International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

CRITICAL ANALYSIS FOR CONGESTION CONTROL IN COMPUTER NETWORKS

1. SHAIKH SHARIFUL HABIB, 2. DR. MD. NURUL MUSTAFA, 3. MOHAMMAD JAHANGIR ALAM

M.Sc Student, Computer Science & Information Technology, Southern University Bangladesh
Assistant Professor, CSE Department, International Islamic University Chittagong, Bangladesh[1]

Professor, Department of Computer Science and Engineering (CSE),
Southern University Bangladesh, Chittagong, Bangladesh.[2]

Assistant Professor, Department of Computer Science and Engineering (CSE),
Southern University Bangladesh, Chittagong, Bangladesh.3

e-mail:habib_iuc@yahoo.com[1], nurul.mustafa@southern.edu.bd[2] jahangir@southern.edu.bd[3]

ABSTRACT

Computer network is a very important issue. We can't imagine anything without computer network in this modern civilization. But in the network, congestion plays an important role in degrading the performance of the network. So it is important to detect and control the congestion to improve the performance of the computer network. Different factors are responsible for congestion. One of the main important factors is buffer overflow or managing the buffer to hold the packets by hop or router from the source. There may have different techniques in this respect. Burstiness is another cause of congestion, it is used to regulate average rate and burstiness of traffic. The emergency of high speed networks are expected to carry a broad range of traffic (video, audio and data). The variation (i.e., standard deviation) in the packet arrival times is very important to consider. In this paper we have tried to give different solutions by developing different models for different situations of congestion and also tried to analyze those models to give more better ideas for the researchers in IT field.

Keywords: Token Bucket, Traffic Jam, Average Length Queue, Jitter, Multicasting

1. INTRODUCTION:

When too many packets are present in (a part of) the subnet, performance degrades. This situation is called congestion. When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent. However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely and almost no packets are delivered. Slow processors can also cause congestion. If the routers' CPUs are slow at performing the bookkeeping tasks required of them (queuing buffers, updating tables, etc.), queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion. Upgrading the lines but not changing the processors, or vice versa, often helps a little, but frequently just shifts the bottleneck. Also, upgrading part, but not all, of the system, often just moves the bottleneck somewhere else. The real problem is frequently a mismatch between parts of the system. This problem will persist until all the components are in balance. Congestion control has to do with making sure the subnet is able to carry the offered traffic. It is a global issue, involving the behavior of all the hosts, all the routers, the store-and-forwarding processing with in the routers, and all the other factors that tend to diminish the carrying capacity of the subnet. Flow control, in contrast, relates to the point-to-point traffic between a given sender and a given receiver. Its job is to make sure that a fast sender cannot continually transmit data faster than the receiver is able to absorb it.

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

Flow control frequently involves some direct feedback from the receiver to the sender to tell the sender how things are doing at the other end.

2. GENERAL PRINCIPLES OF CONGESTION CONTROL:

Many problems in complex systems, such as computer networks, can be viewed from a control theory point of view. This approach leads to dividing all solutions into two groups: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made.

Tools for doing open-loop control include deciding when to accept new traffic, deciding when to discard packets and which ones, and making scheduling decisions at various points in the network. All of these have in common the fact that they make decisions without regard to the current state of the network.

In contrast, closed loop solutions are based on the concept of a feedback loop. This approach has three parts when applied to congestion control:

1. Monitor the system to detect when and where congestion occurs.
2. Pass this information to places where action can be taken.
3. Adjust system operation to correct the problem.

3. REVIEWED TOKEN BUCKET:

In [1] the author describes the improved step for token bucket algorithm. Server service time was controlled. But it is **quite difficult to control the server service time** by inputting some predefined value. Server service time may be arbitrary. In the open loop system we have to design a system in such a way that the total system can face the over burden of packets. We have taken different samples of data and found problems. So we think there is a solution to control the system in the following way and we applied it:

Service time of server may be arbitrary. We have to choose the N-server. **That is how many packets server can handle instead of one packet independently**. In this way the server will have enough capability to take a number of packets.

Here we have taken the simulation time 10. In this time if we take single packet server (one packet will be served and then another) and arbitrary service time, the model would be the Model-1 and we will get the following output figures:

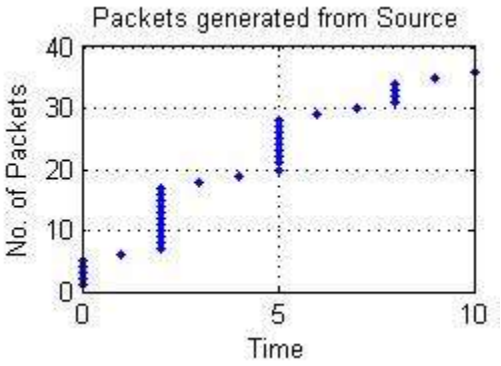


Fig-1 Packets generated from Source

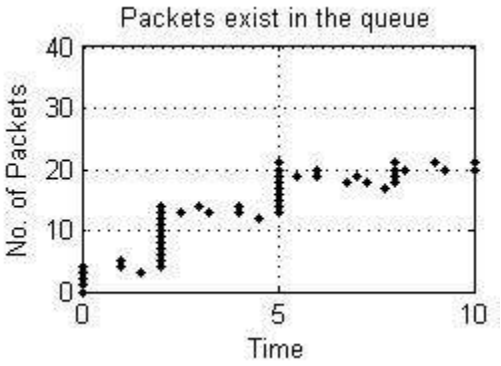


Fig-2 Packets exist in the queue

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

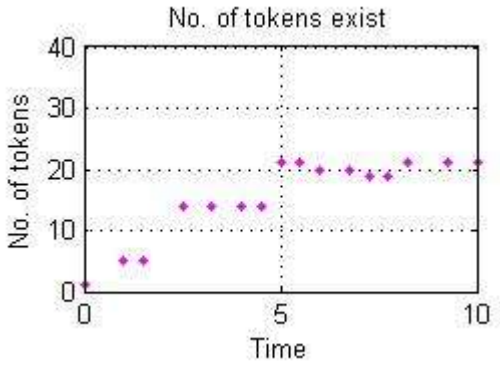


Fig-3 No. of tokens exist

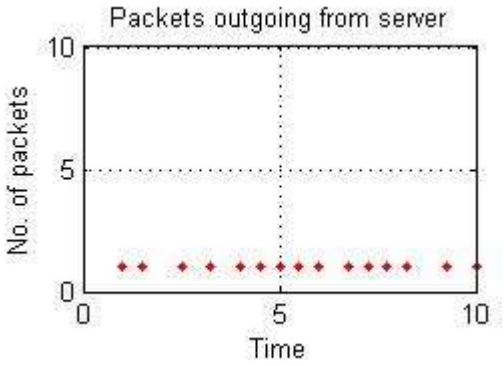
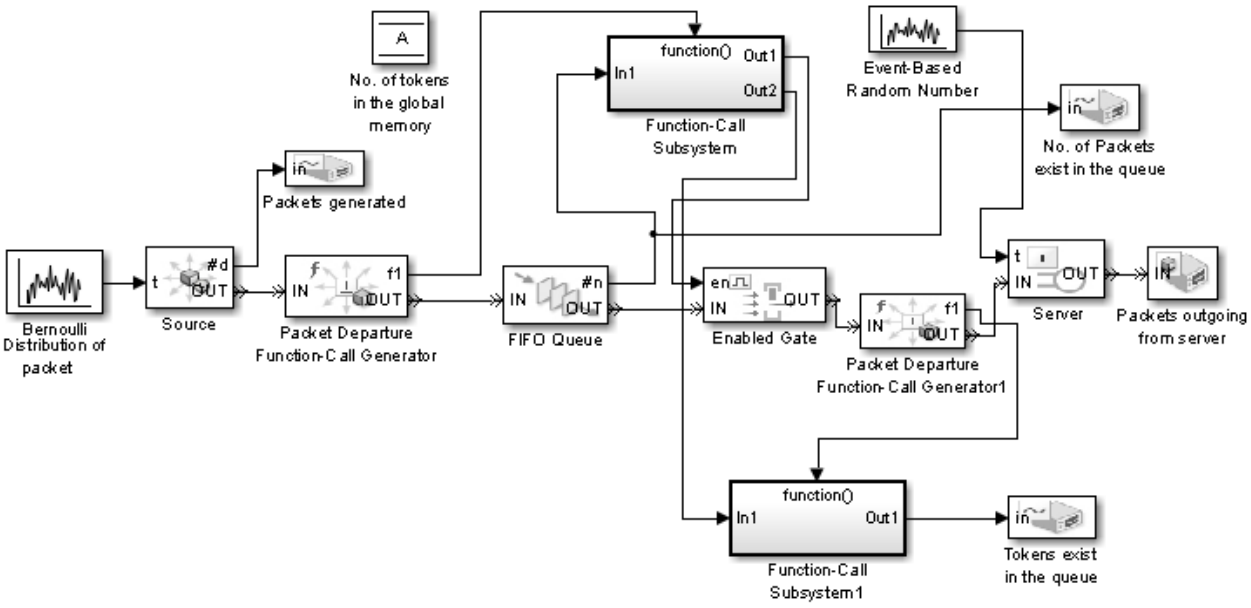


Fig-4 Packets outgoing from server



Model-1 Reviewed token bucket

In the model we used the Event-Based Random Number block to show that the server service time is arbitrary. Here we have got 37 packets from the source (Fig-1) , about 22 packets exist in the queue (Fig-2) and less number of packets outgoing from the server (Fig-4). Here **throughput (packets/second) is (packets outgoing from server/simulation time) = 15/10=1.5.**

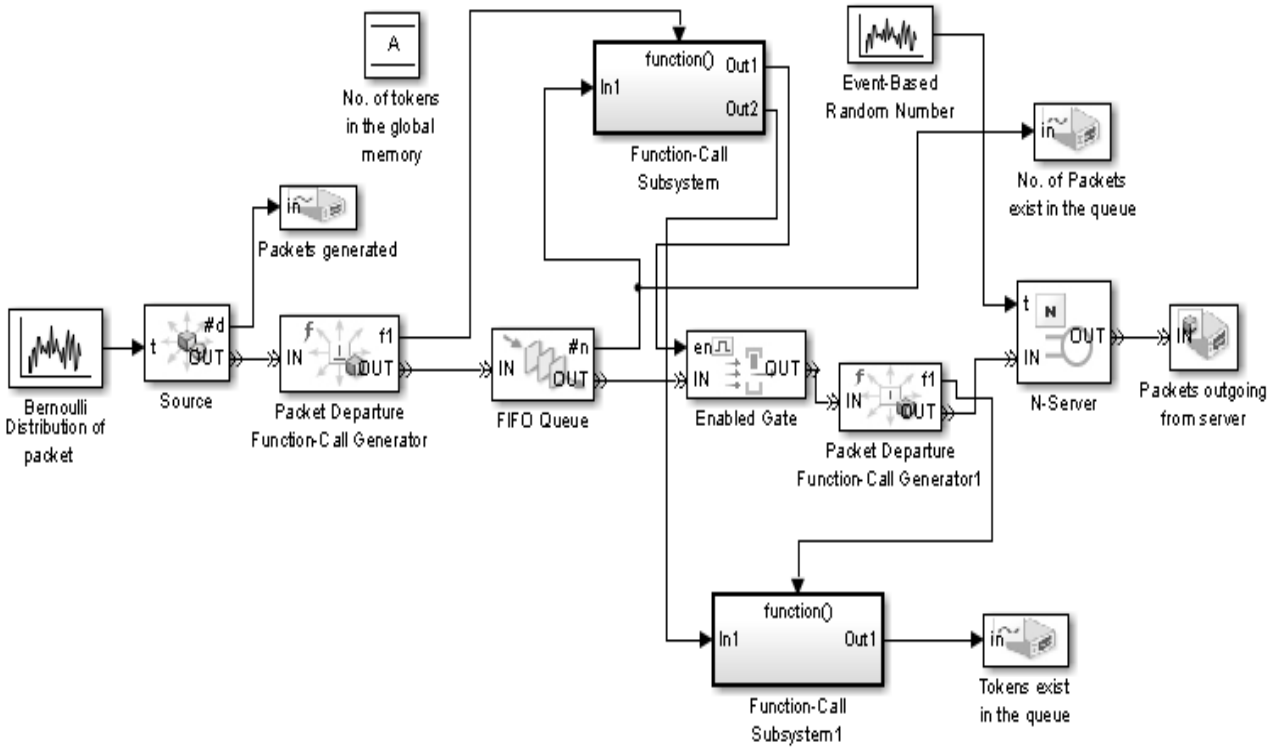
If we want to get more throughput , we have found out a formula and taken N-server in the model like the following (Model-2):

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

The formula will be like this:
Desired number of packets in server (N-server)= α/β
α = packets exist in the queue
β= desired number of packets will be exist in the queue

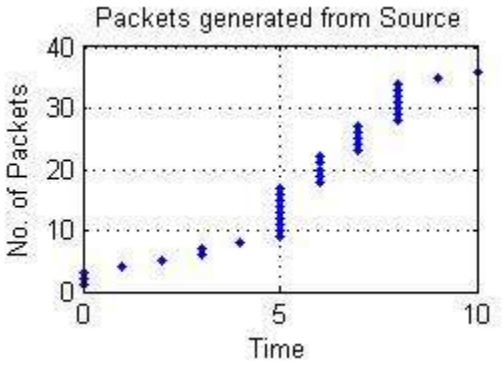
Actually less number of packets will be exist in the queue than the desired number of packets. For example here, from Model-1, we have got 22 packets in the queue. If we would like to exist 5 packets in the queue then 22/5=4 packets can be taken by the server, that is 4-server can be taken. We have got 2 packets (Fig-6) in the queue (which is less than the desired packets) and huge number of packet outgoing from server (Fig-8). Here throuoghput (packets/second) from the model Model-2 is (packets outgoing from server/Time) = 35/10=3.5. **It is more than double than the previous one (Model-1).**



Model-2 More reviewed token bucket

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com



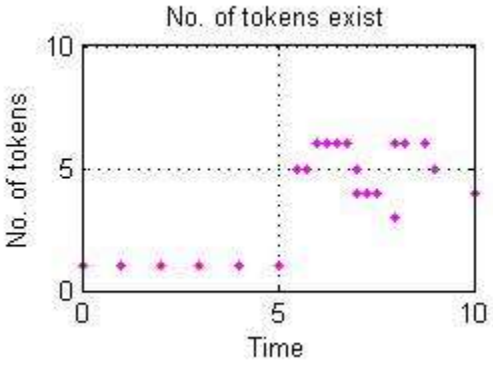


Fig-5 Packets generated from Source

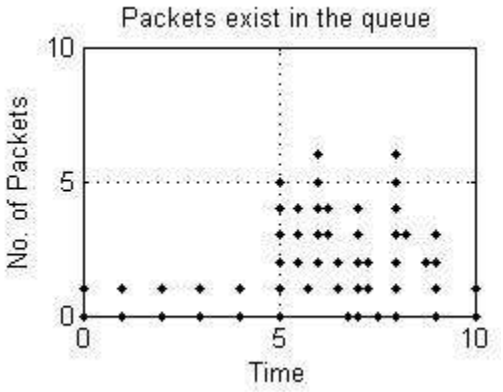


Fig-6 Packets exist in the queue

Fig-7 No. of tokens exist

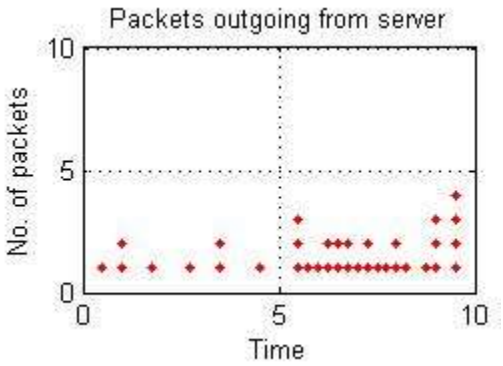


Fig-8 Packets outgoing from server

Configuration parameters of different important blocks of

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

the Model-2 are given below:

1. Source: Bernoulli Distribution of packet
2. FIFO Queue: capacity=25
3. N-Server: 4- servers, service time=from signal port (arbitrary service time)

Coding of different Matlab function blocks are the same as in [1]. However the **working principles** are described below:

Source will send packets. When packets depart from source then the function call will be generated by packet departure function call generator block and then the number of packets in the queue will be measured in the MATLAB FUNCTION block. However the packets exist in the queue=u. If u=0 then x=0, that is the Out1 of the Function-Call Subsystem will be '0' and the gate will be disabled. In this case no packets will be passed. If u<A the gate will be enabled and the packets will be passed. 'A' is a global memory. Its initial value=2 and maximum value =25. The value of 'A' is the number of tokens. When a packet is passed, token number is reduced by '1'(calculated in the Function-Call Subsystem1). Otherwise the token number will not be increased. But one thing is to remember that if A=0, that is if there is no token then token will be increased by some quantity, here it is 5.

4. ANALYSIS OF THE JAM FREE DESIGN OF PACKET CONGESTION:

In [1], the author designed a model of Jam free design of packet congestion. From that model (Fig-8) of the reference [1], we saw packets 15, 10 and 13 from source1, source2 and source3 respectively for 18 seconds simulation time. So there were 38 packets from 18 seconds. So packet generate rate

= total packets/time=38/18=2.11 packets/second=arrival rate=α .

From server (source side) we have found 28 packets from 18.5 seconds. So one packet will pass in 18.5/28=0.66 second=average service time=β.

From the destination side server we have found (in that model of reference [1]) 27 packets in 19.9 seconds. So departure rate = total packet departed/time =27/19.9 =1.36 packets/second=throughput of the system.

Now we can take the difference between arrival rate and departure rate as packets remain/second in the system. We can get 2.11-1.36=0.75 packets in the system per second.

For arrival rate 2.11 we get 0.75 packets in the system. Therefore we get (0.75*100)/2.11=35.5% remaining of packets in the system.

Here 9 packets exist in the queue in about 18.5 seconds. So of course it is congested or Jam. Therefore 9/18.5=0.49 packets/second as a jam=queue length.

So jam is proportional to queue length

Therefore Jam=k * queue length

So k=jam/queue length

Here we got 35.5% packets remain in the system. **It is not a good system (Fig-8) of the reference[1].** So if we take it as a jam situation then we can apply the formula above. Therefore we can get k=0.75/0.49=1.5. So queue length=35.5/1.5=24. So average queue length will be 24 if 100 packets come per second.

We got average queue length=8 (maxth = maximum threshold) for queuing capacity 10. Therefore here we will get (capacity * total queue length)/average queue length = (10*24)/8 = 30, which is a measured queue length. Always we will take the round figure for queuing capacity. However here we have got 30. So we will need 3 queue of having capacity 10 each. Thereforefor 3 sources, number of recommended queues will be 3. I mean the ratio of source and queue will be 1:1. **But this ratio is not always good enough because we may not have enough space or buffer to hold the packet.** Therefore the sources should take the controlling power of generating packets which was shown in that algorithm of Fig-8 in the reference [1].

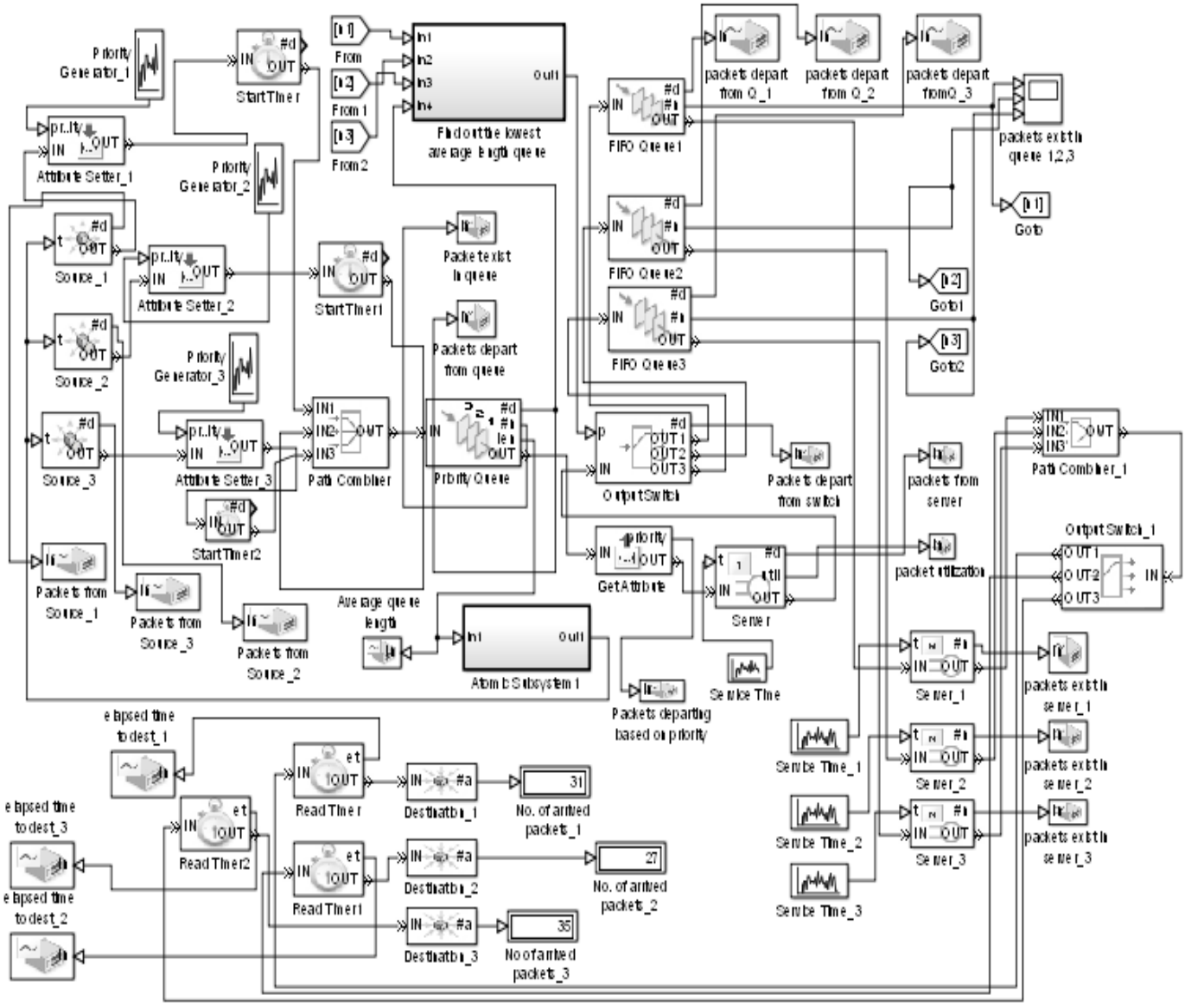
In that paper [1] Fig-17 Atomic Subsystem1 describing the controlled intergeneration time will be Uniform random number. **But taking critical analysis we are now thinking like the following**:

If we use the uniform random distribution for packet generation where Minimum parameter and Maximum parameter are '0' and '1' respectively. So the probability of having 0 to 0.3 is $\int_0^{0.3} 1dx$ =1(0.3-0) = 0.3**. But 0 to 0.3**

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

is a very little amount of intergeneration time. If very little amount of time comes again and again the system may be congested. So if we change the value of (0 to 0.3) to (0.4 to 0.9) after one occurrence of (0 to 0.3) within three occurrence of (0 to 0.3) then we will get extra jam free situation from the following model:



Model-3 Reviewed Jam free design of packet congestion

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

```
//Code of the MATLAB block function to get extra jam
free situation (Fig-9 ):
function y = fcn(u,v)
%#codegen
global A;
global B;
maxth=8;
minth=6;
if u<=minth
       if (v<=0.3) &&(A==1)
          y =v;
          A=A+1;
       else
            if v<=0.3
       y=1;
   else
       y=5;
          y=B;
            A=A+1;
           B=B+0.1;
       else y=v;
       end;
       if A>3
          A=1;
       end;
       if B>0.9
          B=0.5;
       end;
     end;
    else
if u<maxth
   end;
end;
```

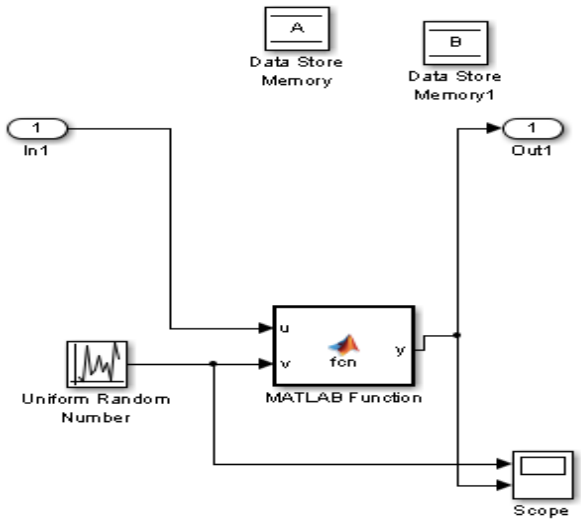


Fig-9 Atomic Subsystem1 details

Running the above model Model-3, we get the following figures:

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

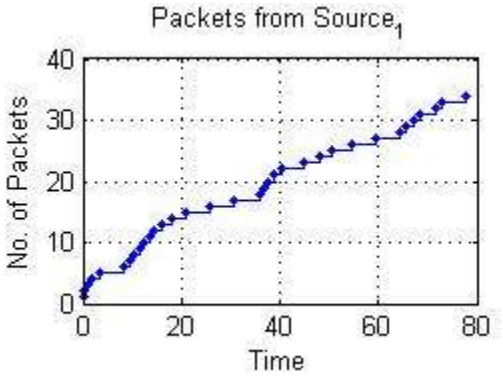


Fig-10 Packets from Source1

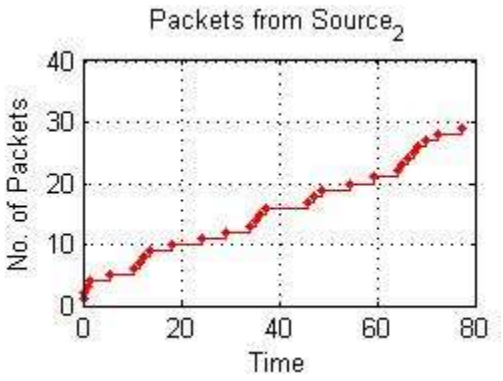


Fig-11 Packets from Source2

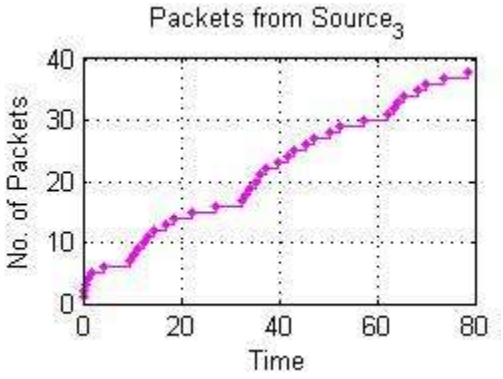


Fig-12 Packets from Source3

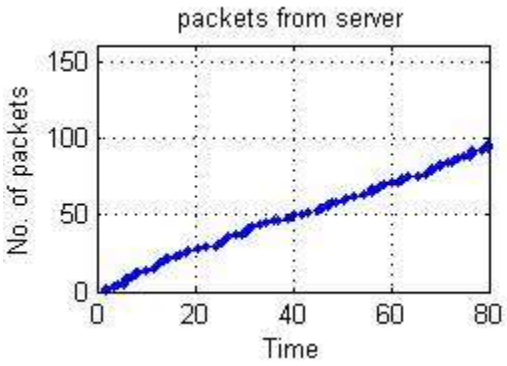


Fig-13 Packets from server

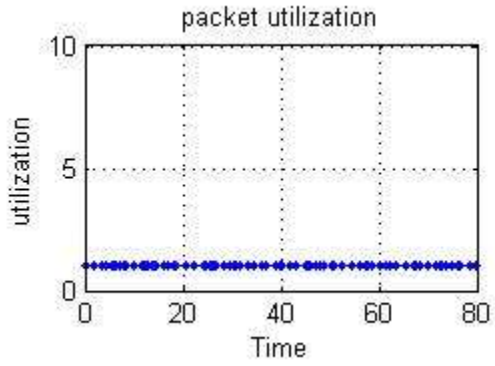


Fig-14 Packet utilization

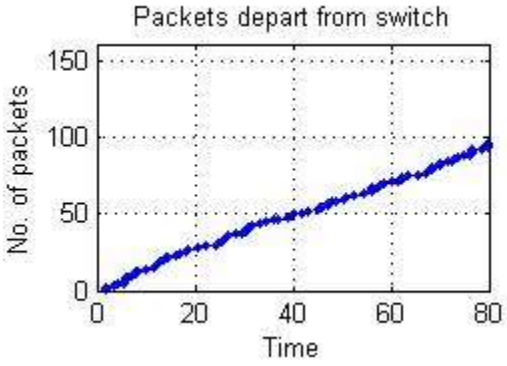


Fig-15 Packets depart from switch

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

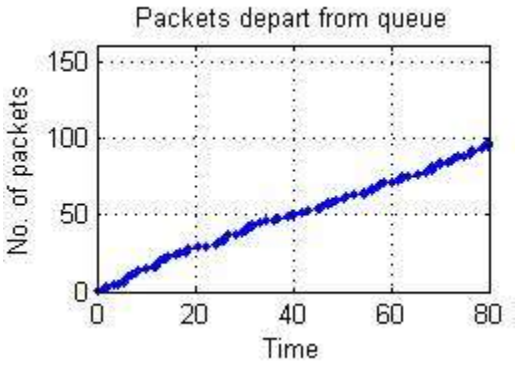


Fig-16 Packets depart from queue

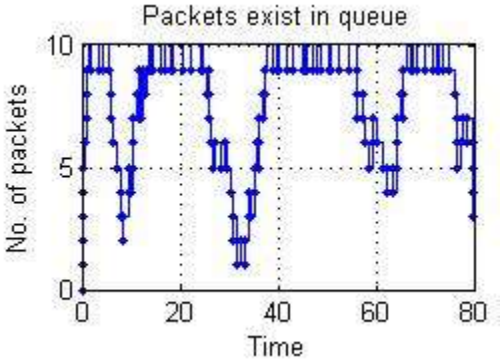


Fig-17 Packets exist in queue

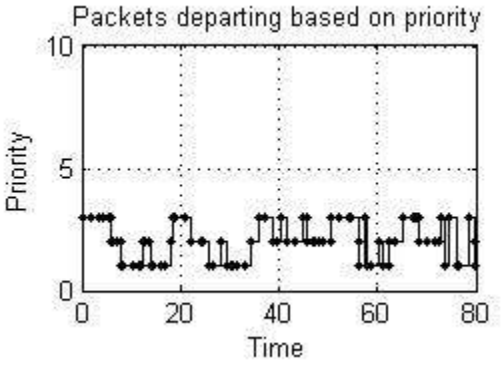


Fig-18 Packets departing based on priority

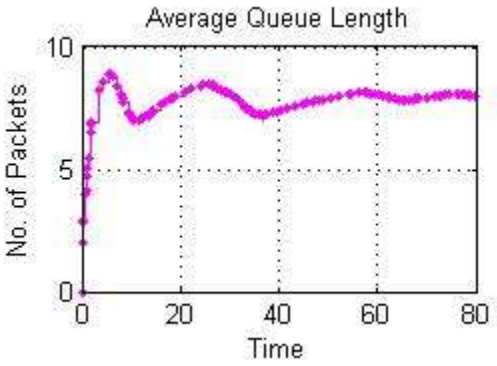


Fig-19 Average queue length

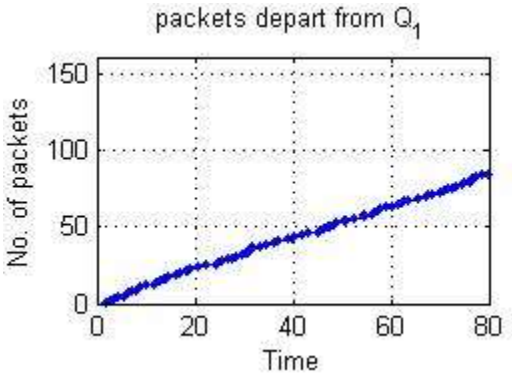


Fig-20 Packets depart from queue1

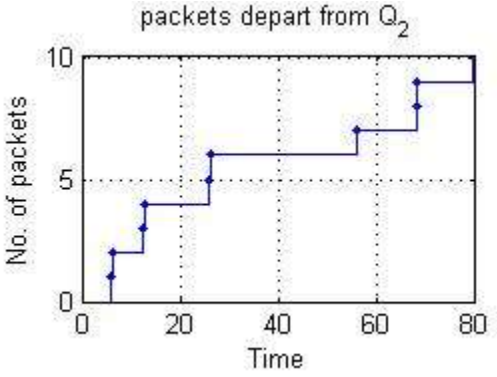


Fig-21 Packets depart from queue2

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

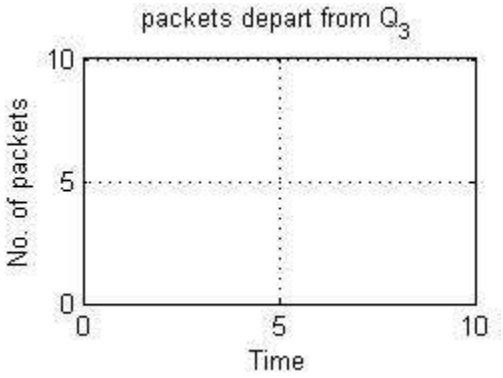


Fig-22 Packets depart from queue3

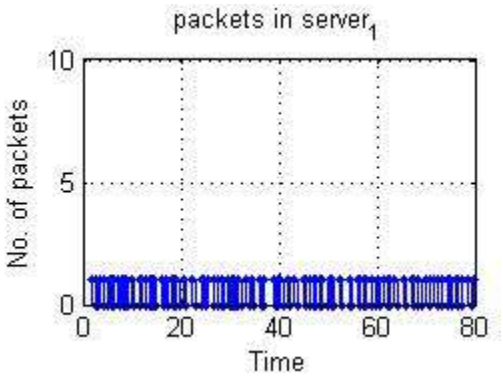


Fig-23 Packets in server1

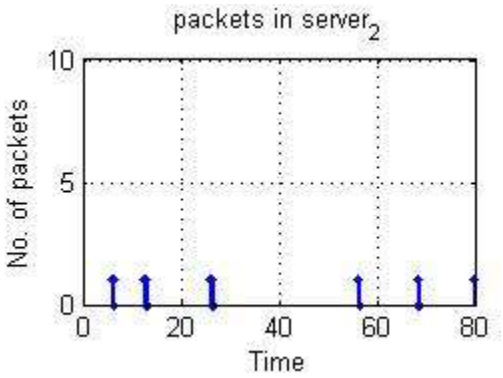


Fig-24 Packets in server2

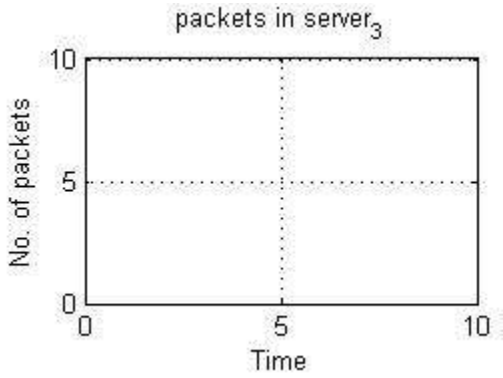


Fig-25 Packets in server3

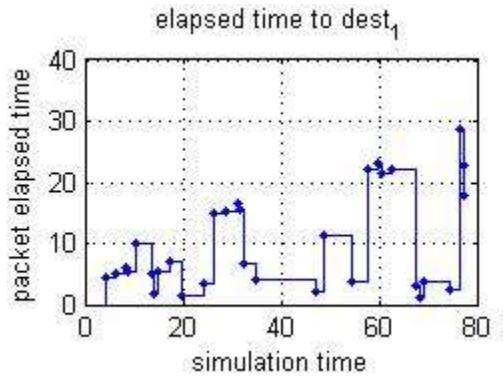


Fig-26 Elapsed time to destination1

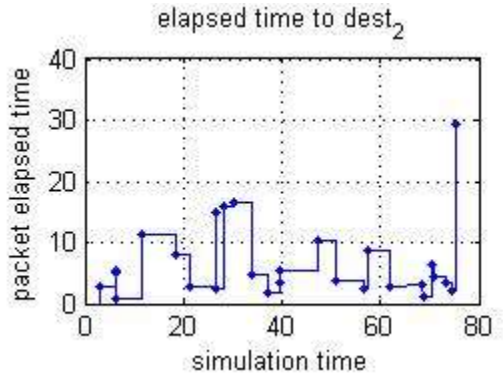


Fig-27 Elapsed time to destination2

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

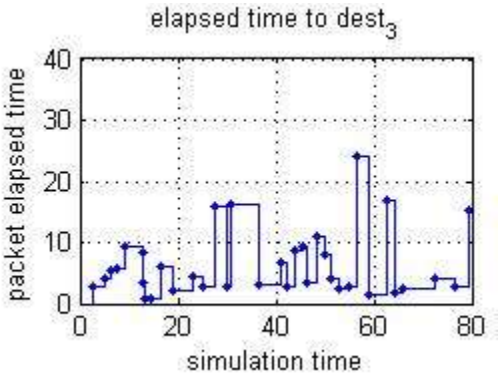


Fig-28 Elapsed time to destination3

Running the above model (Model-3), we get 34, 29 and 38 packets from source_1, source_2 and source_3 respectively during a simulation time of 80 seconds. So we have got 101 packets from 79 seconds. Therefore the arrival rate of packets in the system is 101/79=1.28 packets/second. We have 93 packets departed in 80 seconds. So the departure rate = total packet departed from the system /total time = 93/80 =1.16 packets/second. Packets remaining in the system is 1.28-1.16=0.12 packets/second. So for 100 packets we have got (0.12*100)/1.28=9.4 packets/second remained in the system. Therefore **the throughput is very high**.

Configuration of different important blocks parameters of Model 4-9 are given below:

1. Source_1: generation event priority=500
2. Source_2: generation event priority=400
3. Source_3: generation event priority=300
4. Attribute Setter_1: priority=signal port, dest=1
5. Attribute Setter_2: priority=signal port, dest=2
6. Attribute Setter_3: priority=signal port, dest=3
7. Priority Queue: capacity=10, sorting attribute name = priority, sorting direction=descending
8. Server: single server: service time=from signal port t
9. Output Switch: number of entity output ports=3, switching criterion=from signal port p
10. FIFO Queue1: capacity=04
11. FIFO Queue2: capacity=05
12. FIFO Queue3: capacity=04
13. Server_1: no. of servers=1
14. Server_2: no. of servers=1
15. Server_3: no. of servers=1
16. Output Switch_1: number of entity output ports=3, switching criterion= from attribute=dest

5. WORKING PRINCIPLE OF THE REVIEWED JAM FREE DESIGN OF PACKET CONGESTION (MODEL-3):

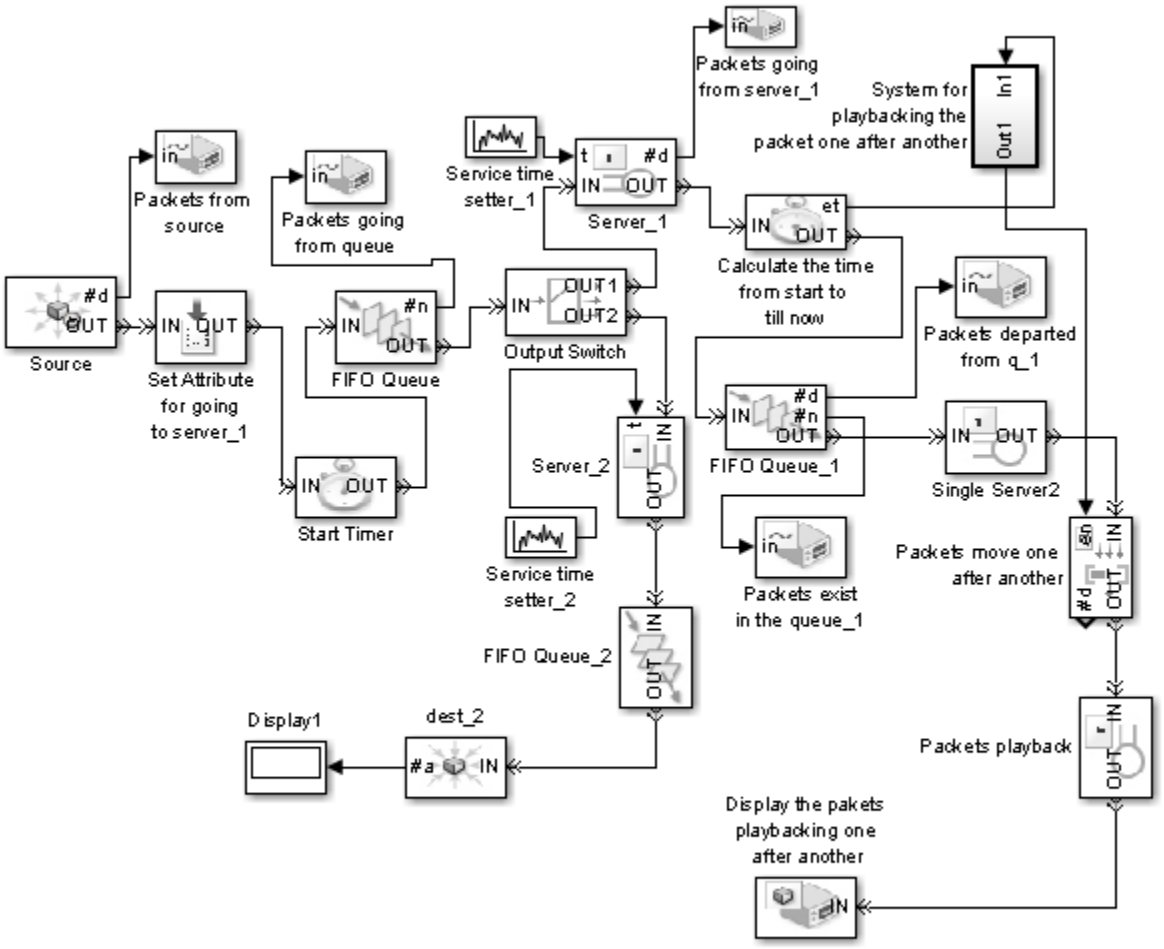
Source_1, Source_2 & Source_3 will send packets. Attribute Setter_1, Attribute Setter_2 and Attribute Setter_3 will set the attributes of the packets of Source_1, Source_2 and Source_3 respectively. Two attributes are: i) priority and ii) dest. The packets will go according to priority and destination (dest). Server will give service according to arbitrary service time. This is the source side server. Destination side servers like Server_1, Server_2 and Server_3 also take arbitrary service time. Here in this model we have used single sever, that is one packet will be served and then another. Here in Atomic Subsystem1 block (Fig-9), there are some calculations performed, that is the sources will take the special controls. Here global variable 'A' is used for maintaining the repetition of 0 to 0.3 and 'B' is used for using 0.4 to 0.9 instead of repetition of (0 to 0.3) again and again.

6. JITTER CONTROL:

Jitter Control for applications such as audio and video streaming, it does not matter much if the packets take 20 msec or 30 msec to be delivered, as long as the transit time is constant. The variation (i.e., standard deviation) in the packet arrival times is called jitter.

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com



Model-4 Jitter Control

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

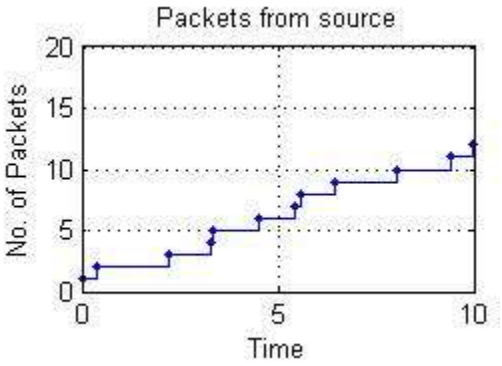


Fig-29 Packets from source

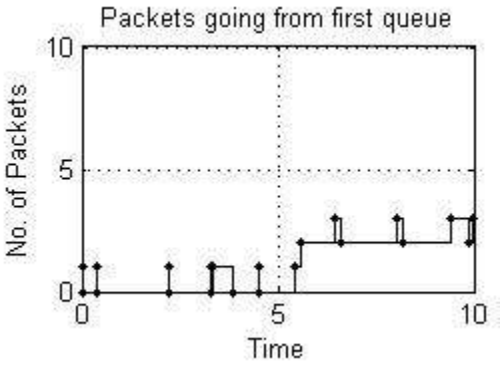


Fig-30 Packets going from first queue

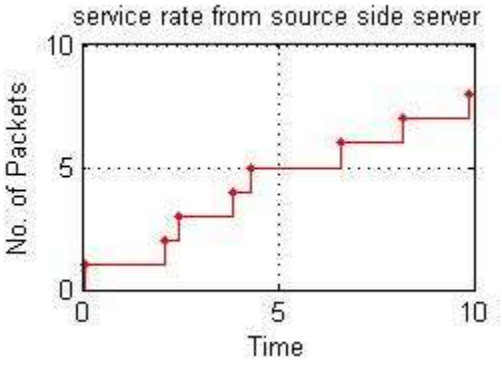


Fig-31 Service rate from source side server

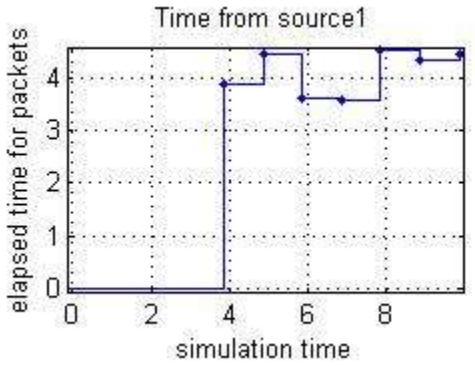


Fig-32 Time from source1

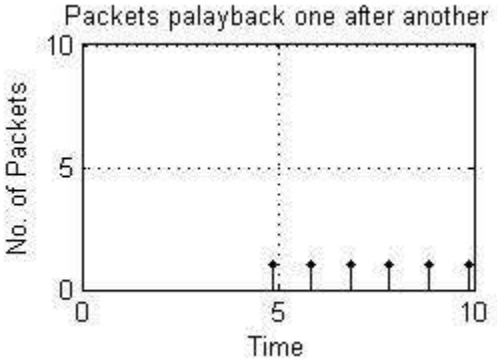


Fig-33 Packets play back one after another

From the figure Fig-29 we have found 12 packets in 10 seconds (simulation time) arrived in the system. So here the arrival rate = packets/time= 12/10=1.2 packets/second.

From the figure Fig-32 we have found that 7 packets have arrived in 3.9, 4.5, 3.7, 3.7, 4.5, 4.3 and 4.5 seconds respectively. So each packet spends average in the queue = (3.9+4.5+3.7+3.7+4.5+4.3+4.5)/7 =4.157=4.2 seconds.

From Little's law we know $L=\lambda\omega$
Where L=average number of packets in the system at any time

λ= arrival rate of packets
ω=packet spends average time in the queue

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

Therefore L=λω=1.2*4.2=5.04

Here the two queues, FIFO Queue and FIFO Queue_1 have total queue length=10+10=20
Therefore space in the queues free=20-L=20-5=15

So (free space *100)/total queue length = (15*100)/20 =75% free space I have got. So this type of system is very helpful to capture bursty traffic.

Variance = $\sum_{i=1}^{7} \frac{(x_i-a)^2}{7}$, where xi= i-th packet existing time in the system, a=mean existing time = 4.2

$=\{(3.9-4.2)^2 +(4.5-4.2)^2 +(3.7-4.2)^2 +(3.7-4.2)^2 +(4.5-4.2)^2 +(4.3-4.2)^2 +(4.5-4.2)^2\}/7$

=0.12428

So the standard deviation= $\sqrt{0.12428}$ =0.0503
Therefor **the packets exist very effectively in the system.**

The code for the figure Fig-34 is given below:

```
function y = fcn(u)
%#codegen
global A;
if (A==0||A<=2)
A=A+u;
y=0;
else
   y=1;
end;
```

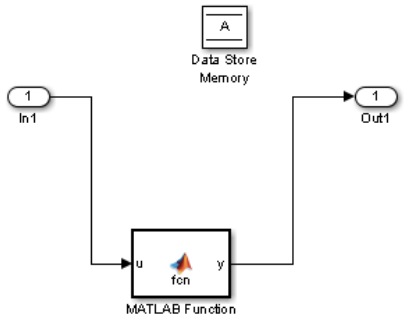


Fig-34 Syatem for playbacking the packet one after another

Configurations of different important blocks parameters of the figure Model-4 are given below:

1. Source: packet distribution =exponential
2. Set Attribute for going to server_1: dest=1
3. FIFO Queue: capacity=10
4. Server_1: single server, service time=from signal port t
5. Output Switch: number of entity output ports=2
6. Switching criterion: from attribute=dest
7. FIFO Queue_1: capacity=10
8. Single Server2: service time=0 and dialog system
9. Packets for playback: time=1 second
10. Packets move one after another: enable gate=1, disable gate=0

Working Principle of the model Model-4, Jitter Control is given below:

Source will send data. 'Set Attribute' block for going to 'Server_1' block sets the attribute 'dest' (destination) is '1'. The output switch will pass the packet by attribute des t=1. That is the packet will pass from Out1 port. This path is for controlling jitter. The packets will go through FIFO Queue_1 and finally packet will pass through the 'Packets move one after another' block which is a gate. This gate will be enabled when the 'system for play backing the packet one after another' block sends an output '1'. This block will give the output '0' when A=0 and A<=2. 'A' is a global memory and its initial value is '0'. The value of 'A' like this so that the time may reach up to some quantity for maintaining the jitter. Here it is '2'. After time is greater than '2' the gate will be enabled.

7. CONGESTION CONTROL IN MULTICASTING:

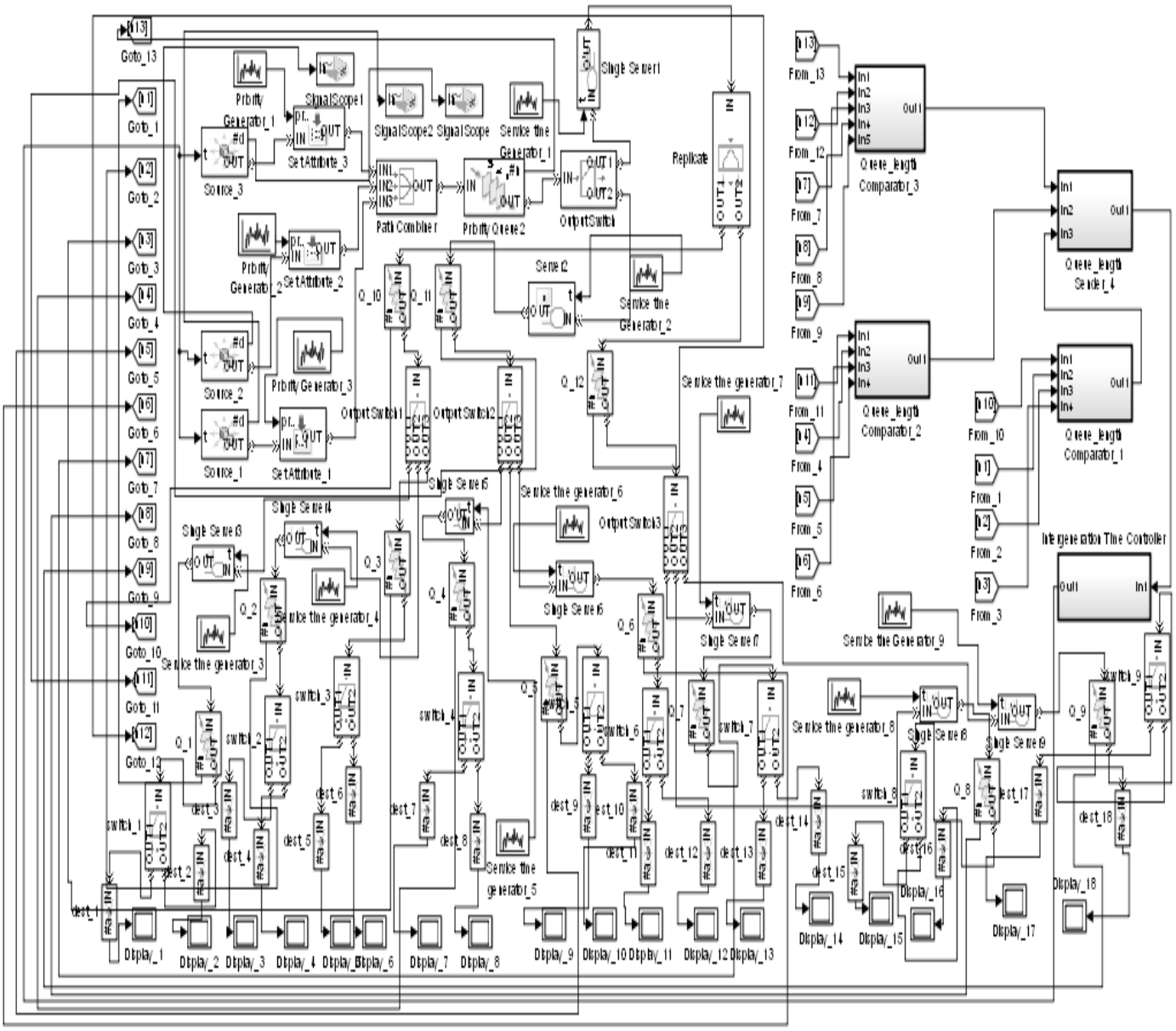
ISSN 2304-0777 www.jitbm.com

In [2]. The author developed an algorithm for the congestion control of multicasting. But we have found an improved step:

We think if the sources can reduce their packet sending rate by u/2 (‘u’ is measured by the following algorithms of different MATLAB block functions and we have got the very effective result of mitigating maximum jam of congestion. The AIMD [2] is well. Here multiple decreases can take place. But we think it is not sufficient. Because how much congestion has taken place is very much important and that’s why we have taken u/2. The u/2 is described below sequentially**:**

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com



Model-5 Multicast congestion control

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

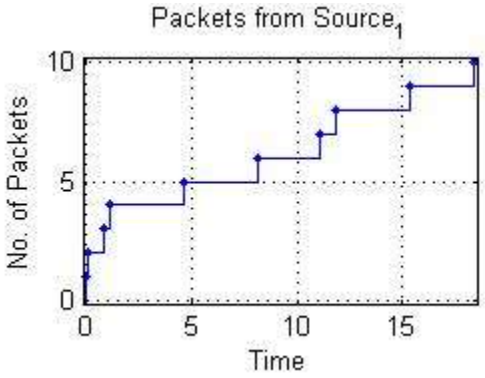


Fig-35 Packets from Source1

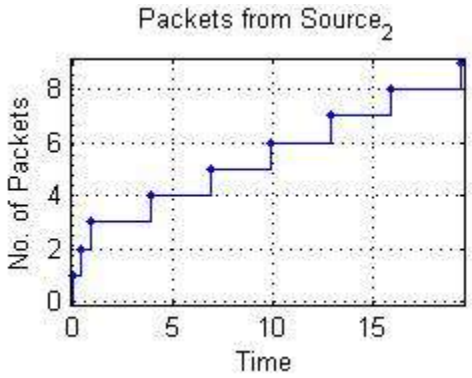


Fig-36 Packets from Source2

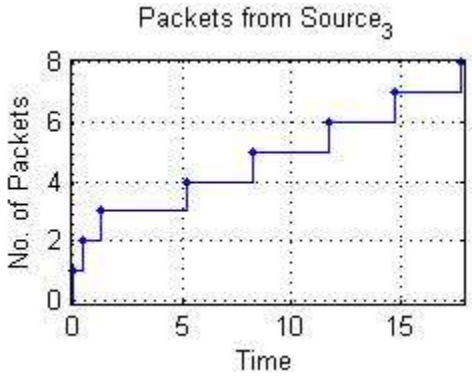


Fig-37 Packets from Source3

It is difficult to block all the congested portion when one congestion indication has found. So the revised congestion selection is given below:
Here are three blocks: Queue_length_Comparator_1, Queue_length_Comparator_2 and Queue_length_Comparator_3. From Comparator_1 we get a congestion indication queue length. From Comparator_2, and Comparator_3 we also get the congestion indication queue length.

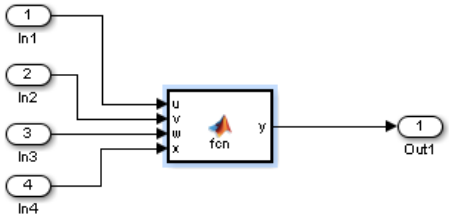


Fig-38 MATLAB Function of Queue_length Comparator_1

// code of the above figure Fig-38

```
function y = fcn(u,v,w,x)
%#codegen
if u>=6
   y=u;
else
   if v>=6
      y=v;
   else
      if w>=6
       y=w;
      else
          if x>=6
             y=x;
          else
             y=0;
          end;
      end;
   end
end;
```

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

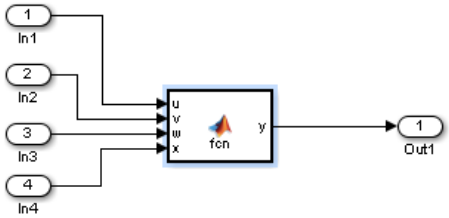


Fig-39 MATLAB Function of Queue_length Comparator_2

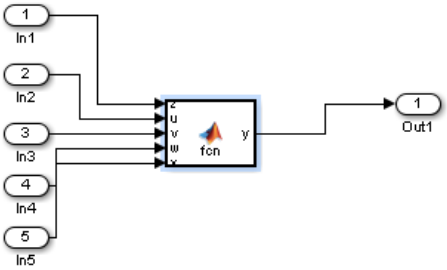


Fig-40 MATLAB Function of Queue_length Comparator_3

//code of the above figure Fig-39

```
function y = fcn(u,v,w,x)
%#codegen
if u>=6
   y=u;
else
   if v>=6
      y=v;
   else
      if w>=6
       y=w;
      else
          if x>=6
             y=x;
          else y=0;
          end;
      end;
   end
end;
```

//code of the figure Fig-40

```
function y = fcn(z,u,v,w,x)
%#codegen
if z>=6
   y=z;
else
  if u>=6
    y=u;
  else
    if v>=6
      y=v;
    else
      if w>=6
        y=w;
      else
          if x>=6
            y=x;
          else y=0;
          end;
      end;
    end;
  end;
end;
```

International Journal of Information Technology and Business Management
29th Aug 2015. Vol.40 No.1

ISSN 2304-0777 www.jitbm.com

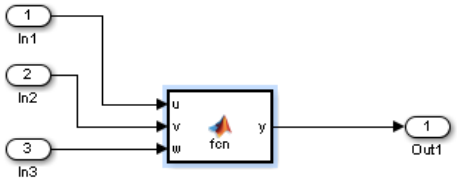


Fig-41 MATLAB Function of Queue_length Sender_4

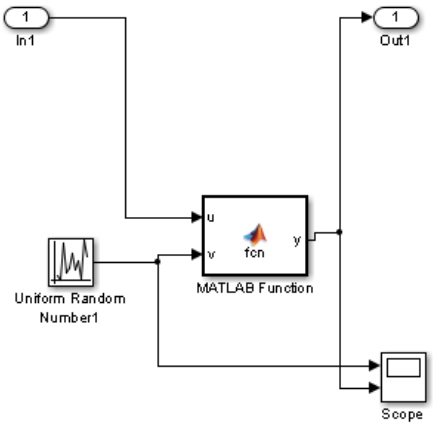


Fig-42 Intergeneration time controller subsystem

//code of the above figure Fig-41

```
function y  = fcn(u,v,w)
%#codegen
 a=[u,v,w];
 r=0;
 s=0;
 t=0;
 if a(1)~=0
    r=a(1);
 else
    if a(2)~=0
       s=a(2);
    else
       if a(3)~=0
          t=a(3);
       end;
    end;
 end;
 p=max([r s t]);
 if p==0
    y=0;
 else
    y=p;
 end;
```

Queue_length sender_4 block sends the maximum queue length of the Comparator_1, Comparator_2 and Comparator_3.

//code of the intergeneration time controlller

```
function y = fcn(u,v)
%#codegen
if u>=6

    y =u/2;
else
    y=v;

end;
```

This block finally measures the data sending rate (intergeneration time). That is the sender will send data by the measured time (u/2).

Finally the **revised algorithm** is:
1. The Congestion is considered if 60% or above queue length is taken place.
2. Congestions from different portions is taken into consideration and the maximum queue length (u) is taken.
3. Finally u/2 is measured as intergeneration time of packets.

8. FURTHER RESEARCH:

One thing is important here is that u/2 is measured by all the source. But if the particular source can be identified for more jam and if this source can be controlled, throughput can be increased.

ISSN 2304-0777 www.jitbm.com

However the configuration parameter of different important blocks of the Model-5 are given below:

1. Source_1 block: Generation event priority=300
2. Source_2 block: Generation event priority=400
3. Source_3 block: Generation event priority=500
4. Set Attribute_1: priority=Signal port, dest=2, dest1=1
5. Set Attribute: priority=Signal port, dest=2, dest1=2
6. Set Attribute: priority=Signal port, dest=1, dest_1=2
7. Priority Queue2: capacity=10
8. Output Switch: Switching criterion=From attribute=dest
9. Replicator: Number of entity output ports=2, departure port precedence=OUT1 port
10. Intergeneration time controller: Uniform random number generator: minimum=0, maximum=1

9. WORKING PRINCIPLE OF THE MUTICAST CONGESTION CONTROL BLOCK:

Sources will send packets and output switch will pass the packet according to the value of 'dest'. For example if dest is '1' then packet will go through Out1 port. Here multicasting is used. That is packet will be passed to the group of destinations. Moreover one packet may be replicated to the group of destinations.

10. CONCLUSION AND FUTURE WORKS:

In future we will try to find out an algorithm implementing both closed-loop and open loop systems for the removal of congestion in computer networks.

11. REFERENCE:

1. Shaikh Shariful Habib. Improved algorithms for Token bucket and closed-loop control system in Congestion control of computer networks. International Journal of Information Technology and Business Management pp 026 - 034 Vol. 037. No 1 – 2015

2. Shaikh Shariful Habib. An Improved Step In Multicast Congestion Control of Computer Networks. International Journal of InformationTechnology and Business Management pp 012 016 Vol. 036. No. 1 -- 2015
3. Andrew S. Tanenbaum, Computer Networks, fourth Edition