POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Round-Robin Scheduler

PROJECT IN
TRAFFIC THEORY

Author: **Tuğrul KÖK**

Student ID:	221047
Instructor:	Ilario FILIPPINI
Academic Year:	2022-23

Contents

List of Figures

1. Implemenation of the Code

In this chapter how a Round-Robin scheduler based on the model presented as SourceQueueSink is implemented will be shown.

1.1 How the Message Processing Time is Modeled?

In this project, a parameter "MsgSize" is defined in the Source. This MsgSize is an integer value that shows how many service times that message needs to be served. For example, if the MsgSize is 2, the packet needs 2 $e^{avgServiceTime}$ to be processed.

intuniform function is used to create random uniform packets. The expected value of MsgSize is controlled by a parameter "meanofMsgSize", which can be assigned from omnetpp.ini file.

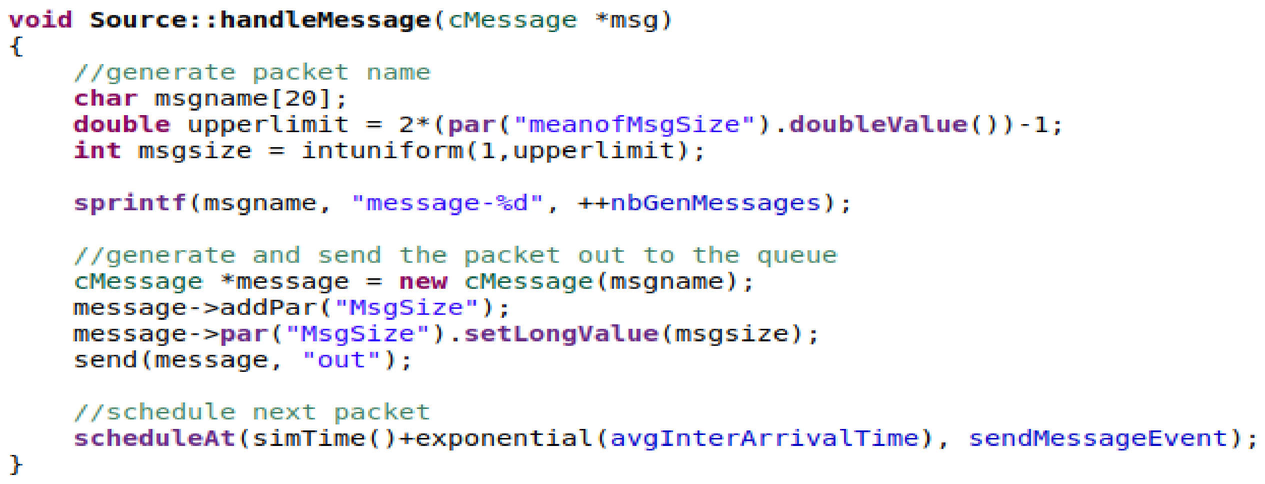
The minimum value of MsgSize is 1. The mean of the uniform distrubition can be find by:

$$Mean = \frac{1 + upperlimit}{2}$$

Then:

$$upperlimit = 2 * Mean - 1$$

By using this formula, we can assign an upper limit according to desired mean, which was decided in omnetpp.ini as "meanofMsgSize" The edited code is shown in the next figure:



```
void Source::handleMessage(cMessage *msg)
{
    //generate packet name
    char msgname[20];
    double upperlimit = 2*(par("meanofMsgSize").doubleValue())-1;
    int msgsize = intuniform(1,upperlimit);

    sprintf(msgname, "message-%d", ++nbGenMessages);

    //generate and send the packet out to the queue
    cMessage *message = new cMessage(msgname);
    message->addPar("MsgSize");
    message->par("MsgSize").setLongValue(msgsize);
    send(message, "out");

    //schedule next packet
    scheduleAt(simTime()+exponential(avgInterArrivalTime), sendMessageEvent);
}
```

Figure 1: Modelling the Message Size

1.2 Omnetpp.ini Configuration

In the omnetpp.ini file, the boolean parameter “isRoundRobin” decides the type of system, if isRoundRobin is true, it means that system is configured with Round-Robin scheduler, if it is false, then the system is configured as M/M/1. “meanofMsgSize” is also set from here. We can decide the utilization factor of the system by assigning “meanofMsgSize”, “avgInterArrivalTime”, and “avgServiceTime”.

$$\rho = \lambda E[S]$$

Where ρ is the utilization factor

λ is the arrival rate

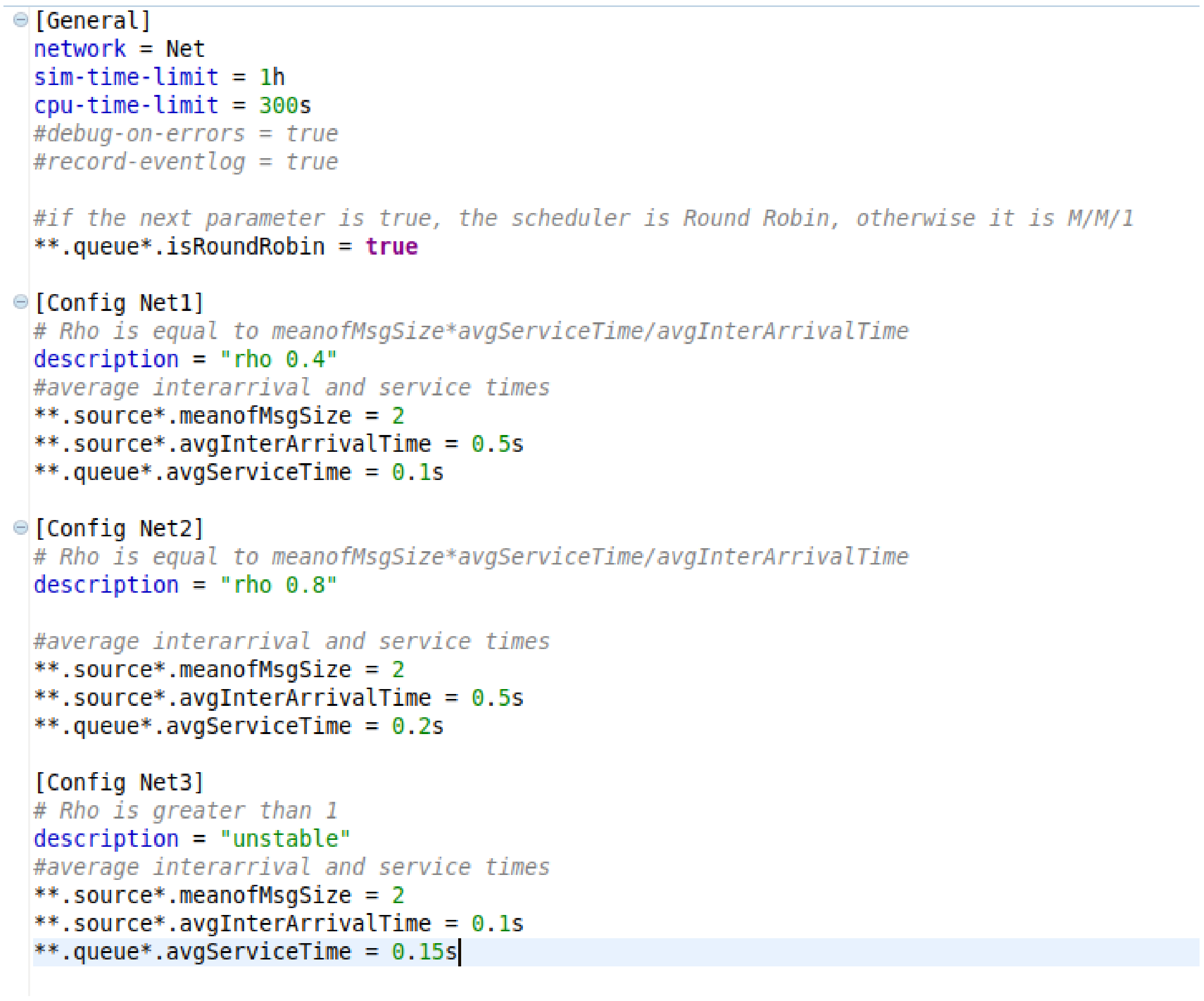
E[S] is average service time

$$E[S] = \frac{mean\ of\ message\ time}{average\ service\ rate} = \frac{meanofMsgSize}{\mu}$$

Then

$$\rho = \lambda \cdot \frac{meanofMsgsize}{\mu} = meanofMsgsize \cdot \frac{average\ service\ time}{average\ arrival\ time}$$

In this omnetpp.ini file, 3 configurations were made. One for rho equals to 0.4, one for rho equals to 0.8, and one for rho is greater than 1, which is unstable



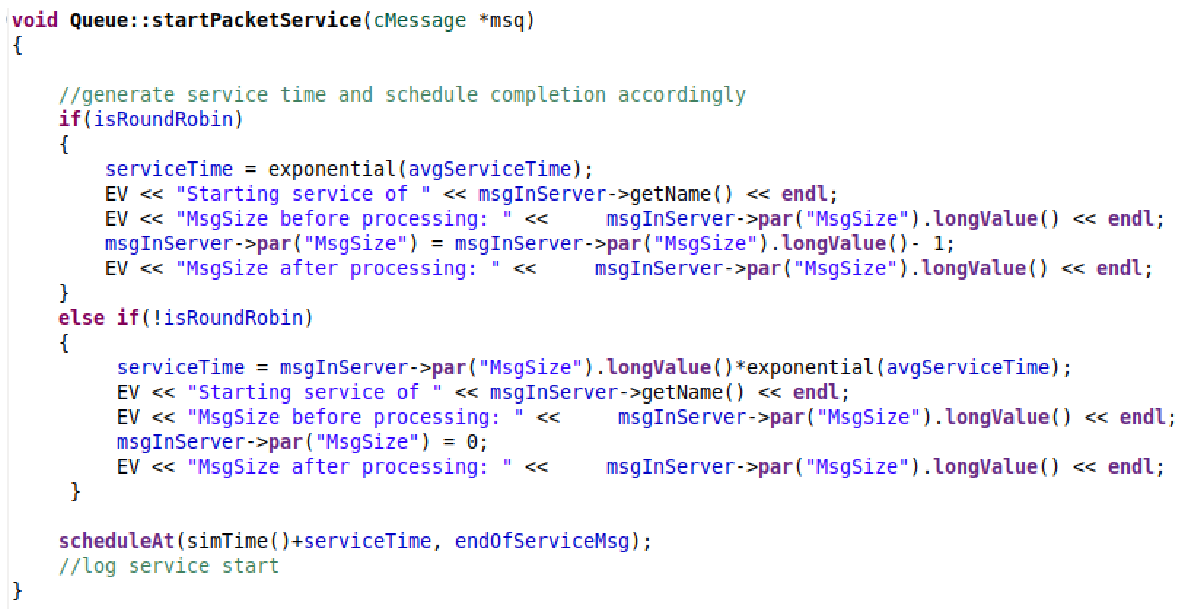
```
[General]
network = Net
sim-time-limit = 1h
cpu-time-limit = 300s
#debug-on-errors = true
#record-eventlog = true

#if the next parameter is true, the scheduler is Round Robin, otherwise it is M/M/1
**.queue*.isRoundRobin = true

[Config Net1]
# Rho is equal to meanofMsgSize*avgServiceTime/avgInterArrivalTime
description = "rho 0.4"
#average interarrival and service times
**.source*.meanofMsgSize = 2
**.source*.avgInterArrivalTime = 0.5s
**.queue*.avgServiceTime = 0.1s

[Config Net2]
# Rho is equal to meanofMsgSize*avgServiceTime/avgInterArrivalTime
description = "rho 0.8"

#average interarrival and service times
**.source*.meanofMsgSize = 2
**.source*.avgInterArrivalTime = 0.5s
**.queue*.avgServiceTime = 0.2s

[Config Net3]
# Rho is greater than 1
description = "unstable"
#average interarrival and service times
**.source*.meanofMsgSize = 2
**.source*.avgInterArrivalTime = 0.1s
**.queue*.avgServiceTime = 0.15s
```

Figure 2: The omnetpp.ini file

1.3 Handling the Messages

How the messages are handled is explained in this section. The messages treated according to it is a Round-Robin based system or M/M/1. The boolean parameter "isRoundRobin", assigned in omnetpp.ini, indicates the type of the system.

In the startPacketService function, the service time is assigned as $e^{avgServiceTime}$. The function decreases MsgSize by 1 if it is Round-Robin system, otherwise, it decreases MsgSize to 0 while assigning service time as MsgSize * $e^{avgServiceTime}$. After that, to simulate serving processes, it sends a self message after serviceTime.



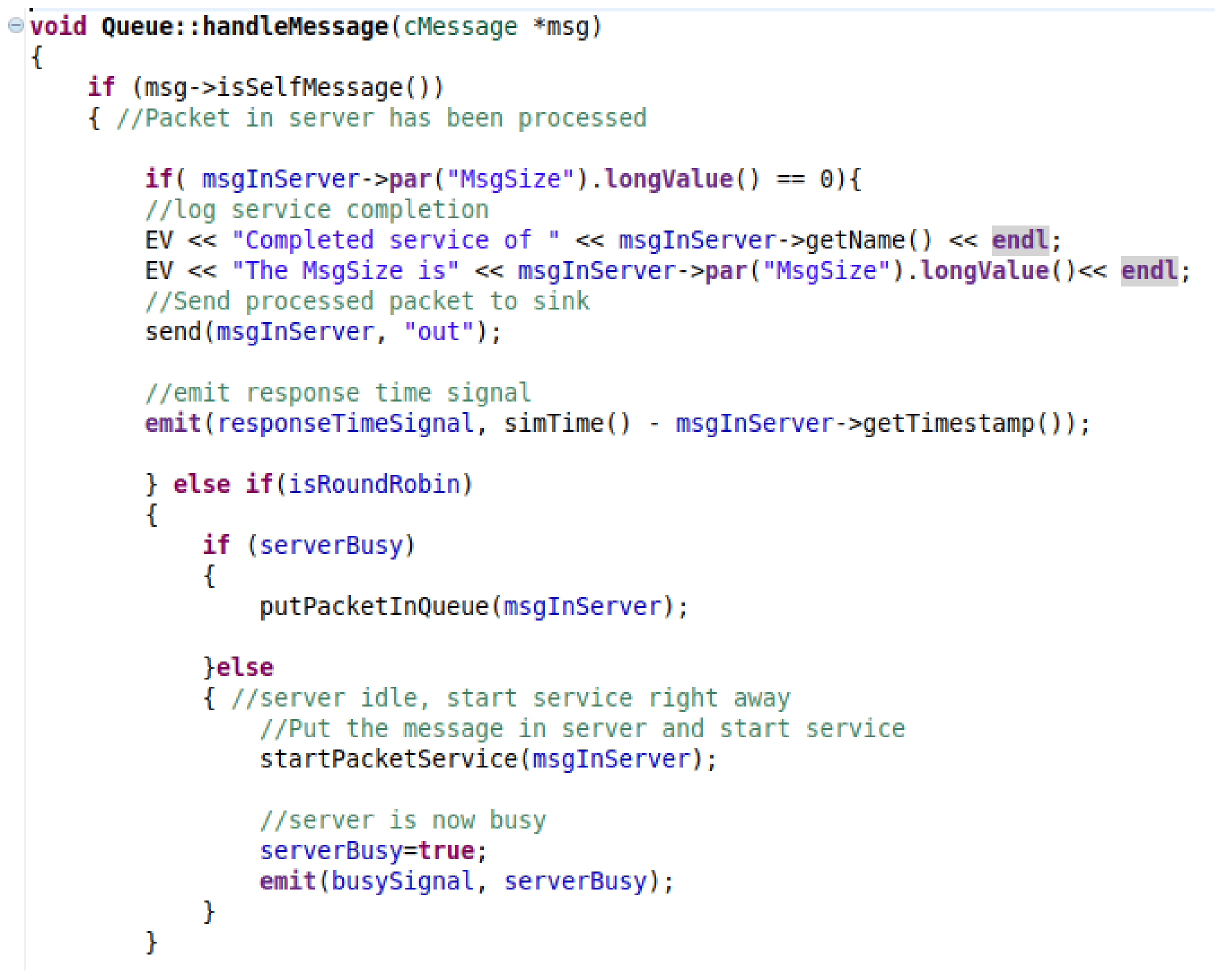
```
void Queue::startPacketService(cMessage *msq)
{

    //generate service time and schedule completion accordingly
    if(isRoundRobin)
    {
        serviceTime = exponential(avgServiceTime);
        EV << "Starting service of " << msgInServer->getName() << endl;
        EV << "MsgSize before processing: " <<     msgInServer->par("MsgSize").longValue() << endl;
        msgInServer->par("MsgSize") = msgInServer->par("MsgSize").longValue()- 1;
        EV << "MsgSize after processing: " <<     msgInServer->par("MsgSize").longValue() << endl;
    }
    else if(!isRoundRobin)
    {
         serviceTime = msgInServer->par("MsgSize").longValue()*exponential(avgServiceTime);
         EV << "Starting service of " << msgInServer->getName() << endl;
         EV << "MsgSize before processing: " <<     msgInServer->par("MsgSize").longValue() << endl;
         msgInServer->par("MsgSize") = 0;
         EV << "MsgSize after processing: " <<     msgInServer->par("MsgSize").longValue() << endl;
     }

    scheduleAt(simTime()+serviceTime, endOfServiceMsg);
    //log service start
}
```

Figure 3: Processing a packet

1.4 Receiving Self-Message

When a self-message arrived to Queue node, it means that a message is processed. Then it checks the MsgSize. If the MsgSize is equal to 0, it means that the message has been processed completely. It sends the message out and stamps the response time.

If the MsgSize not equal to 0, and it is a Round-Robin System, the message is put back in the queue, otherwise, it processes the message again.



```
void Queue::handleMessage(cMessage *msg)
{
    if (msg->isSelfMessage())
    { //Packet in server has been processed

        if( msgInServer->par("MsgSize").longValue() == 0){
        //log service completion
        EV << "Completed service of " << msgInServer->getName() << endl;
        EV << "The MsgSize is" << msgInServer->par("MsgSize").longValue()<< endl;
        //Send processed packet to sink
        send(msgInServer, "out");

        //emit response time signal
        emit(responseTimeSignal, simTime() - msgInServer->getTimestamp());

        } else if(isRoundRobin)
        {
            if (serverBusy)
            {
                putPacketInQueue(msgInServer);

            }else
            { //server idle, start service right away
                //Put the message in server and start service
                startPacketService(msgInServer);

                //server is now busy
                serverBusy=true;
                emit(busySignal, serverBusy);
            }
        }
```

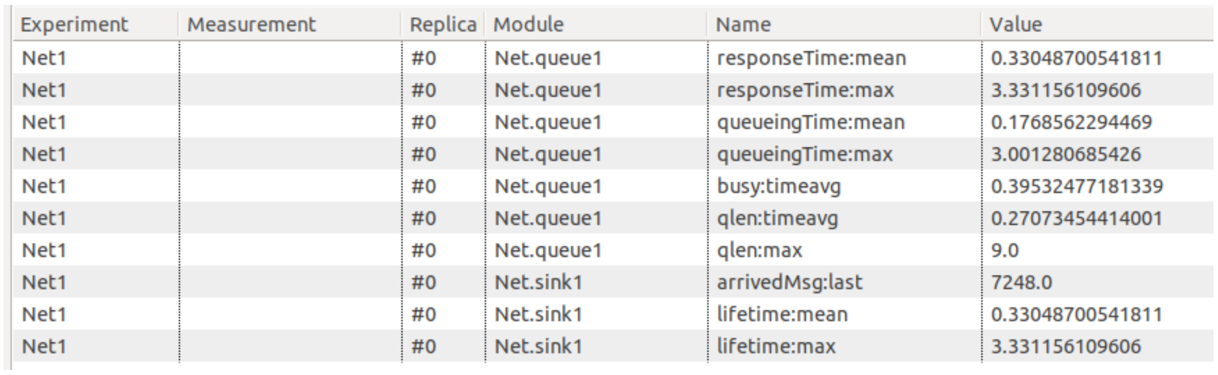
Figure 4: Handling self-messages

In the other cases, the system treats packets same with SourceQueueSink system.

2. Results and Conclusion

2.1 Systems with $\rho = 0.4$

When we runned with configuration 1 as Round-Robin System, we get these results:



Experiment	Measurement	Replica	Module	Name	Value
Net1		#0	Net.queue1	responseTime:mean	0.33048700541811
Net1		#0	Net.queue1	responseTime:max	3.331156109606
Net1		#0	Net.queue1	queueingTime:mean	0.1768562294469
Net1		#0	Net.queue1	queueingTime:max	3.001280685426
Net1		#0	Net.queue1	busy:timeavg	0.39532477181339
Net1		#0	Net.queue1	qlen:timeavg	0.27073454414001
Net1		#0	Net.queue1	qlen:max	9.0
Net1		#0	Net.sink1	arrivedMsg:last	7248.0
Net1		#0	Net.sink1	lifetime:mean	0.33048700541811
Net1		#0	Net.sink1	lifetime:max	3.331156109606

Figure 5: Results of Round-Robin System with $\rho =$ 0.4

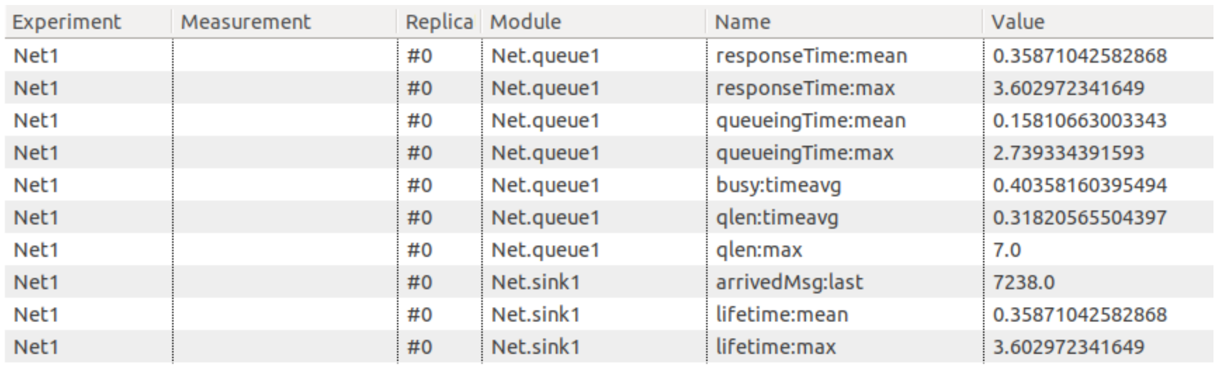
"busy:timeavg" gives us the utilization factor, it is very close to 0.4, which was designed.

"responseTime:mean" states that the average generic response time, which is 330 ms.

"queueingTime:mean" shows the conditional average queueing time, which is 177 ms.

By taking the difference of response time and queueing time, we can obtain the service time. The average service time is 153 ms.

When we runned with configuration 1 as M/M/1 System, we get these results:



Experiment	Measurement	Replica	Module	Name	Value
Net1		#0	Net.queue1	responseTime:mean	0.35871042582868
Net1		#0	Net.queue1	responseTime:max	3.602972341649
Net1		#0	Net.queue1	queueingTime:mean	0.15810663003343
Net1		#0	Net.queue1	queueingTime:max	2.739334391593
Net1		#0	Net.queue1	busy:timeavg	0.40358160395494
Net1		#0	Net.queue1	qlen:timeavg	0.31820565504397
Net1		#0	Net.queue1	qlen:max	7.0
Net1		#0	Net.sink1	arrivedMsg:last	7238.0
Net1		#0	Net.sink1	lifetime:mean	0.35871042582868
Net1		#0	Net.sink1	lifetime:max	3.602972341649

Figure 6: Results of M/M/1 System with $\rho =$ 0.4

"busy:timeavg" gives us the utilization factor, it is very close to 0.4, which was designed.

"responseTime:mean" states that the average generic response time, which is 358 ms. It is sligthly higher then Round-Robin system.

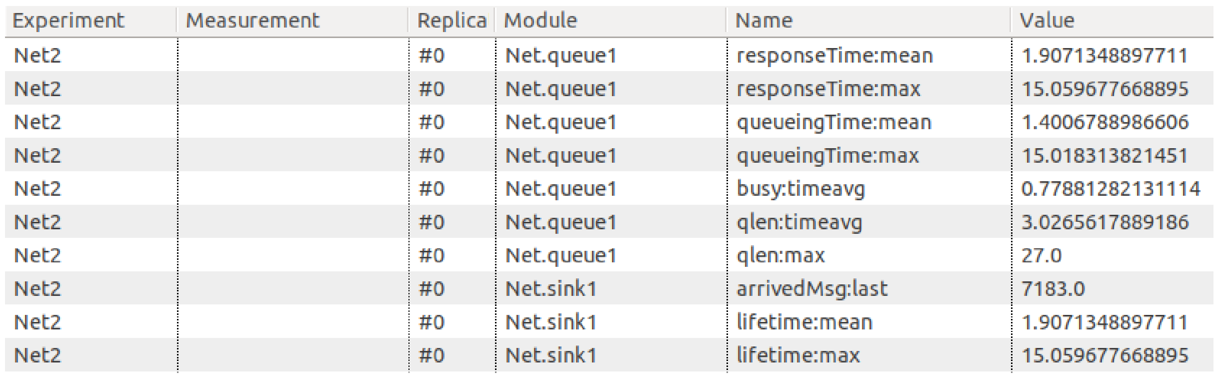
"queueingTime:mean" shows the conditional average queueing time, which is 158 ms. It is slightly lower than Round-Robin scheduler, although theorotically they need to be the same. The reason could be the difference between the number of the processed packets. While 7248 messages arrived in the Round-Robin system, 7238 messages arrive in M/M/1 system because in the Round-Robin scheduler the packages get fair services. In M/M/1 the last 10 packets were still waiting although they need less message sizes.

By taking the difference of response time and queueing time, we can obtain the service time. The average service time is 200 ms. We can conclude that the average service time is increased. This is due to 10 packets that has not processed which was sent last.

From this observation we can conclude that, the fair sharing of the service time decreased the average response time. It leads to processing more messages in the same time.

2.2 Systems with $\rho = 0.8$

In this section, the results with $\rho = 0.8$ is discussed.



Experiment	Measurement	Replica	Module	Name	Value
Net2		#0	Net.queue1	responseTime:mean	1.9071348897711
Net2		#0	Net.queue1	responseTime:max	15.059677668895
Net2		#0	Net.queue1	queueingTime:mean	1.4006788986606
Net2		#0	Net.queue1	queueingTime:max	15.018313821451
Net2		#0	Net.queue1	busy:timeavg	0.77881282131114
Net2		#0	Net.queue1	qlen:timeavg	3.0265617889186
Net2		#0	Net.queue1	qlen:max	27.0
Net2		#0	Net.sink1	arrivedMsg:last	7183.0
Net2		#0	Net.sink1	lifetime:mean	1.9071348897711
Net2		#0	Net.sink1	lifetime:max	15.059677668895

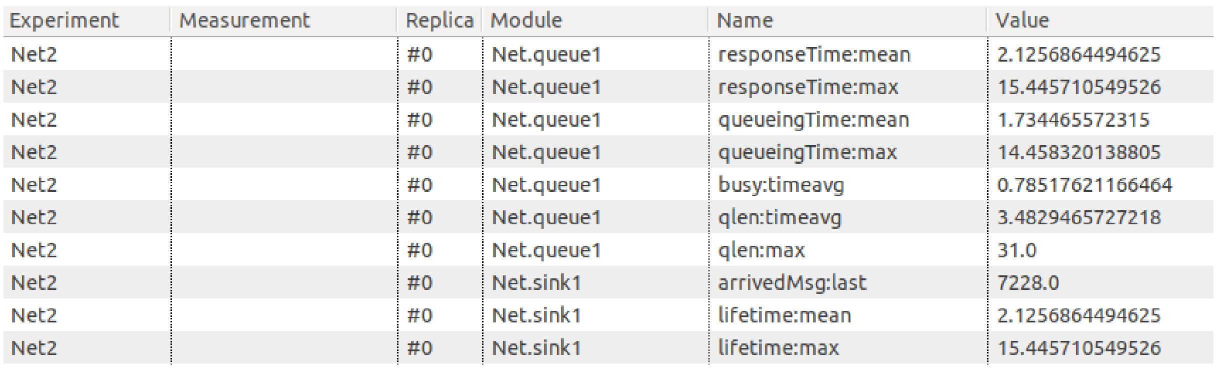
Figure 7: Results of Round-Robin System with $\rho = 0.8$

"busy:timeavg" gives us the utilization factor, it is very close to 0.8, which was designed.

"responseTime:mean" states that the average generic response time, which is 1.9 s.

“queueingTime:mean” shows the conditional average queueing time, which is 1.4 s.

By taking the difference of response time and queueing time, we can obtain the service time. The average service time is 507 ms.



Experiment	Measurement	Replica	Module	Name	Value
Net2		#0	Net.queue1	responseTime:mean	2.1256864494625
Net2		#0	Net.queue1	responseTime:max	15.445710549526
Net2		#0	Net.queue1	queueingTime:mean	1.734465572315
Net2		#0	Net.queue1	queueingTime:max	14.458320138805
Net2		#0	Net.queue1	busy:timeavg	0.78517621166464
Net2		#0	Net.queue1	qlen:timeavg	3.4829465727218
Net2		#0	Net.queue1	qlen:max	31.0
Net2		#0	Net.sink1	arrivedMsg:last	7228.0
Net2		#0	Net.sink1	lifetime:mean	2.1256864494625
Net2		#0	Net.sink1	lifetime:max	15.445710549526

Figure 8: Results of M/M/1 System with $\rho = 0.8$

“busy:timeavg” gives us the utilization factor, it is very close to 0.8, which was designed.

“responseTime:mean” states that the average generic response time, which is 2.12 s. It is higher then Round-Robin system.

“queueingTime:mean” shows the conditional average queueing time, which is 1.73 s. It is also higher then Round-Robin system, which should be the same. The reason is also the number of processed messages. unlike previous systems with $\rho = 0.4$, the M/M/1 system processed more messages.

By taking the difference of response time and queueing time, we can obtain the service time. The average service time is 391 ms.

If we consider, user with nQ service enters n times, at each entrance he finds approximatively E[N] users in front of him, each taking a Q seconds service

The conditional response time can be approximated by:

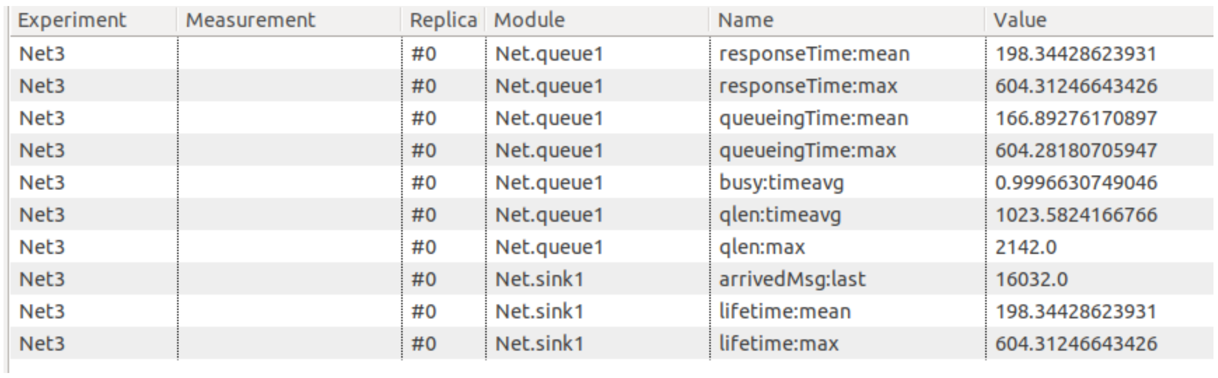
$$W_n^{RR,approx} = nQE[N] + nQ$$

$$W_n^{M/M/1,approx} = \frac{1}{1-\sigma} QE[N] + nQ$$

It means if nQ < $\frac{1}{1-\sigma} = E[S]$, user gets a faster service with Round-Robin than M/M/1.

In the case of $\rho = 0.8$ Round Robin provided faster service than M/M/1 while M/M/1 were providing faster service than Round Robin with $\rho = 0.4$

2.3 Unstable Systems with $\rho > 1$



Experiment	Measurement	Replica	Module	Name	Value
Net3		#0	Net.queue1	responseTime:mean	198.34428623931
Net3		#0	Net.queue1	responseTime:max	604.31246643426
Net3		#0	Net.queue1	queueingTime:mean	166.89276170897
Net3		#0	Net.queue1	queueingTime:max	604.28180705947
Net3		#0	Net.queue1	busy:timeavg	0.9996630749046
Net3		#0	Net.queue1	qlen:timeavg	1023.5824166766
Net3		#0	Net.queue1	qlen:max	2142.0
Net3		#0	Net.sink1	arrivedMsg:last	16032.0
Net3		#0	Net.sink1	lifetime:mean	198.34428623931
Net3		#0	Net.sink1	lifetime:max	604.31246643426

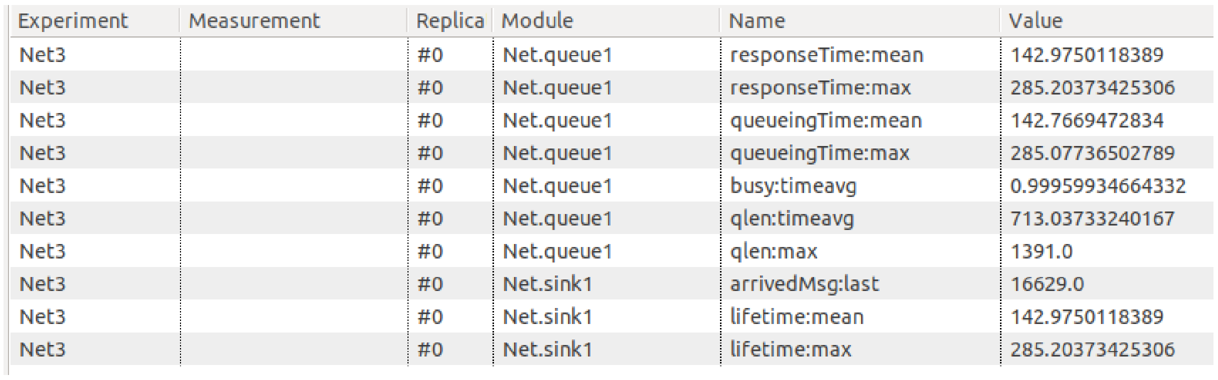
Figure 9: Results of Round-Robin System with $\rho = 1.1$

“busy:timeavg” gives us the utilization factor, it is almost 1 because the server worked in the all availabe time.

“responseTime:mean” states that the average generic response time, which is 198 s.

“queueingTime:mean” shows the conditional average queueing time, which is 167 s.

By taking the difference of response time and queueing time, we can obtain the service time. The average service time is 31 s.



Experiment	Measurement	Replica	Module	Name	Value
Net3		#0	Net.queue1	responseTime:mean	142.9750118389
Net3		#0	Net.queue1	responseTime:max	285.20373425306
Net3		#0	Net.queue1	queueingTime:mean	142.7669472834
Net3		#0	Net.queue1	queueingTime:max	285.07736502789
Net3		#0	Net.queue1	busy:timeavg	0.99959934664332
Net3		#0	Net.queue1	qlen:timeavg	713.03733240167
Net3		#0	Net.queue1	qlen:max	1391.0
Net3		#0	Net.sink1	arrivedMsg:last	16629.0
Net3		#0	Net.sink1	lifetime:mean	142.9750118389
Net3		#0	Net.sink1	lifetime:max	285.20373425306

Figure 10: Results of M/M/1 System with $\rho = 1.1$

“busy:timeavg” gives us the utilization factor, it is almost 1 because the server worked in the all availabe time.

“responseTime:mean” states that the average generic response time, which is 142.97 s.

“queueingTime:mean” shows the conditional average queueing time, which is 142.76 s.

By taking the difference of response time and queueing time, we can obtain the service time. The average service time is 209 ms.

When $\rho > 1$, we can observe that the M/M/1 performed better than Round Robin in the all cases.

3. References

In this work, the materials and slides that were provided by Prof. Filippini were used.