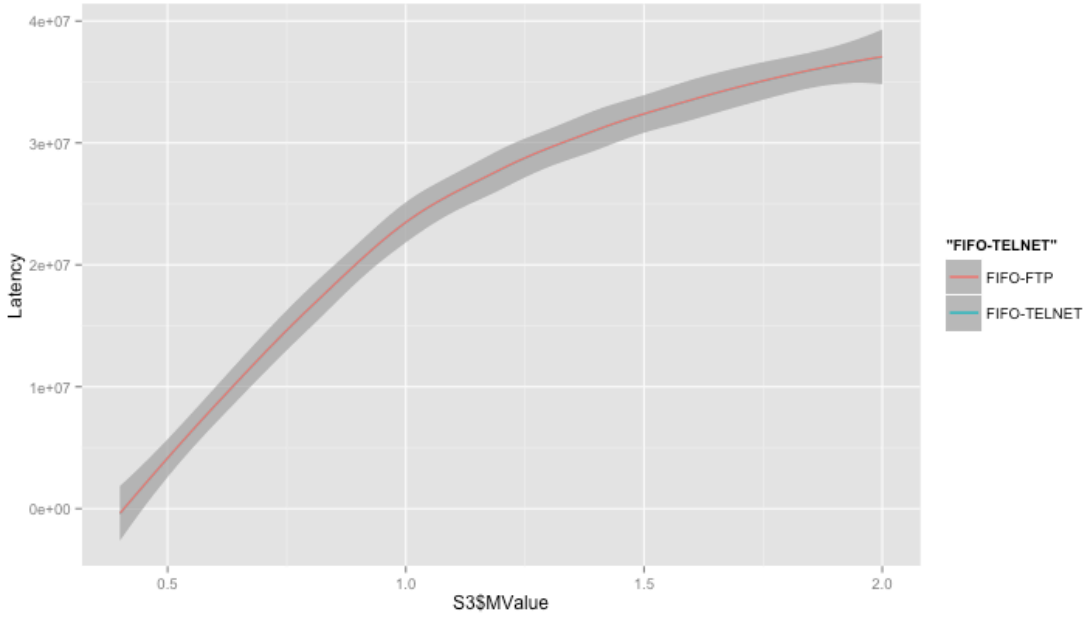
Due to M/M/1 results is for a single source and a single FCFS (infinite) queue, using Little's Law to verify the results. We could calculate the average total number Q for every source, and then multiply by the the average inter-arrival time. If the product is nearly the same as the latency time we get from the simulation, it verifies the correctness of our simulation:

(1) Compute average Q

(2) Compute average response time T and throughput λ(rate of departures)

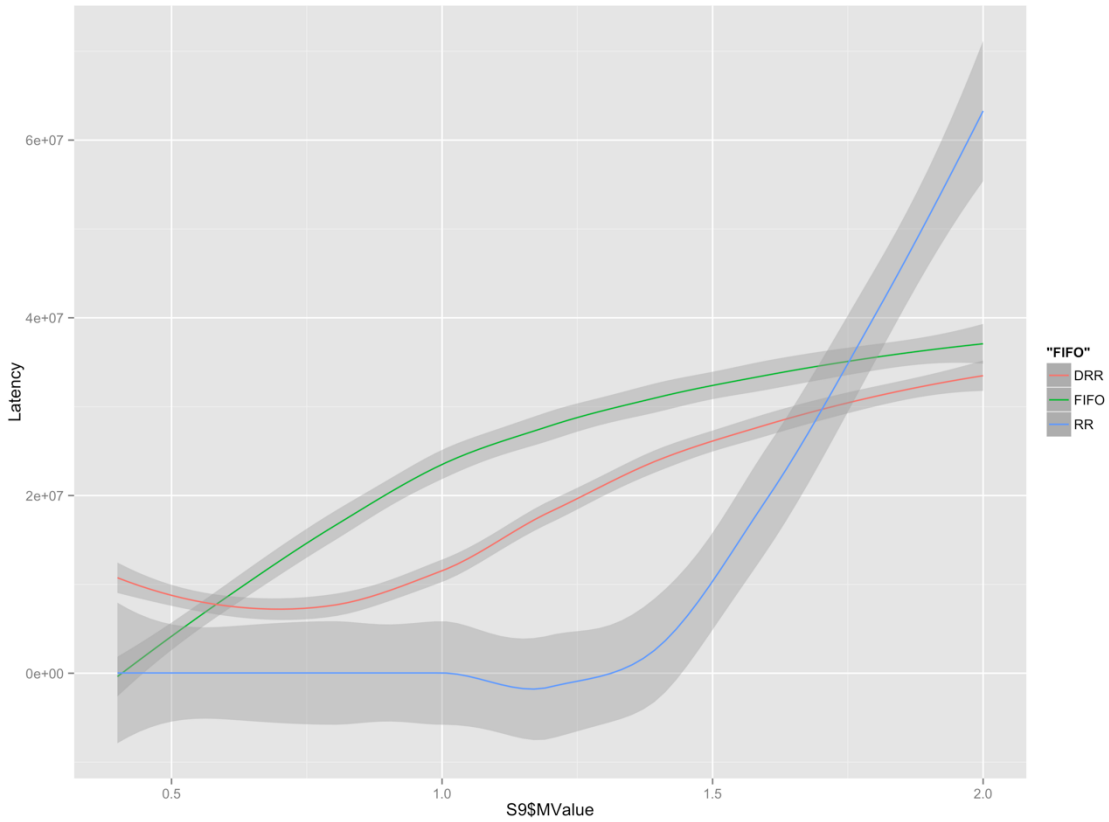
(3) Little's Law: Q = λ * T

2. As M increases under FIFO, delay increases quickly as soon as M exceeds 0.4:



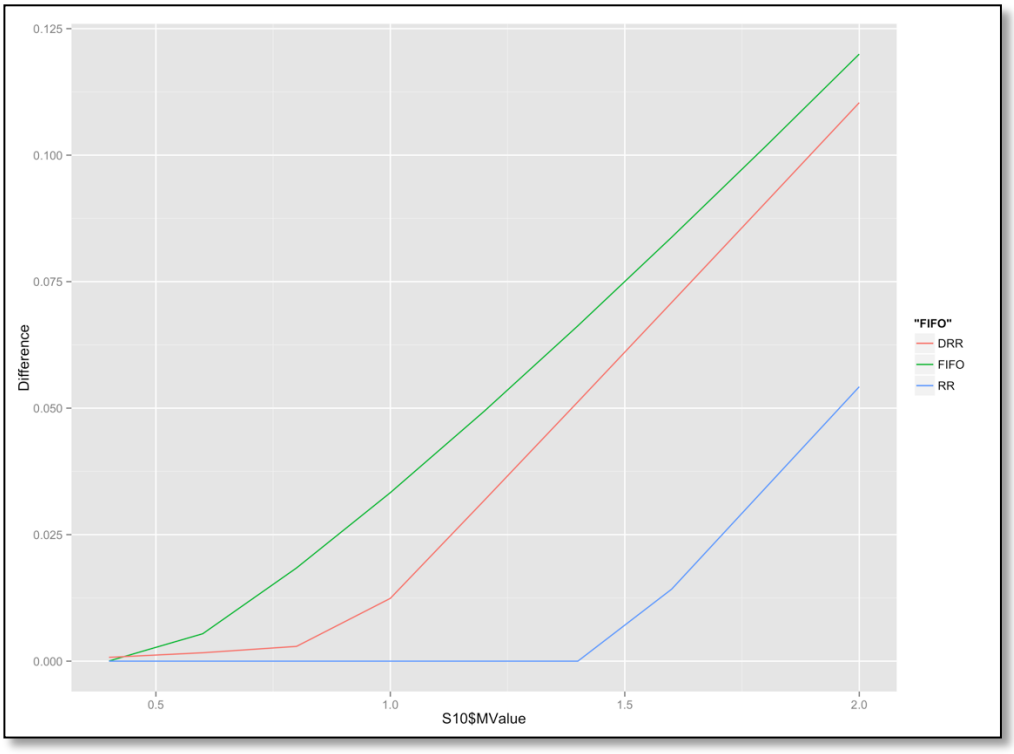
above image shows: as M increases under FIFO, latency increases quickly both for FTP and TELNET.

3. As M increases under RR or DRR, delay increases but now quickly increases only after M exceeds 0.9.

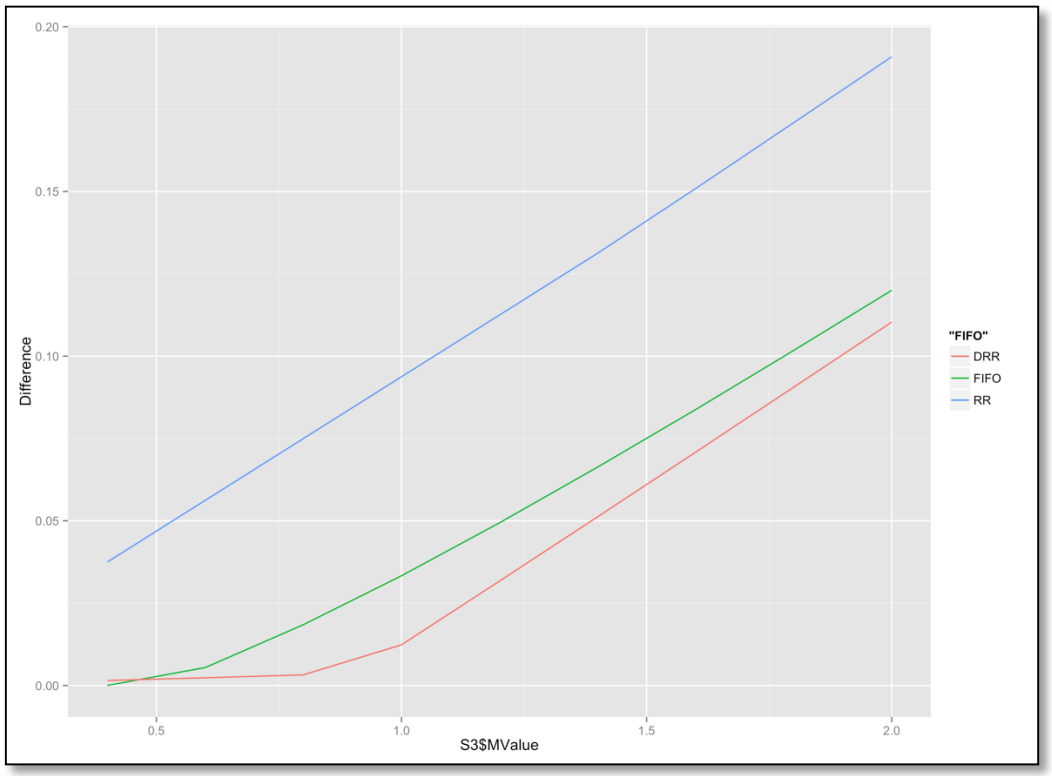


Take source9(FTP5, we set tel0, tel1 … ftp6, rogue as S1, S2 … S10, S11) for example. We could see when value of M increase, latency increase quickly showed in the graph.

4. As M increases, measured load increases but difference from offered load is bigger under FIFO for both TELNET and FTP [[or, their throughput is higher under RR and DRR.



above image shows FTP source's Difference against increasing M values. It's obviously that difference from offered load is bigger under FIFO.



above image shows Telnet source's Difference against increasing M values. It's obviously that difference from offered load under FIFO is bigger than DRR, but there RR is bigger than both DRR and FIFO, there is something wrong with RR with Telnet sources.

5. FIFO is not fair, RR isolates the rogue source, DRR is even more fair when packet sizes not same

   FIFO is not fair because it doesn't provide any throughput or delay guarantees due to the difference of packet arrival-time. The faster the packet arrives, the bigger throughput it gets. RR is more favorable for large packet size(like rogue source) because every round each source could send one packet. The bigger the packet size is, the bigger throughput it gets. DRR set quantum for each queue and

6. compare time and space complexities:

   All the time complexities of FIFO, RR and DRR is O(1) because every time just one packet has been sent from the head of queue. The space complexity of RR is O(n) because we need to set a pointer for n queues to see whether it is rounded. The space complexity of DRR is O(n) because we need to set a "quantum" to limit the packet size and a "deficit" counter to store quantum being left for n queues, O(2n) = O(n).

7. Argument on whether DRR is worth it, or is RR enough

   DRR is worth it because when the difference of packet sizes is huge(for example huge packet size of rogue), RR is not fair for the packet sent from Telnet, which results in huge latency and low throughput for Telnet. DRR is much fair for different packet size source and controls the latency well.

8. Confidence intervals
   In the experiment we could get a steady-state average performance of latency, difference and throughput, set it as $\overline{x}$. Due to the results obey normal distribution. We could compute a confidence interval around $\overline{x}$ within which the true mean μ lies with probability 1- α .

$$P(\overline{x} - \Delta \leq \mu \leq \overline{x} + \Delta) = 1 - \alpha$$

   it can be re-written as: $P(\mu - \Delta \leq \overline{x} \leq \mu + \Delta) = 1 - \alpha$

   And we have $\Delta = \mathrm{d}s_{\overline{x}}$, $s_{\overline{x}} = \sigma/\sqrt{N}$. Although we don't know the true distribution,

we could calculate an estimate of σ as:

$$\widehat{\sigma} = \sqrt{\frac{\sum(x_i - \overline{x})^2}{N-1}}$$

We set N = 6(means we calculate every average performance from 6 experiments and store in one csv file). As N is small, we have :

$$d = t_{N-1,1-\frac{\alpha}{2}}$$

As we need 90% confidence interval, according to the student t-distribution table, we could get d = 2.015. And then we could get the confidence interval for each value of different source, which are shown in the graphs with shadow.

i. All the experiment data is attached in the mean data folder.
ii. All the graphs needed are attached in graph folder.
iii. We still have remaining error about RR algorithm by the time we submit the pa0, but the other algorithms, DRR and FIFO are correct.