POLITECNICO DI TORINO

Master's Degree in Communications and Computer Networks Engineering

Network Simulation Laboratory

Supervisors

Prof. Paolo GIACCONE

Candidate

Razieh HEIDARIAN

01 2020

Table of Contents

Chapter 1

First Laboratory

1.1 $\mu/\mu/1$ QUEUE

Aim of this laboratory is to code completely the model of an M/M/1, to learn how to collect simulation samples and to analyze them at the end of the simulation. For implement this project we have to follow these steps:
• Step 1. Define the ned file to describe the composition of modules to build the network and their parameters.
• Step 2. Define the three cc files to code each simple module
• Step 3. Define the ini file.
The M/M/1 model:
In queueing theory, a discipline within the mathematical theory of probability, an M/M/1 queue represents the queue length in a system having a single server, where arrivals are determined by a Poisson process and job service times have an exponential distribution. The model name is written in Kendall's notation.
• Arrivals occur at rate λ according to a Poisson process and move the process from state i to i+1.
• Service times have an exponential distribution with rate parameter μ in the $\mu/\mu/1$ queue, where $1/\mu$ is the mean service time.
• A single server serves customers one at a time from the front of the queue, according to a first- come, first-served discipline. When the service is complete the customer leaves the queue and the number of customers in the system reduces by one.
• The buffer is of infinite size, so there is no limit on the number of customers it can contain. For NED file we design the composition of the network model in terms of simple modules. For simple modules in C++ we need three C++ files as: Generator, Queue, Sink that they work as follows:

1.1.1 Generator:

The task of this module is to create a self-Message in the initialization of simulation in exponentially time and also create the message as a packet to send to the queue module with departure time that is from when packet generated until the simtime. moreover, schedule the new packet at exponentially time .

1.1.2 Queue:

Now we have to check the queue that ; message is end service event, then dequeue and send the packet to the output and if another packet is available in the buffer and server idling then enqueue the packet to service and start a new service.

1.1.3 Sink:

The job of this module is to create the files and calculate all the statistics from the files.Here there is the complex network M/M/1:
1. Design and develop the simulator.

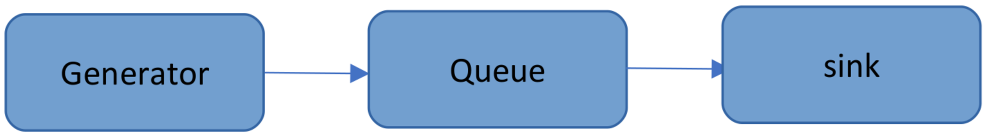


Figure 1.1: The main elements

2. Run the simulation with GUI. What is the required command line?
Mqueue.exe
3. Run the simulation without GUI. What is the required command line?
Mqueue -u Cmdenv
4. Analyze the simulation results collected under the directory results/. What are the values of all the statistics collected during the simulation? What is the duration of the simulation?
By running the command scavetool we can see all the simulation result that is collected by statistics. Therefore, as the arriving time is exponentially and the size of queue is infinity we can see huge delay for packets because of increasing slightly arrival time and so the packets are serviced by high delay. Moreover, we can see the minimum and maximum delay .
5. If the parameters in the ned were not declared as volatile, what would be the effect on the model?
The volatile modifier causes the parameter's value expression to be evaluated

every time the parameter is read. This has significance if the expression is not constant; for example, it involves numbers drawn from a random number generator. In contrast, non-volatile parameters are evaluated only once. (This practically means that they are evaluated and replaced with the resulting constant at the start of the simulation). Because of the volatile modifier, the queue module's C++ implementation is expected to re-read the service Time parameter whenever a value is needed; that is, for every job serviced. To highlight this effect, here's how one can have a time-varying parameter by exploiting the simTime() NED function that returns the current simulation time. In practice, a volatile parameter is typically used as a configurable source of random numbers for modules..

NOTE

This does not mean that a non-volatile parameter could not be assigned a random value. It can, but that would have a totally different effect: the simulation would use a constant service time, chosen randomly at the beginning of the simulation.

1.1.4 Comparison of theoretical and simulation result

This laboratory shows how to model a single-queue,single-server system with a single traffic source and an infinite storage capacity. In the notation, the μ stands for Markovian; $\mu/\mu/1$ means that the system has a Poisson arrival process, an exponential service time distribution, and one server. Queuing theory provides exact theoretical results for some performance measures of an M/M/1 queuing system, and this model makes it easy to compare empirical results with the corresponding theoretical results(to implement this system, we require the state-stability, and so we have to run several time the simulation). In the $\mu/\mu/1$ system, λ is the rate at which packets arrive at the queue, and μ is the rate at which the server will serve packets. The ratio λ/μ is called utilization ρ; if this ratio is greater than 1, that says customers are arriving faster than they can be served, and so the line will grow without bound. My experiments are for the ratio is less than one that follows this formula: ρ= λ/μ<1 (that means the system is stable). Thus I start the experiment from $\rho = 0.1$ and increase it by step 0.1; the average delay and theoretical delay increase slightly until $\rho = 0.9$, and after that, I see a sharply increase because the queue was filled much faster and so it will grow without limitation as well as packets are served with high delay. At this point, our queue is infinity, and so each arriving packet will either wait at the back of the line to be serviced. Furthermore, each arriving packet that wants to be serviced sees a full queue (or almost full queue) on arrival has to wait for N seconds to be served, where N is the time it takes to serve a full queue. The larger the queue size is, the longer each packet that is not dropped will be delayed.

As we see in the Figure "Waiting Time[s]: Theoretical" and "Waiting Time[s]: Simulation" showing the theoretical and empirical values of the waiting time in the queue, on one ax and “Server Utilization[s]: normalized load” which shows the utilization of the single server over the course of the simulation, in this plot you can see how the empirical values evolve during the simulation and compare them with the theoretical value.

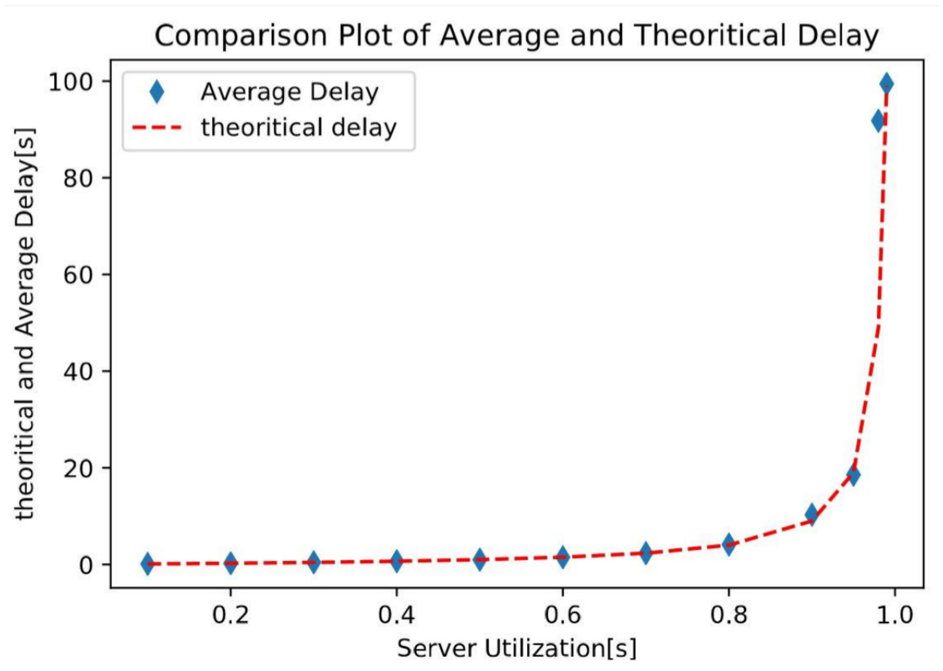


Figure 1.2: The graph of average and theoretical delay

Chapter 2

Second Laboratory

The purpose of this laboratory is to evaluate the performance of a data transfer over the network, observing the impact of different data rates and offer loads.

2.1 Transmission on a communication channel :

To get the measurement, I design the Ethernet with one-directional in omnet++. I use four simple modules connected by input and output gates; two hosts are connected throughout an Ethernet switch, as shown in Figure-1. Packets are sent from H1 to the switch, which forwards the traffic to H2 with specific data rates and inter-arrival times. Assume :

1. Packets are generated in H1 according to a Poisson process;
2. The packet size is fixed and equal to 125 bytes;
3. The packet size is fixed and equal to 125 bytes;
4. The queue length is fixed and equal to 1000;
5. The transmission rate of H1 interface (txRateH1) is 100 Mbps;
6. The transmission rate of H2 interface (txRateH2) is 100Mbps ,10 Mbps ,1Gbps;
7. The propagation delay of both channels is 1 microsecond;
8. The performance metrics in function of the offeredLoad= 4, 8, 40, 80 Mbps ;
9. The simulation time is fixed 10s.

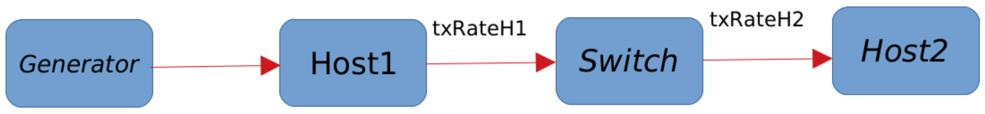


Figure 2.1: The main elements

2.1.1 Performance of Ethernet when the rate between switch and Host2 is 10Mbps :

I simulate for 10 seconds for each of the scenarios. We check the performances(packet loss and throughput), while the first link is 100Mbps and the second link is 10Mbps, as we expect there is a bottleneck in the second link. I observe that in the lower rate, we have fewer loss packets and a minor average delay of almost 0.0000145 [microsecond] that is quite acceptable since we increase the rate it sends much fewer packets per seconds as well as increment in average delay approximately 0.1 [s] the due to the bottleneck and limit switch's buffer size a lot of packets suffer from delay.

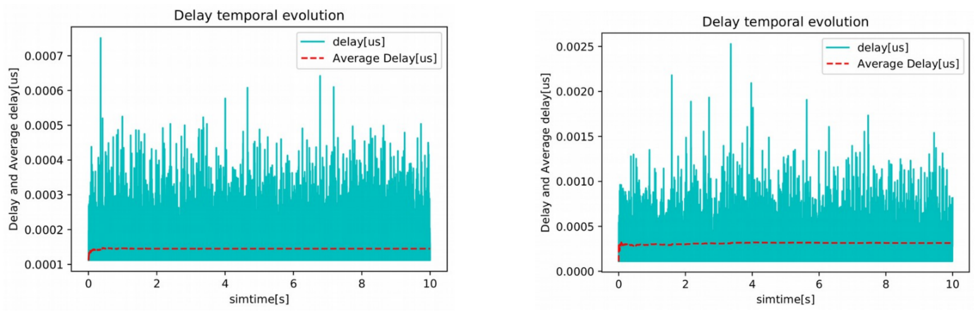


Figure 2.2: The graph of total delay and average delay

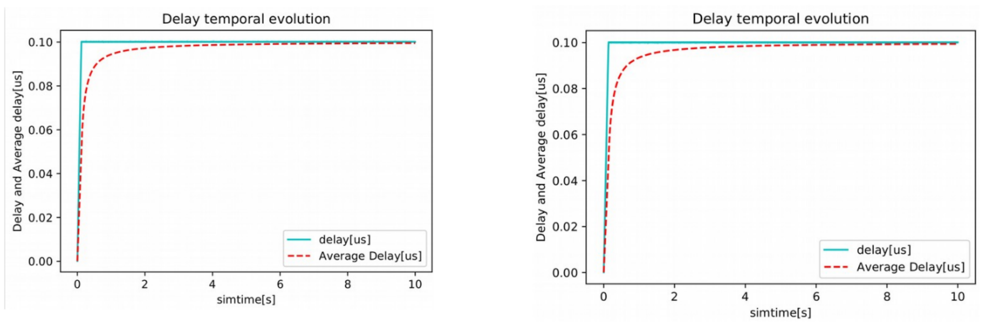


Figure 2.3: The graph of total delay and average delay

Furthermore, as you see in I measure the offered loads[Mbps] by the source that generator creates packets according to Poisson process and it grow up linearly, but as we have a bottleneck in the second link, so the throughput by Host2 can not achieve pick rate, and it will remain steady at 10Mbps.

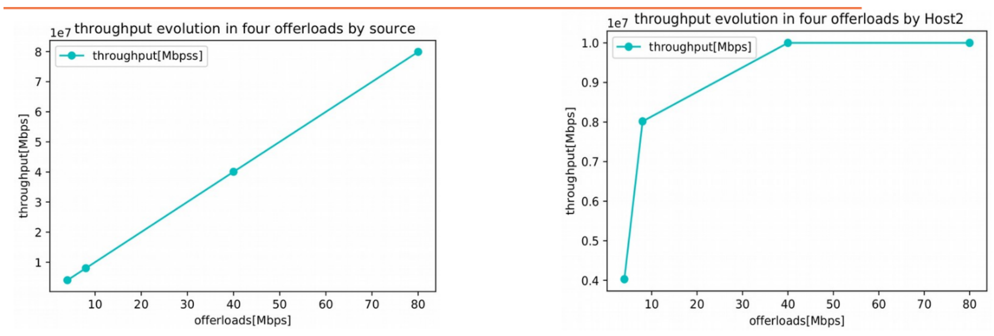


Figure 2.4: The graph of throughput in host2

Besides, as I mentioned above, in rate four and rate eight, we have acceptable throughput and respectable average inter-arrival time, but in rate 40 and 80, you can observe throughput can not reach the pick the average inter-arrival time is decreasing.

(Figure 5).

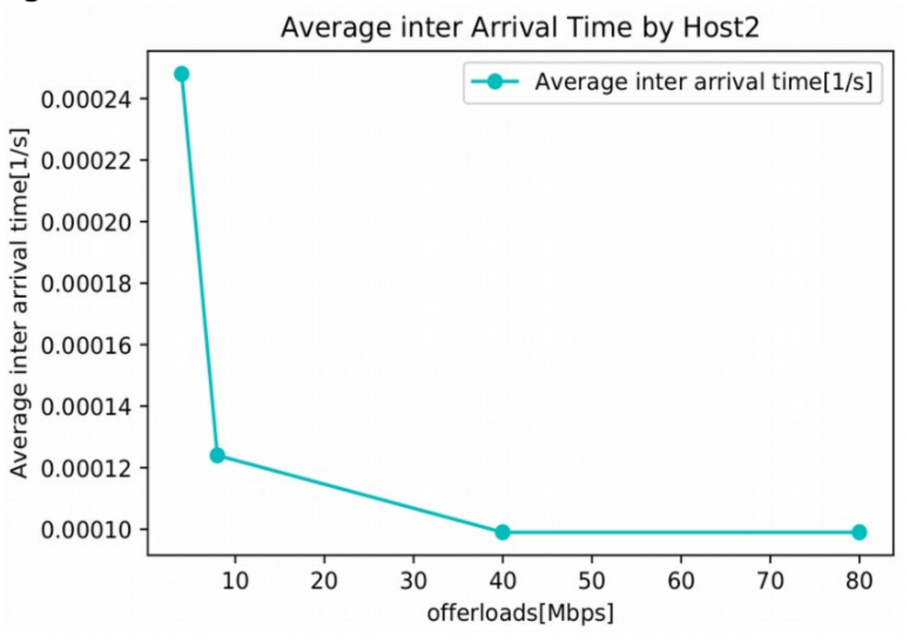


Figure 2.5: The graph of average inter arrival time by host2

2.1.2 Performance of Ethernet when the rate between switch and Host2 is 100Mbps :

In this scenario, we measure the behavior of the network while both of the links are 100Mbps.In this experiment, it is possible to know what happens, as we expected due to equality, the channel is reliable, and so we do not observe dropping packets, and they arrive completely, however as I observe because of speed in the channel, by increasing the rate we have a delay in packets, but I could say it is negligible. Moreover, the measured offer loads in this part are the same as the previous scenario, and the generator generates packets due to the Poisson process, and I measure throughout for each offer load as you see it raises.

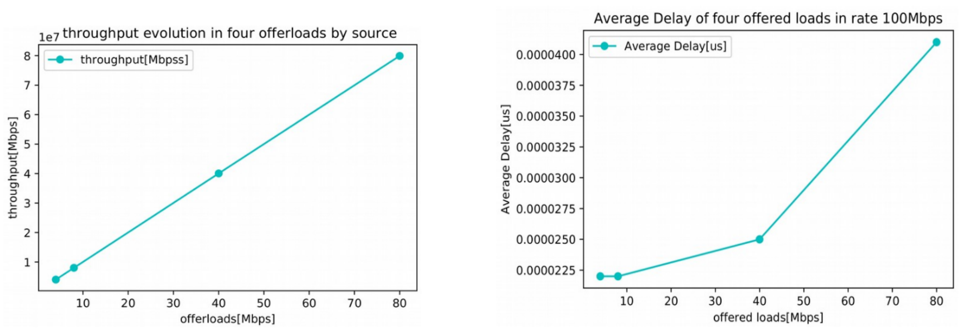


Figure 2.6: The graph of total delay with rate 100Mb/s

Additionally, as you see, I measure the throughput for each offer loads by Host2 to find out in each offer loads how much throughput we have. Because of the equality and speed of the connections, we can get the acceptable amount of packets, and throughput almost is as the number of offer loads, and it increases linearly. Besides, the average inter-arrival time in high rates is much less between packets than in the last scenario.

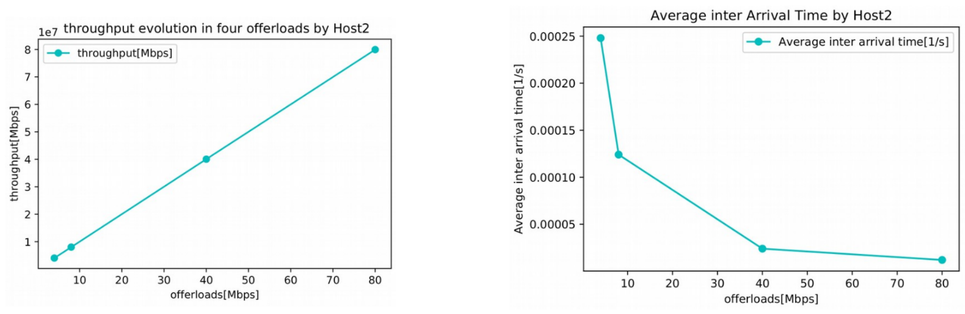
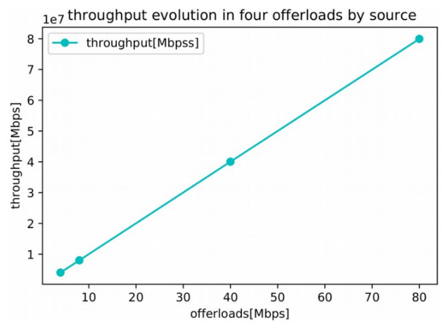


Figure 2.7: The graph of average inter arrival time by host2

2.1.3 Performance of Ethernet when the rate between switch and Host2 is 1Gbps :

In this experiment, I measure the performance over the link with 100Mbps that forward packets to the switch and the Host2 with a data rate 1Gbps. As you see in the previous scenario, we can observe no dropping packets. Also, in comparison with the method earlier, we detect a minor average delay in each packet. However, the source's measured offer load is the same as the other scenarios because generating packets is according to the Poisson process. As you see in it is increasing linearly.



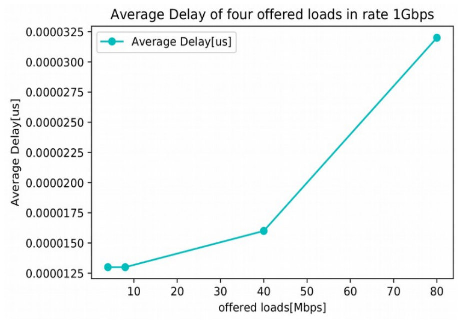
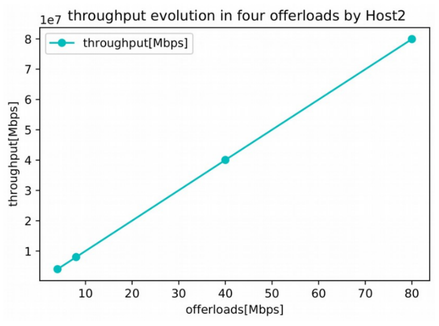


Figure 2.8: The graph of total delay with rate 1Gb/s

Also, the throughput of this scenario is the same as the last one but more reliable, and we can say by the confidence that we can get real throughput due to the high speed of the second link and also less delay for each packet the amount of throughput is close to offering loads. Moreover, each rate's inter-arrival time becomes much decreasing, and at high rates, more packets arrive at Host2.



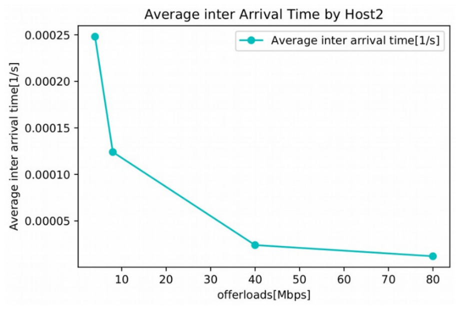


Figure 2.9: The graph of average inter arrival time by host2

Chapter 3

Third Laboratory

Scheduling policy in a packet-switching communication network manages queuing discipline, managing the order of serving traffic in the transmit and receiving the network interface queues. Accordingly, in this laboratory, we implement the two well-known scheduling policies Round Robin and Strict Priority. The main goal is to compare the two scheduling policies to evaluate the performance of two different types of traffic, variable Bit Rate (VBR) and Constant Bit Rate (CBR). VBR traffic is composed of packets of different sizes, indeed transporting variable bit rate traffic. On the other hand, the CBR is a traffic category used for connections that transport traffic constantly.

3.1 Implementation and design:

In this laboratory we have designed network topology with two Hosts and one switch, and a sink shown in the figure 1. First Host H1 generates CBR traffic, and the second Host H2 generates VBR traffic. We considered all elements simple modules. Therefore, we have 6 simple modules VBR Generator, CBR, two ueues called CBR queue and VBR queue, a switch and a sink. Figure 1 depicts the network topology.

3.1.1 Input Parameters and Connections:

1. CBR packet size =125;
2. VBR packet size = uniform (64,1518);
3. CBR interGeneration time as constant;
4. VBR interGeneration time as volatile, exponential;

5. Queue size, same for all queues;

Each generator has an output port, while queues have one input and Figure 1 Network Topology one output, also the switch has 2 input ports and one output port, and as usual sink has just one input port. For CBR traffic we have fixed packet size, however, since VBR traffic has variable packet size, to calculate InterGenerationTime we considered the average packet size, 791 Bytes.

3.1.2 Exchanged Messages:

In this experiment we used 2 types of messages, cMessage and My-Packet, which is extended of cPackets. My-packet type contains additional integer variable called Flow that My-Packet extends cPacket is used to differentiate two types of traffics, which required to measure statistics independently. Class of My-Packet.msg described as the following and after first compile the My-Packet an int Flow; My-Packet created and automatically linking to simulation.
CBR traffic: Flow Id = 1;
VBR traffic: Flow Id = 2;

3.1.3 Main Data Structures:

In each basic sub-module, we define data structures to be used:

Generator:

o cMessage *sendOfTx that is used as the self-message.
o long counter that is incremented whenever a packet is being sent out.

Buffer:

o int size to check whether there is available space to store the packet in the buffer.
o cStdDev and cOutVector to collect and record the lost packets and to count number of packets that are being sent out.
o cMessage TxEnd to check whether it is a self-message or not. packets that are being sent out.

Sink:

packets that are being sent out. o cStdDev and cOutVector to collect and record the throughput, the coefficient variance, and the average delay.
o long BitsRx to count the total received bits.

3.2 Results:

Simulation is done for 20 s. and the default seed is considered as seed = 65503. Moreover, simulation is repeated for 5 different seeds equal to = 5, 2000, 30000, 65503, 88888, the seeds chosen by chance but with high gap among them to avoid overlap.

3.2.1 Round Robin Evaluation:

The Throughput of the Round Robin in the function of offered load. Considering the saturation in the graph, we have 50 percent of load assigning to each traffic. The throughput of CBR traffic will decrease by the ratio of the average packet size corresponding to each flow. Indeed, packets with a larger size will be served more. After saturation, the increasing loss leads to the delay of packets represented. Besides, the delay of CBR traffic has a significant increase concerning the VBR traffic, and this is because of more packet loss. At low loads (before saturation zone), while the throughput of CBR is equal to VBR, the CBR traffic experience less delay concerning VBR, which can be justified because of packet size. In statistics, the coefficient of variation (CV) is a simple measure of relative delay dispersion. Corresponding delay coefficient variance depicted.

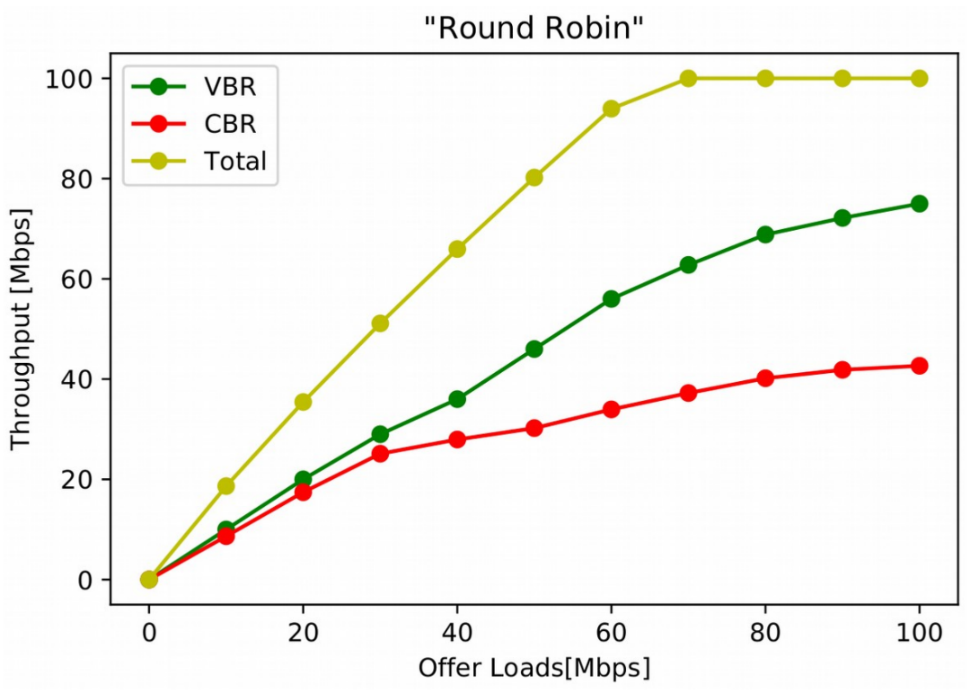


Figure 3.1: Round Robin Evaluation

3.2.2 Strict priority Evaluations:

In the strict priority scheduling, we considered CBR traffic with higher priority. Based on this consideration, when we are in the saturation zone, the CBR traffic must serve more frequently than the VBR traffic. This priority leads to more packet loss for VBR traffic after saturation. Therefore, the delay of corresponding traffic increases. This is almost the opposite in the case of the round-robin. For load equal to 100, the VBR traffic has no chance to get service; therefore, its delay goes to infinity (here it is considered as a very large number) however, in the round-robin at load 100 both traffic are served, while the CBR service decreases significantly.

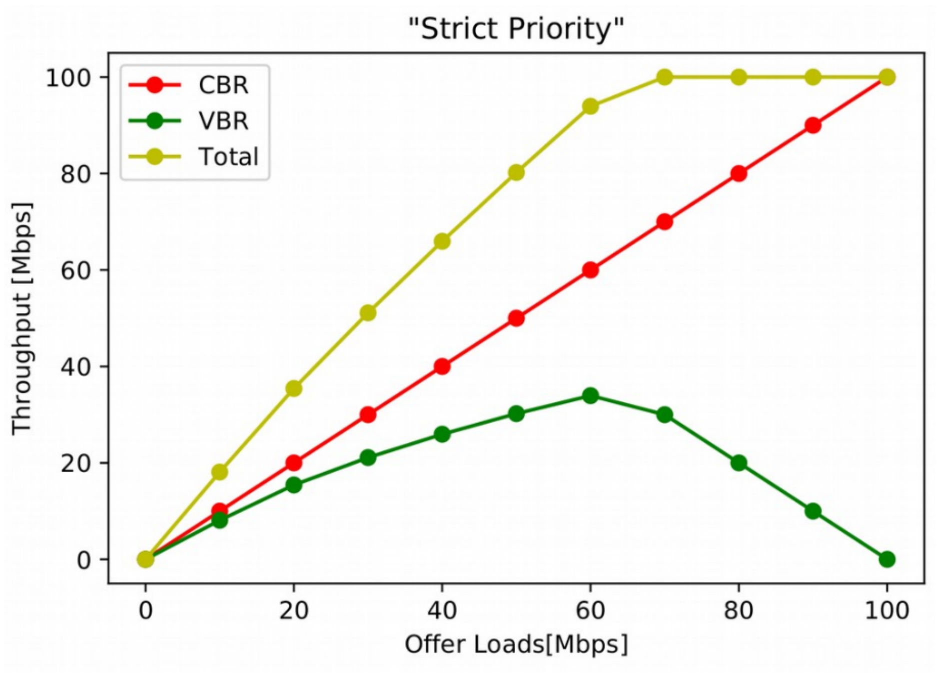


Figure 3.2: Strict priority Evaluation

3.3 warmup-period

The warmup-period option specifies the length of the initial warm-up period. When set, results belonging. The first x seconds of the simulation will not be recorded into output vectors and will not be counted into the calculation of output scalars. This option is helpful for steady-state simulations. The default is 0s (no warm-up period). Based on data recorded in file .vec, we calculated the cumulative sum of delay. Since load = 50 bps is saturation zone, we choose the warm-up period based on this load. Afterward, we can find out the transient is at the first second of simulation by inspection, after the system reaches to steady-state condition. Therefore, we fix the warm-up period =1 s.