

Performance Evaluation of Computer Systems
Prof. Krishna Moorthy Sivalingam
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture No. # 37

Programming aspects of discrete-event simulations-II

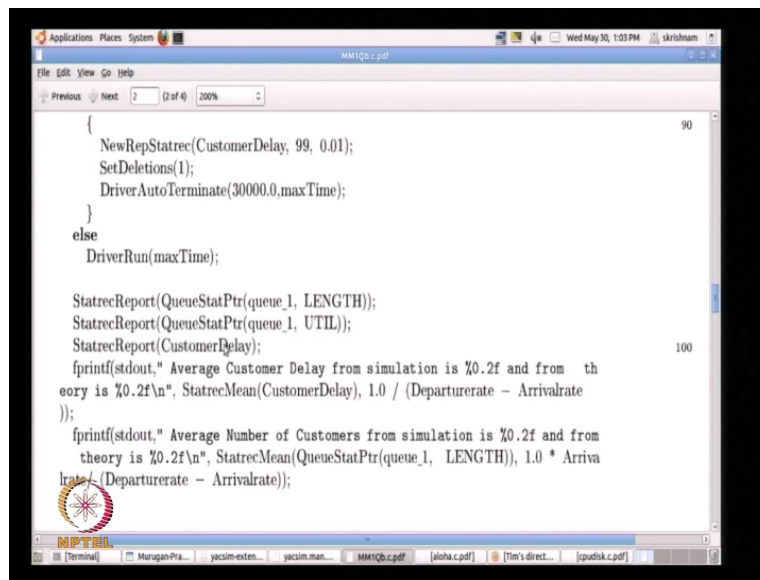
We are looking at this C program implementation that the action of the mm 1 system, so we talked about different components, the arrival process **right**, which captures how packets are the generator and queued. And actually this service process is hidden, we use the resources with available in action do that, so we are actually we can also done the separately, we could written our own code for adding a packeted queue, dequeuing packet all that done by our self.

But for the sake of convenience, we are using that resources that is available **right**, we look at that other example where we doing dequeuing our self; this example we are simply just **right**, using a resource, so the questions?

(())

So, the question is how do we find out the queue length of the system, at intermediate point in the time **right**, in this case I am only printing the queue length at the end of the stimulation.

(Refer Slide Time: 01:09)



```

{
    NewRepStatrec(CustomerDelay, 99, 0.01);
    SetDeletions(1);
    DriverAutoTerminate(30000.0,maxTime);
}
else
    DriverRun(maxTime);

StatrecReport(QueueStatPtr(queue_1, LENGTH));
StatrecReport(QueueStatPtr(queue_1, UTIL));
StatrecReport(CustomerDelay);
printf(stdout," Average Customer Delay from simulation is %0.2f and from th
eory is %0.2f\n", StatrecMean(CustomerDelay), 1.0 / (Departurerate - Arrivale
rate));
printf(stdout," Average Number of Customers from simulation is %0.2f and from
theory is %0.2f\n", StatrecMean(QueueStatPtr(queue_1, LENGTH)), 1.0 * Arriva
lrate / (Departurerate - Arrivale
rate));

```

So, this if you look at the Statrec report function, this take as an argument the queue stat point function, which in turn return the Stat record, and this Statrec continuously updated by the stimulator, by action. So, we can call other any point and time to look at current length of the queue; so you can print not only just the mean **right**. You can print other things also you can print the range of value that your looking at, so there is a fairly sophisticated set of functions that are available part of the Statrec to get all other values **right**.

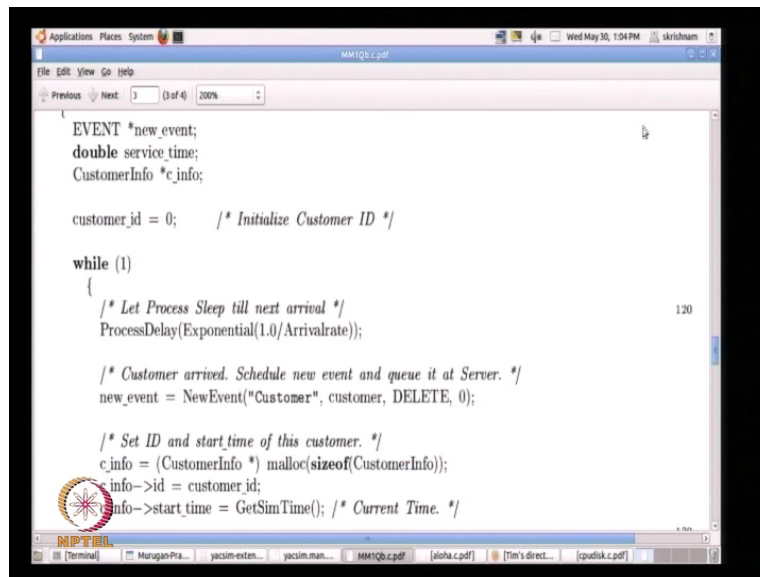
So, we can look at the mean length of the queue at difference point in time **right**, so that is what here I am looking at the mean length of the queue at the end of the stimulation **right**, based on all the samples that has been collected in various systems. So, in terms of sample collection for mean length customer delay is very clear, because every time a packet finishes, I compute the **(())**, I just update the statistic record, whereas mean length is not like that **right**.

Because, the length of queue can be measured at any point and time but you look at some specified in instance of time and you can measure the length of the queue at that time, or you can also measure the length of the queue only at packet arrivals or packet departure. You can define as to when you want to do these measurements, and then update the samples according **right**, and you can get it corresponding.

So, that is the basic notation of that is the basic idea **right**, there is an arrival process which we discussed earlier on, and then there is the customer which is also discussed earlier on; so

these are the two, one is even handler customer and the other is the process logic right. And if you look at typically all the process logic functions will be like this, will be mostly endless loop the keep on generating package, and then when the system terminate automatically everything it is cleaned up, so that is what have so far.

(Refer Slide Time: 02:45)



```
EVENT *new_event;
double service_time;
CustomerInfo *c_info;

customer_id = 0; /* Initialize Customer ID */

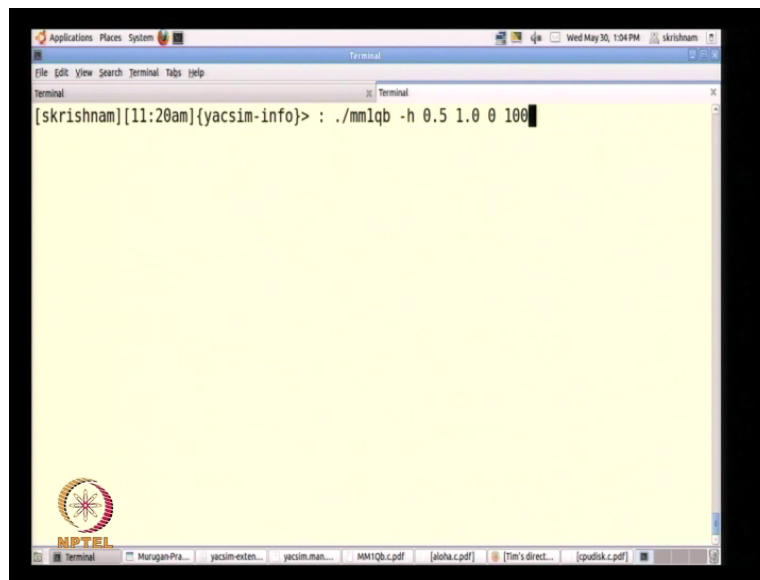
while (1)
{
    /* Let Process Sleep till next arrival */
    ProcessDelay(Exponential(1.0/Arrivalrate));

    /* Customer arrived. Schedule new event and queue it at Server. */
    new_event = NewEvent("Customer", customer, DELETE, 0);

    /* Set ID and start_time of this customer. */
    c_info = (CustomerInfo *) malloc(sizeof(CustomerInfo));
    c_info->id = customer_id;
    c_info->start_time = GetSimTime(); /* Current Time. */
}
```

Now, let us try to run this code (Refer Slide Time: 02:50) and see it is actually are able to match some of these right, so I set that we have this mm 1 q minus h right, and different parameter where arrival rate, 0.5 departure rate 1.0. And we said, we do not want the repudiation right, I am not doing that convert I am just giving it some raw values, so let us say I will run this for 100 units of the time.

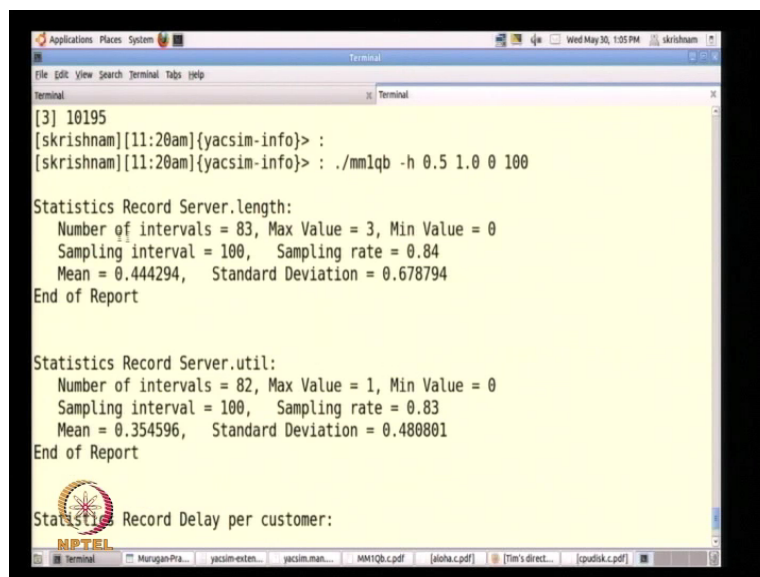
(Refer Slide Time: 03:13)



```
Applications Places System
Terminal
File Edit View Search Terminal Tabs Help
Terminal
[skrishnam][11:20am]{yacsim-info}> : ./mm1qb -h 0.5 1.0 0 100
```

So, now let us look at us what the output of this system is, there is whole bunch of values **past a way**, let us look at this slowly.

(Refer Slide Time: 03:22)



```
Applications Places System
Terminal
File Edit View Search Terminal Tabs Help
Terminal
[3] 10195
[skrishnam][11:20am]{yacsim-info}> :
[skrishnam][11:20am]{yacsim-info}> : ./mm1qb -h 0.5 1.0 0 100

Statistics Record Server.length:
  Number of intervals = 83, Max Value = 3, Min Value = 0
  Sampling interval = 100, Sampling rate = 0.84
  Mean = 0.444294, Standard Deviation = 0.678794
End of Report

Statistics Record Server.util:
  Number of intervals = 82, Max Value = 1, Min Value = 0
  Sampling interval = 100, Sampling rate = 0.83
  Mean = 0.354596, Standard Deviation = 0.480801
End of Report

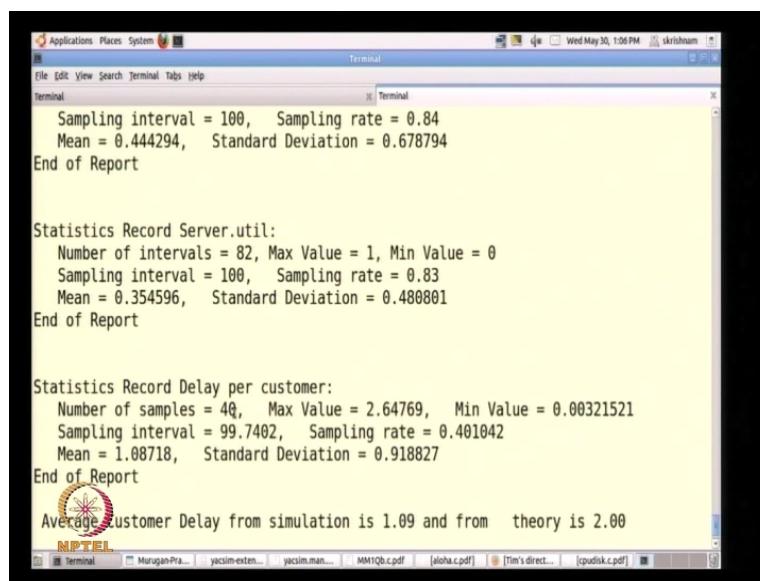
Statistics Record Delay per customer:
```

So, first we print at the length right, and so the number of samples taken so far that is this the mean length, let us look at the mean length of the queue are is 0.444294, that is something that the action measure for us. And the mean utilization of system is 0.354596, this is the mean utilization of the server. Now, we have to look at these two numbers, and say does this relate to what we know right, is what is utilization defined as in mm 1 q? Ratio, of arrival rate

to departure rate right λ by μ , so λ is here is 0.5 μ is 1.0, but the mean reported by the statistics server utilization only 0.354596.

So, what is wrong there, I should be getting 0.5 right, 0.5 should be my row value utilization value, I am only getting 0.35, so I would agree; because I am not run the stimulation long enough, it is only 100 units of time number of samples collected is very, very small. Now, how many samples do you thing would I been collected in terms of customer completions for this given set of parameters, how many customers would have completed service approximately? It is arrival rate 0.5 into this is what number of arrivals from per unit time, this is the time **defends** the stimulation run the stimulation for. Therefore, should be about 50 packet right, 50 customer should have been so this stat record delay per customer this is what we update every time at customer has finished service right, so hopefully that will be around 50 number of samples is only 40.

(Refer Slide Time: 05:06)



```
Applications Places System
Terminal
File Edit View Search Terminal Tabs Help
Terminal
Sampling interval = 100, Sampling rate = 0.84
Mean = 0.444294, Standard Deviation = 0.678794
End of Report

Statistics Record Server.util:
Number of intervals = 82, Max Value = 1, Min Value = 0
Sampling interval = 100, Sampling rate = 0.83
Mean = 0.354596, Standard Deviation = 0.480801
End of Report

Statistics Record Delay per customer:
Number of samples = 40, Max Value = 2.64769, Min Value = 0.00321521
Sampling interval = 99.7402, Sampling rate = 0.401042
Mean = 1.08718, Standard Deviation = 0.918827
End of Report

Average Customer Delay from simulation is 1.09 and from theory is 2.00
```

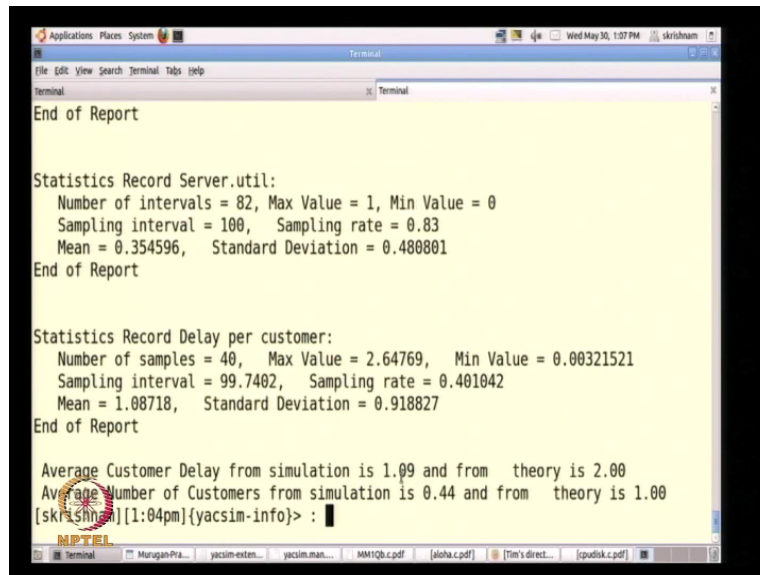
Again why, why is it not 50?

(())

Because yeah this is number of thing is this is again remember that this is not uniform process, it is a random this is exponential process for inter arrival time. Therefore, if you look at only 100 units of time it is not likely, likely not see 50 samples right, only I very long

periods of time will you be able to see that you know point the number of customer equal exactly 0.5 into this stimulation time unit ((C)).

(Refer Slide Time: 05:37)



```

End of Report

Statistics Record Server.util:
  Number of intervals = 82, Max Value = 1, Min Value = 0
  Sampling interval = 100, Sampling rate = 0.83
  Mean = 0.354596, Standard Deviation = 0.480801
End of Report

Statistics Record Delay per customer:
  Number of samples = 40, Max Value = 2.64769, Min Value = 0.00321521
  Sampling interval = 99.7402, Sampling rate = 0.401042
  Mean = 1.08718, Standard Deviation = 0.918827
End of Report

Average Customer Delay from simulation is 1.09 and from theory is 2.00
Average Number of Customers from simulation is 0.44 and from theory is 1.00
[skrishnam][1:04pm]{yacsim-info}> :
  
```

Now, let us look at finally, this important result right, so from stimulation I am finding that the average customer delay is 1.09, whereas theory tells me it is 2 remember right, theory is $\frac{1}{\mu - \lambda}$ the average delay for mm 1 system. So, that is $\frac{1}{1 - 0.5}$ therefore, $\frac{1}{0.5}$ equal to 2, but the customer delay from stimulation is only 1.09, so clearly stimulation is not being run long enough.

Now, for every system it is not possible they get a theoretical value, so what we do is we try to get some **some** approximate values that let us know whether our values are correct or not. Let us say when you write stimulation, right you always have to know what is some closer expected value, why is delay 34 milliseconds, why is not 100 milliseconds, how do you know that numbers are correct that is in important question that we have to ask, before we publish the result right.

In this case thankfully there is something we can compare, so what we see is that the number of customer from stimulation is 0.44, whereas ρ of n rate ρ of n is also from theory it is 1 right, remember ρ of n is λ into ρ of t. So, λ is 0.5 ρ of t is 2 therefore, number of customer should be one and that is also or row by $\frac{1}{1 - \rho}$ which ever you want to look at right. So, therefore this is clearly not adequate so now let us run is work 10000 units of time, and let see what the result are. So far 10000 unit of time how many samples were recorded,

now about 5000 customers right I been compare have completed service that kind of that is as in expected right.

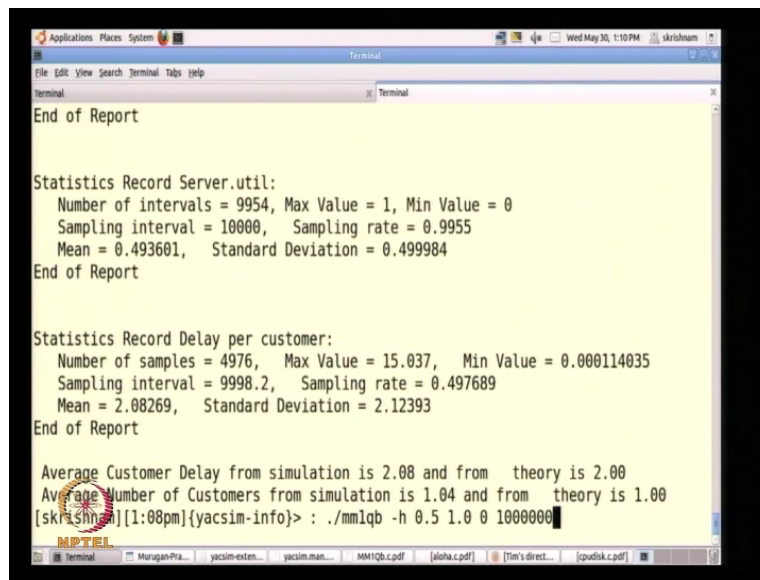
So, we know that our stimulation is kind of it is kind of correct right because we can never know, whether stimulation is correct or not, always there is bug in the system some where you make a mistake, this is the way of validating stimulation, this is very, very important. Many people tend to forget that write an n s 2 t c l script, ran it get the values and printed out just keep going without really checking whether the number actually make sense or not.

So, then look at the delay per customer, delay per customer is quiet varying the mean is what 2 right, but look at the value maximum value 15 in some cases, very long time it is been send. In some cases it is very, very small right 0.00011; which means, because it is exponential in departure time this packet at very small, the service time and there was no other packet in the queue, that is why this is right quiet small 0.00011 and so on.

And so that is so the standard deviation is about to 2.12, so mean is 2 and standard deviation is 2 does it make sense, what is this standard deviation for delay it is also true remember, the distribution for average packet, total delay in the system what is the distribution, you should go back and look James book right. So, delay mean delay, mean packet delay is $1 / (\mu - \lambda)$ and it is also exponential distributor, the packet delay is exponential distributor with parameter $1 / (\mu - \lambda)$.

And then if you look at in the case of in exponential variable is mean and standard deviation equal, in the case of Poisson right, what is v m r equal to 1? In the case of Poisson variance, and mean are equaled right in the case of exponential standard deviation should equal mean. Again this is remainder for you to go back and look at the text book, and make sure that these values are indeed correct.

(Refer Slide Time: 09:02)



```
Applications Places System
Terminal
File Edit View Search Terminal Tabs Help
Terminal
End of Report

Statistics Record Server.util:
  Number of intervals = 9954, Max Value = 1, Min Value = 0
  Sampling interval = 10000, Sampling rate = 0.9955
  Mean = 0.493601, Standard Deviation = 0.499984
End of Report

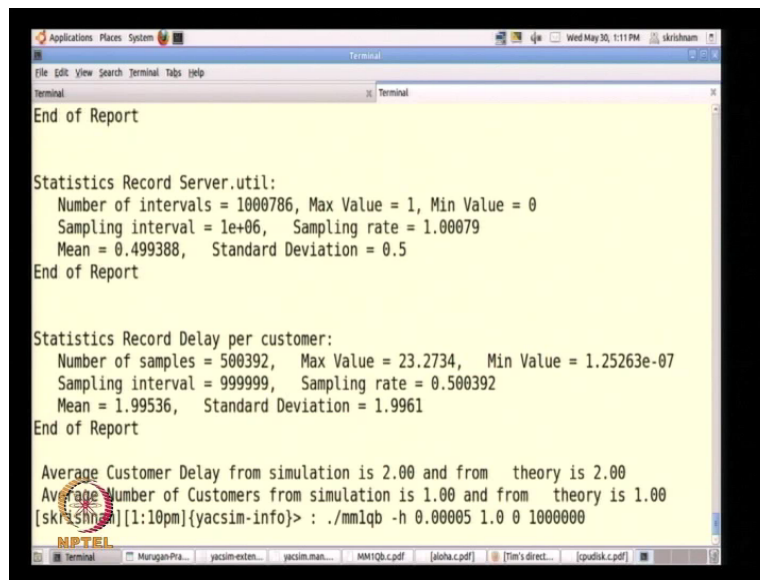
Statistics Record Delay per customer:
  Number of samples = 4976, Max Value = 15.037, Min Value = 0.000114035
  Sampling interval = 9998.2, Sampling rate = 0.497689
  Mean = 2.08269, Standard Deviation = 2.12393
End of Report

Average Customer Delay from simulation is 2.08 and from theory is 2.00
Average Number of Customers from simulation is 1.04 and from theory is 1.00
[skrishnam][1:08pm]{yacsim-info}> : ./mmlqb -h 0.5 1.0 0 1000000
```

So, now you run it for 10000 time units, now we going to get, and it for say around 10 million, 1 million of time units, now might take a little bit longer to finish, but it finish pretty quickly. Now what do I see there right, so delay from stimulation is 2 and there from theory is 2.00, they are know perfectly matching right. And again we are able to get million samples right sorry 500000 samples, which is as expected, so therefore for this particular scenario this was adequate right.

We got enough samples and therefore, we are finding that there is a good correlation between what theory tells us and what is stimulation tells us. So I think whatever you have done is probably, most likely correct in the system. And therefore, this value for 100000, so what peoples do is now can I use this same values of the 100000, even if my lambda is this small (Refer Slide Time: 09:46)

(Refer Slide Time: 10:04)



```
Applications Places System
Terminal
File Edit View Search Terminal Tabs Help
Terminal
End of Report

Statistics Record Server.util:
  Number of intervals = 1000786, Max Value = 1, Min Value = 0
  Sampling interval = 1e+06, Sampling rate = 1.00079
  Mean = 0.499388, Standard Deviation = 0.5
End of Report

Statistics Record Delay per customer:
  Number of samples = 500392, Max Value = 23.2734, Min Value = 1.25263e-07
  Sampling interval = 999999, Sampling rate = 0.500392
  Mean = 1.99536, Standard Deviation = 1.9961
End of Report

Average Customer Delay from simulation is 2.00 and from theory is 2.00
Average Number of Customers from simulation is 1.00 and from theory is 1.00
[skrishnam][1:10pm]{yacsim-info}> : ./mmlqb -h 0.00005 1.0 0 1000000
```

Now, if I change my lambda to small value and run it for 100000 this year 1 million time units, which will how long will it take compare to lambda equal to 0.5 and 1 million time units and lambda equal this 0.00005 1 million time units. In terms of stimulation run time **right**, which will be faster, which will finish sooner?

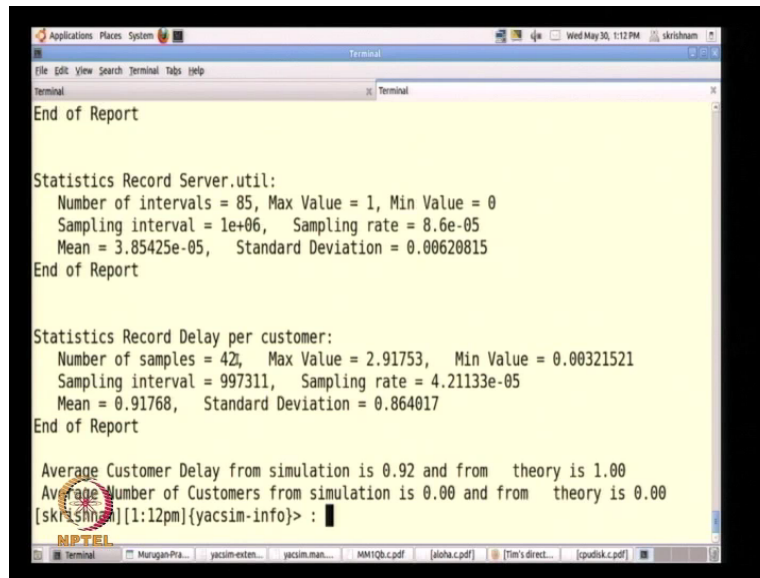
(())

The lower delay why?

(())

Because, the number of event generated it is very small **right**, remember that the time for discrete event stimulation to run is not really proportional to the time, that max time that value you are setting, it is proportional to the number of events that are being generated. So, when I set my lambda value very low which means that the number of events, the number of packet generator is very, very small **right**, and we will try to see where after that.

(Refer Slide Time: 10:45)



```
End of Report

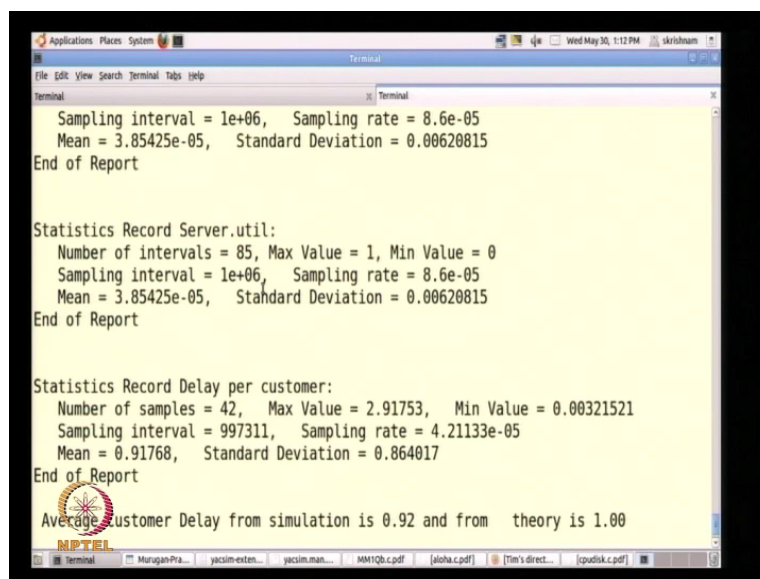
Statistics Record Server.util:
  Number of intervals = 85, Max Value = 1, Min Value = 0
  Sampling interval = 1e+06, Sampling rate = 8.6e-05
  Mean = 3.85425e-05, Standard Deviation = 0.00620815
End of Report

Statistics Record Delay per customer:
  Number of samples = 42, Max Value = 2.91753, Min Value = 0.00321521
  Sampling interval = 997311, Sampling rate = 4.21133e-05
  Mean = 0.91768, Standard Deviation = 0.864017
End of Report

Average Customer Delay from simulation is 0.92 and from theory is 1.00
Average Number of Customers from simulation is 0.00 and from theory is 0.00
[skrishnam] 1:12pm {yacsim-info}> :
```

So, how many packets are generated, only 42 packets is generated right, because that that is therefore, the number of packets are very small therefore, the number evens to be handled also very, very small. Inclusion time of the code will be very small in that case right. But now is this what do we see now, average customer delay from stimulation is 0.92 number of customer from stimulation 0.0, it is very, very small right. Because the length of the queue is very, very negligible right, mean is 3.85 into 10 to power minus 5 that is what showing as 0 here, but it is actually a negligible value (Refer Slide Time: 11:14).

(Refer Slide Time: 11:17)



```
Sampling interval = 1e+06, Sampling rate = 8.6e-05
Mean = 3.85425e-05, Standard Deviation = 0.00620815
End of Report

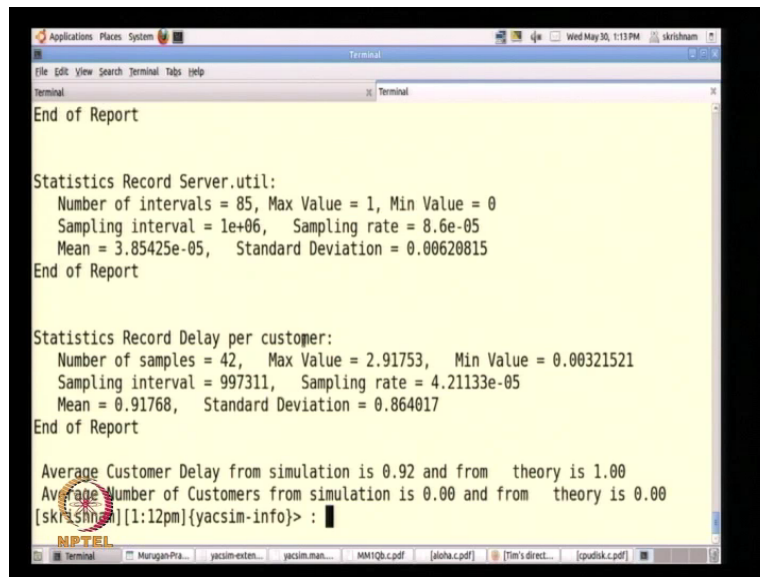
Statistics Record Server.util:
  Number of intervals = 85, Max Value = 1, Min Value = 0
  Sampling interval = 1e+06, Sampling rate = 8.6e-05
  Mean = 3.85425e-05, Stahdard Deviation = 0.00620815
End of Report

Statistics Record Delay per customer:
  Number of samples = 42, Max Value = 2.91753, Min Value = 0.00321521
  Sampling interval = 997311, Sampling rate = 4.21133e-05
  Mean = 0.91768, Standard Deviation = 0.864017
End of Report

Average Customer Delay from simulation is 0.92 and from theory is 1.00
```

We that is what you will expect, right if you are λ is so small, then what will be the mean right, will be row by 1 minus row it will be very, very small right, it will be close to 0 almost. Because row is close to 0 in this case right, it will be closed 0 by 1 and therefore, it is going to be close to 0.

(Refer Slide Time: 11:32)



```

Applications Places System
Terminal
File Edit View Search Terminal Tabs Help
Terminal
End of Report

Statistics Record Server.util:
  Number of intervals = 85, Max Value = 1, Min Value = 0
  Sampling interval = 1e+06, Sampling rate = 8.6e-05
  Mean = 3.85425e-05, Standard Deviation = 0.00620815
End of Report

Statistics Record Delay per customer:
  Number of samples = 42, Max Value = 2.91753, Min Value = 0.00321521
  Sampling interval = 997311, Sampling rate = 4.21133e-05
  Mean = 0.91768, Standard Deviation = 0.864017
End of Report

Average Customer Delay from simulation is 0.92 and from theory is 1.00
Average Number of Customers from simulation is 0.00 and from theory is 0.00
[skrishna][1:12pm]{yacsim-info}> :
  
```

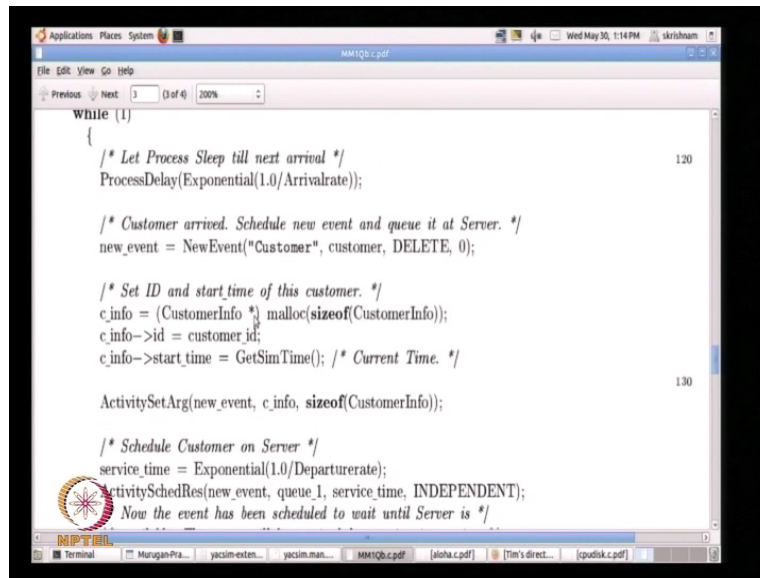
So, that is the way that you, so now this is just our right basics of the system, so we are now what you have not done is I am not turned on all the other option, in terms of debugging anything like that. Debugging tracing is non-trivial in such an environment because of the multiple process is lot of parallel process involved here; and we involve process synchronization is going to be even harder to the **(O)**, so that is the our basic of the system now.

So, question on these

(O)

No questions, so now let us look at the code and see where one could make changes, so I would suggest right, that you modify some of these things for examples right. So, we said that I can replace the arrival process, how if you want to replace the arrival process with something else, what will I do, where will I change that, where will I change the arrival process.

(Refer Slide Time: 12:28)



```
while (1)
{
    /* Let Process Sleep till next arrival */
    ProcessDelay(Exponential(1.0/Arrivalrate));

    /* Customer arrived. Schedule new event and queue it at Server. */
    new_event = NewEvent("Customer", customer, DELETE, 0);

    /* Set ID and start_time of this customer. */
    c_info = (CustomerInfo *) malloc(sizeof(CustomerInfo));
    c_info->id = customer_id;
    c_info->start_time = GetSimTime(); /* Current Time. */

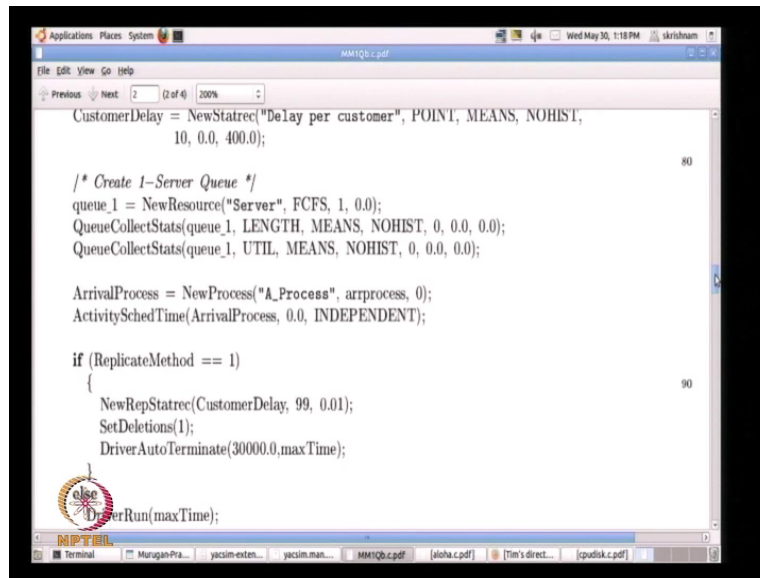
    ActivitySetArg(new_event, c_info, sizeof(CustomerInfo));

    /* Schedule Customer on Server */
    service_time = Exponential(1.0/Departurerate);
    activitySchedRes(new_event, queue_1, service_time, INDEPENDENT);
    Now the event has been scheduled to wait until Server is
}
```

So, this delay right, I can play with the inter arrival time, I can replace this process with some other **(())** say random, random between 0. And say 50 or if I want to make it uniform I simply use the constant value, if you want generate packet every 10 units of time, I simply say generate every 10 units of time right. This can be the constant value, this can be some random value, if you look at the action there are other kinds of random function also defined, you can use any others particular distribution that you want, so this is very flexible you can change the arrival process very simply.

If you want to change the departure process where will I change that, instead of if I want g g 1 right, I can again replace with any other, all I want is random number for this service time. And for service time exponential is very clear, I have pre-defined; if want something else I can even define my own distribution for the packet service time and pick from the distribution also right. So, therefore, this is where I would go on change it, so I can change the mm 1 to g g 1 system very quickly, now how do I change this to mm m system, you want make this multiple server system how will I do that, where will I change that.

(Refer Slide Time: 13:50)



```
CustomerDelay = NewStatrec("Delay per customer", POINT, MEANS, NOHIST,
10, 0.0, 400.0);

/* Create 1-Server Queue */
queue_1 = NewResource("Server", FCFS, 1, 0.0);
QueueCollectStats(queue_1, LENGTH, MEANS, NOHIST, 0, 0.0, 0.0);
QueueCollectStats(queue_1, UTIL, MEANS, NOHIST, 0, 0.0, 0.0);

ArrivalProcess = NewProcess("A_Process", arrprocess, 0);
ActivitySchedTime(ArrivalProcess, 0.0, INDEPENDENT);

if (ReplicateMethod == 1)
{
    NewRepStatrec(CustomerDelay, 99, 0.01);
    SetDeletions(1);
    DriverAutoTerminate(30000.0, maxTime);
}
else
{
    ServerRun(maxTime);
}
```

Let us go to user main, somewhere here the answer is somewhere here.

(())

So, new resources this specify 1 server, I change is to 5 I have a 5 servers, and I was change scheduling discipline I can change it here right, I can make this round robin and it actually look at what the service right. So, you can change to g g m what about that you want, so that is very flexible we can change that there. Now, what about we looked m m 1 v right this is the m m 1 system right, is this finite queue, infinite queue, this is infinite queue system.

Because, we do not really specify right, the packet array is simply that queued, but what if you want stimulate the finite buffer queued then what will have to do, so then you will have to go and write your own server. This case I am using the system server, the actual sever which is by default giving me infinite capacity. If I want to replace that with finite capacity, I have to go to on make though changes, I can basically take this thing, and essentially do m m m b system quiet easily right in that is the flexible enough for as too. So you will find on the web right, several programs that says just give me the parameters for lambda mu and I give what the values are and most of the time they use theoretical values or just stimulation like is to that you know right.

And good thing about stimulation you can actually not just look at the mean values right, they you can actually plot for every packet what is that happening right you can plot the

distribution of the packet delay, distribution of the arrival times. So you get a lot more wealth of information you can dig deeper into system behavior with the help of the code like this. You can introduce, variable all over the place where as the in the case of analysis is mathematical model, you cannot you do not have anything else right. Therefore, of course derive everything by the $(())$, but not everything can be derived, especially when come to the $(())$ system right.

So, that is the advantage of the trying to use stimulation, but it has to be again, it is very useful tool just to be used very carefully. And one should always suspect there are bugs in the stimulation right, never be over confident of your stimulation make sure that every value that you are getting is indeed validated right, enough samples are being generated, otherwise you might end up giving everything a million time unit what we saw (Audio not clear 15:50 to 16:05)

Students are where in near yeah (No audio from 16:13 to 16:28). I have not tried that new resources how to make that finite $(())$, but it is easy if you look at the next example aloha $(())$ where actually I use in finite buffer, infinite buffer.

$(())$

Yeah, you can write n queue de queue and only thing is you have to write your own scheduling discipline also in those things not available. So, that is as far as m m l q which is generic performance evaluation any system will try to do that. Let us look at another program that another protocols that we have seen in class right.

We saw this aloha r we have done the analysis's of aloha right the two different ways of analysis's that you are seen in class. So, let us look at that, we all know saw aloha program work straight other protocols the packet arrives in the middle of the slot you simply have $(())$ wait until the beginning of the next slot right that is difference aloha and slotted aloha.

So, therefore, now we have the notion of the slotted time right, there is things happen packet transmission will happen only at beginning of time slot packet, can arrive at any time, but you only take for it the service of the beginning of a time slot. So, this how definition of the aloha protocols. So now I have to emulate, in class we saw different ways of doing right, that m n and we derived from basic g equal to g in to g minus g e to power to the g all those theoretical

formulae we derived. But let us look at, whether we can actually verify the help of this simple stimulation.

So, now to module this aloha stimulator we need to create our own definition of a node, I have to somehow emulate channel also right. So, there are two entries are there one is node other is channel, and the node I can replicate as many as I want 100 nodes, 500 nodes, 50 nodes right once I create one basic node structure, node object I can replicate that to the number extend that is needed, that is a node. And the channel also I have to somehow represent in this particular system.

So, the now the important part of the system is how do you model channel, how do you model the channel in a star aloha system, so we talk about states right, we talked about Marko model, we talked about different states a system can be in. So, what is the different states that a channel can be in. In aloha, in any system in any communication channel right, this state the system the channel can be some few states right, what are those possible states, either is busy or it is idle, that is all.

The channel is idle or channel is busy, but within that busy we can also classify one level further by saying that there is collision busy, which mean more than one user is sending that means a packet is lost or it is generic or there is only one customer, the one user that is currently transmitting right. So, there are three states basically where there is no collision, but transmission, successful transmission going on is or second there is a collision is going on or third the channel itself is idle, so we have to essentially capture the state of the channel in a given slot and this happens every slot.

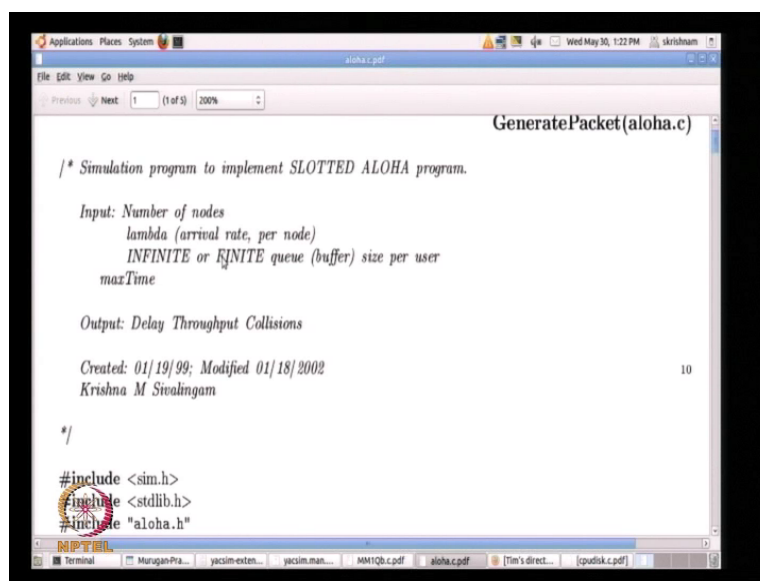
So, if you look at the system behavior, we can just look at the beginning of every slot, at this slot beginning of the slot it see idle or it is successful slot or it is collision and therefore, wasted slot, that is channel aspect. Then again will looked from the node perspective what are the sates node can be in. So, what are states node can be in. Remember we did one slotted aloha I S I model that we saw, we modeled the different states right (()) if you recollect those who are in the class right.

So, there was in the different states right what are the different states, so the user is idle right. And therefore no packet to send or the user is transmitting in particular slot or the user is in and the user need not, listen it assume there it is always on, in this case the user is in back off right. So, what you do is you try to in a given slot either idle, because no packet to send, there

is a definition of aloha $(())$, whenever there is a packet is in you send it next slot first time, but is the rest collision you back off.

So, if you go and back off state that means, you are not going to send the packet right away, but you have a random back off right, that is definition idle or you are sending in a given channel or you had a collision and therefore, you are in a back of state. So these are the three basic states for a particular end user. So this is how you define your system. Now, you have to essentially represents this with the help of your various data structure. So, the parameter is this the number nodes n and arrival rates λ arrival rate per node, this is only two thing we need right. Remember, that is the node on the system each user is generating say 0.1 packets per second. There are 10 user and therefore, is load in the system is basically 10 into 0.1, so there is 1.

(Refer Slide Time: 21:04)



```
/* Simulation program to implement SLOTTED ALOHA program.

Input: Number of nodes
       lambda (arrival rate, per node)
       INFINITE or FINITE queue (buffer) size per user
       maxTime

Output: Delay Throughput Collisions

Created: 01/19/99; Modified 01/18/2002
Krishna M Sivalingam

*/

#include <sim.h>
#include <stdlib.h>
#include "aloha.h"
```

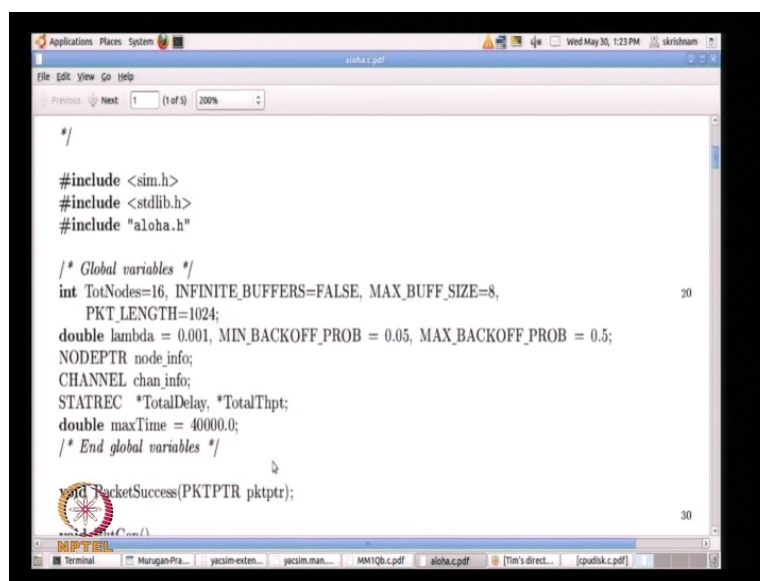
And whether, we have infinite queue or finite queue that is the very small change, we will see how we can do the finite queue business now. **I will...** So, this is the some of the basic global variables total number of nodes is default to 16, infinite buffers size is defaulting to false, I have an upper limit on the buffer size in case it is non infinite; this is the b value that we have seen it $m \cdot b$ packet length is again fixed to be some number 1024, whatever that is.

And default λ this is the arrival rate is this one, when there is a notion back off probability right, so every node has a back off probability that, it will keep updating. If you remember that exponential back off and things like that, so you will choose your next slot

based on this back off probability. So, the back off probability 0.1 and you have a packet to send, then with probability 0.1 you will resend the packet, otherwise 0.1 is back off and sit quietly, that is your back off probability definition.

And then there are node pointer which will contain information about each node, channel will contain information about the channel state, the different state that will we talked about. And there are two different statistic records for calculating delay as well as through put of the given system. So, these are the basic definition in terms of variable in the system.

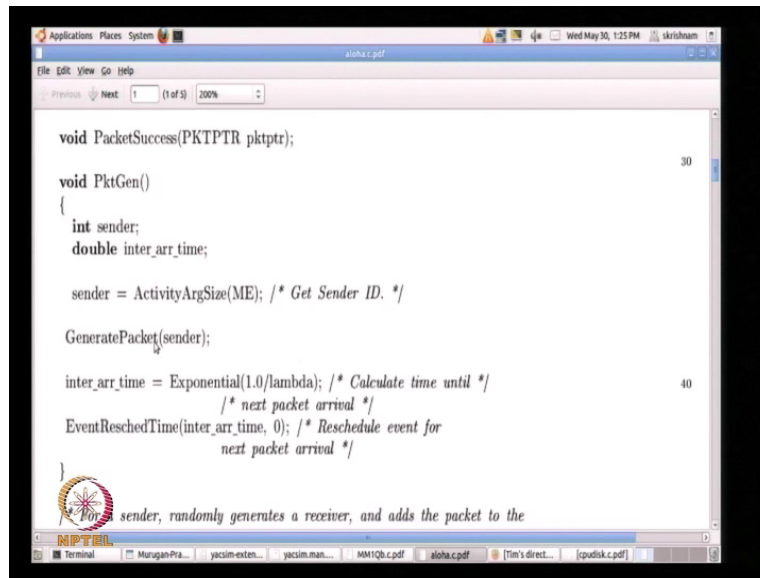
(Refer Slide Time: 22:15)



```
*/  
  
#include <sim.h>  
#include <stdlib.h>  
#include "aloha.h"  
  
/* Global variables */  
int TotNodes=16, INFINITE_BUFFERS=FALSE, MAX_BUFF_SIZE=8, 20  
    PKT_LENGTH=1024;  
double lambda = 0.001, MIN_BACKOFF_PROB = 0.05, MAX_BACKOFF_PROB = 0.5;  
NODEPTR node_info;  
CHANNEL chan_info;  
STATREC *TotalDelay, *TotalThpt;  
double maxTime = 40000.0;  
/* End global variables */  
  
void PacketSuccess(PKTPTR pktptr); 30
```

And we will, now look at some definition, so packet generation similar to the arrival this is the again helper function, this will be this is like an even handler, so whenever the packet is generated this event is going to get called.

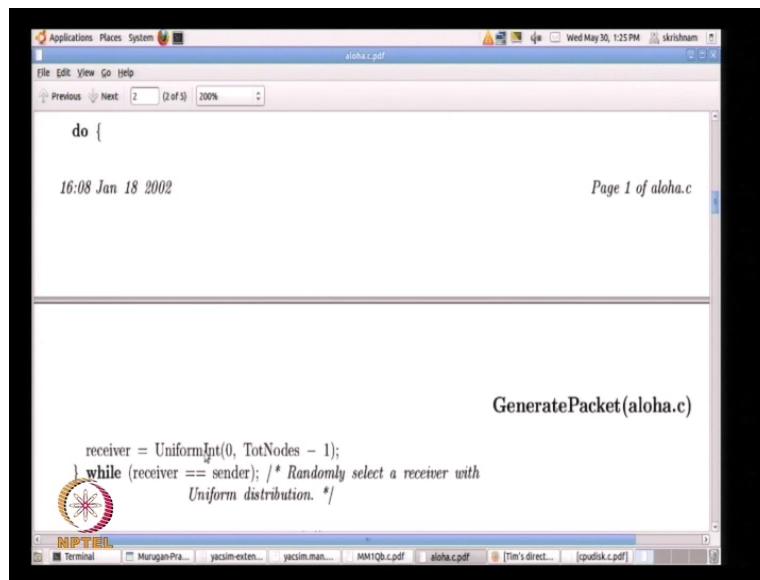
(Refer Slide Time: 22:35)



So, this is very similar to what we saw before, it simply calls this function called generate sender, and then it determines the time that the next packet is going to get generated, again it is based on the exponential distribution. This is not based on process delay, **this** look at the differential, where I use the process, in the previous example for arrival process. Now, I am stimulating the arrivals as the set of events, so I start with one event per node that is the first arrival. And then subsequently it reschedules the same event over and over again, but the time between two events getting rescheduled is simply the inter arrival time and that in this cases exponential distributed right, so that is what we have here.

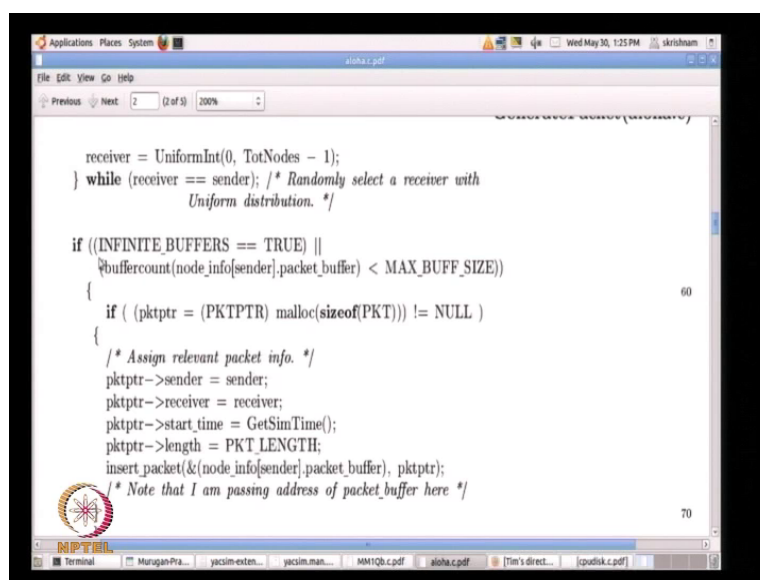
So therefore, assume that in the beginning of the system node, I have do this for every node right there are 10 node, every node has an independent packet generation process, packet generation sequence. So, therefore, we will create initially 10 events if there are 10 nodes and therefore, each event each node will then have series of events, every time this event is triggered a packet is going to get added to the particular queue, that was the system is that is the basic packet generation. Generate packet this where we will, so this is generate packet for a particular sender.

(Refer Slide Time: 23:53)



So, what we do is, we need first determine the receiver for this particular packet. Remember that this is now communication, so every sender has to find, has to tell whether both the intended receiver arrays. And the intended receiver in aloha system is does not matter, because it is a broadcast system, but we just we do a receiver for particular reason this is the aloha with multiple channel also it support right, so we will not worry about that right now.

(Refer Slide Time: 24:14)



So, then we look at this infinite buffer, infinite buffer is true mean I can simply go and write insert the packets; so this is the small routine that I have written is part of adding packet to

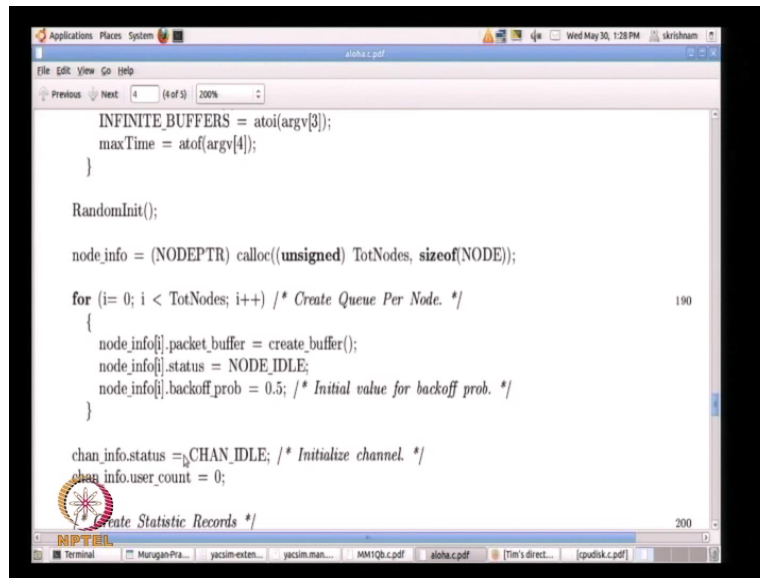
the given queue. So, every node right has the corresponding packet buffer association, there is simply `(())` that is all. And simply keep adding on to the linked list, and then I will do insert delete from that queue as when needed, this is how my buffer is now. I explicitly have a buffer for every sender in this particular scenario.

So, therefore, I simply create the packet figure out who the sender is and then I create the all the information, so the packet length is updated, the packet arrival time is updated, sender, receiver, all of this is stored in the packet and this is intern stored in the sender packet buffer `(())` right. So, the packet arrives I put in the sender buffer, ultimately that is what has happened. So, then as and when packet arrives they keep on getting added to this queue, this is the packet generation process, and this is done for every sender.

So, there are 10 senders means, there are 10 separate threads which are generating packet and the packets keep going to the corresponding senders queue. There are 10 sender queues and every sender queue has its own set of packets generated by that particular node. So, that is what we have here, now the transmit packet, so let us before we come to the transmitting the packet let us user jump to use main that is tell us sequence up what is going on.

So, what is happening in user main, in user main let us `(())` the system start right there are 10 nodes, 15 nodes, whatever it is, for every node, I create the buffer and I set this status to be idle initially. And define some default back off probability, ignore that for time being every node has some back off probability that keeps updating right. All we need to know is for every node, I started being idle and I create the buffer for that node, buffer is of course initially empty. And then the channel status is simply set to idle channel initially.

(Refer Slide Time: 26:12)



```
INFINITE_BUFFERS = atoi(argv[3]);
maxTime = atoi(argv[4]);
}

RandomInit();

node_info = (NODEPTR) calloc((unsigned) TotNodes, sizeof(NODE));

for (i= 0; i < TotNodes; i++) /* Create Queue Per Node. */
{
    node_info[i].packet_buffer = create_buffer();
    node_info[i].status = NODE_IDLE;
    node_info[i].backoff_prob = 0.5; /* Initial value for backoff prob. */
}

chan_info.status = CHAN_IDLE; /* Initialize channel. */
chan_info.user_count = 0;

/* Create Statistic Records */
```

And in a given slot I keep track of the number of users who try to use that channel, who try to send on that channel, and that is what this user count variable is going to be used. This is updated every slot, initially it is set to 0 that is all. I have created my set of nodes and I created the channel information, that is the basic thing. Now, let us try to look at this set of all these statistic records we are going to skip for the time being.

Now, I have to create the first event right so now, look at this loop here, have this loop running for the 0 to all the nodes in the system. And for each node I am creating a packet generation event, so this is the event that is created, and then I am asking it to be associated with packet gen, this is the function that we saw before. So, every node has an event associated which will simply execute that packet gen.

And because there are 10 such or 50 such nodes in the system, I have to somehow encode right when an event is triggered, I need to know whether this is sender for packet or this is the event for packet **sorry** event for sender number 5 or sender number 10. And thing like that; that encoding is done with the help of this set argument which I am skipping for now; it is very easy to encode that a given event belongs to sender number 6, sender number 7 and so on. So, that is very straight forward.

Then again I schedule this event and remember first time I am scheduling it, and the rescheduling takes place inside packet gen itself. And rescheduling is done based on the next arrival time for the particular sender, this is the sequence that is going on. So now you created

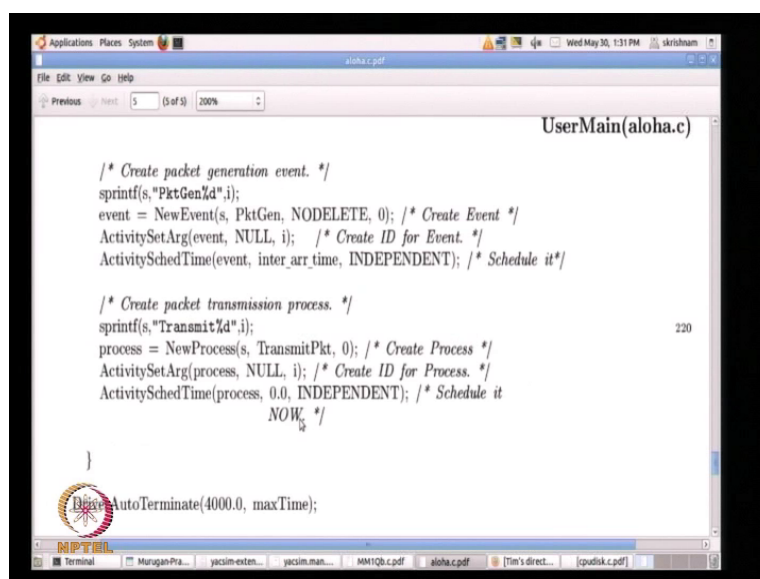
all the packet generation event for all the nodes in the system. And there is also and that is the one that adds packet to the queue I also have to have in other process or other event in this case we are making it a process right. So, for every sender I am also creating a transmit packet process this is actual transmission process.

So, there are packets coming in from the layer to this queue I need to have a process that looks at this queue every time unit process the packet and sends it out right; therefore, I need the process for that. Here, I am using a combination of both event as well as process. So, the packet transmission process is now, equivalent to the server resource that we saw before. Now, I am explicitly coding the queuing and dequeuing in this particular case.

So, there is a transmit process that is associated with every sender every node in the system. And this job is take the top of queue try to send the packet, if it succeeds good, if it does not succeed wait and until success, re transmit later on all these thing done by this transmission packets. So, that is what this is want know for every process of every node I create a new process right; and then encode the process id. So that we can use it for some point time and this is simply scheduled once that is all.

So, this will now been actually we will look at the code for this will be an endless loop right; the while loop will be there and transfer ever, but that is all are try to do. So, these are the two main things in every node I am encoding the addition packet generation process, and the packet transmission process this is all that basic system that we need to do.

(Refer Slide Time: 26:49)



```
/* Create packet generation event. */
sprintf(s, "PktGen%d", i);
event = NewEvent(s, PktGen, NODELETE, 0); /* Create Event */
ActivitySetArg(event, NULL, i); /* Create ID for Event. */
ActivitySchedTime(event, inter_arr_time, INDEPENDENT); /* Schedule it */

/* Create packet transmission process. */
sprintf(s, "Transmit%d", i);
process = NewProcess(s, TransmitPkt, 0); /* Create Process */
ActivitySetArg(process, NULL, i); /* Create ID for Process. */
ActivitySchedTime(process, 0.0, INDEPENDENT); /* Schedule it
NOW */

}

AutoTerminate(4000.0, maxTime);
```

And I am using this driver auto terminate right which is we will not get into this will again if convergence occur sooner, it will stop the stimulation earlier otherwise, it is simply run in to max time. And when the stimulation finishes, at the end of it, I will look at that mean total delay right, per packet and the channel throughput also measure here, and how long this system actually run. This is what at the end of the system that is all; I care about right what is the average delay, what is the average throughput, what is the average throughput defined as. The number of packets, success full packets transmitted per unit time that is the definition of throughput in this particular system.

So, questions generation transmission and the basic recording that is all. Now, what should be the logic that every node should execute, in every slot, what should be the logic before you look at the code that is utilizes the logic. So, the transmit process we look at the queue, if the queue is empty nothing that sender has no job to particular slot. If the queue is not empty, it will simply try to send this packet. Now, remember this happening in parallel if there are 5 senders, each sender will at beginning of that slot try to send their packet at same point of time.

So, somehow I need the mechanism to keep track and these thing will all happen at the same, let us say current stimulation time is 15.0, beginning of the slot, 15.04 nodes have packet to send all of them will at executed same stimulation time in some sequence. It is still a uniprocessor system. But in some sequence, node 5 will try first, node 10, node 11, node 12; but they are all actually executing at same stimulation time.

Sequentially it might happen in different points of real time, but stimulation time wise their all happening it is same instant of time. So, now, what I can do is, I can simply update this user count that number of users accessing the channel right, can simply be updated. So, what happen is it is 0 at beginning of the slot, the first user come it will increment the counter to one, the second user tries in the same slot, the second user the packet has sent, tries to transmit, it will increment the counter to 2, 3, 4 and so on.

So, collision is defined as, under what condition there is collision, when ever user counter is more than 1 and user count equals to 1 this packet has been successful. Now, when will I know that I will know that as soon, as if say 15.0 I did all the attribution I get 15.1 if I look at stimulation if I look at the system rate state at 15.1, 15.2 whatever it is, some intermediate point I can look at the system state on this channel user count. So, I look at this every node

what will do is it first we will go and update this variables at that 15.0 then I am arbitrary defined at 15.5, half way into the slot; these guys will come back and look at the user count.

The user count is greater than 1, it means that somebody else also attempted and therefore, the collision has taken place. That is a way I am defining the collision and I am detecting collision. That is not how system actually behaves, the system will actually, try to listen whether there is the collision or **the user is** receiver will simply have a collision will not able to get the packet at all, but we cannot do it in our simulation.

So, we are trying to emulate, what is going on by looking at the number of users that access the channel at the given point. So, that was the logic is all about. So basic logic is to equal whatever beginning of the slot attempt to send the packet, half way into the slot try to find out, what the channel count is? If the channel count equals 1 wonderful, packet is successful, if it is greater than one, the packet is not successful then you will have to retransmit the packet. That is the basic logic that we will have. Now, let us look at this transfer packet per slots.

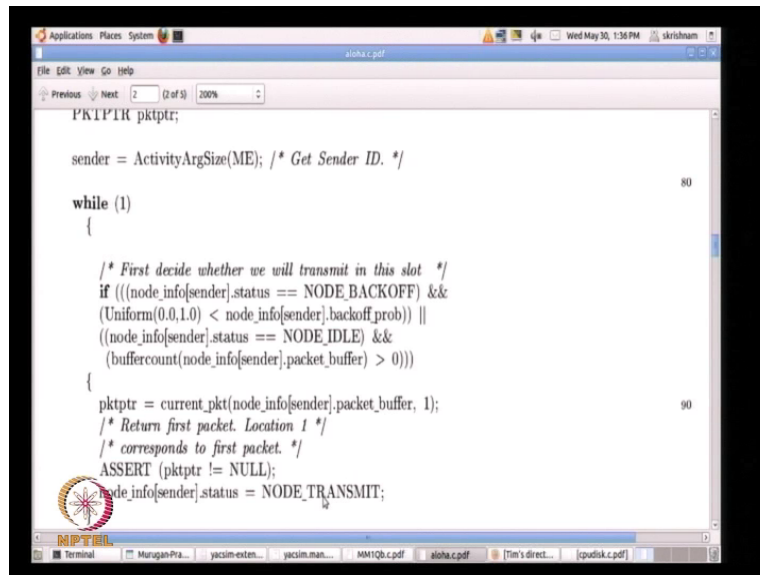
So, little bit to be complicated, because of all these status are going on. So, first is to decide whether a given node will send in a slot, so when will node send in a slot these might be a little bit complicated; but if you dissect this it would not be that hard. If a node currently in back off state which mean it the packet to send, but could not send. And it is no flipping a coin right and let us say this it has back off probability. So, if a flip a coin, back off probability say 0.1 **right** and if I generate the uniform random 0 and 1.0 as long as the uniform number is less than 0.1, it means that I am going to attempt the transmission.

If it is greater than 0.1, I will not attempt transmission that is again **right**. That is the way to defined, this emulating the uniform delay. So, if I am in back off and if I have decided by the state point actually resend then go to transmission state, or if I was idle in previous slot and now, there is packet send the queue this is the two possibilities. Either I had no packet in the queue before and now I have packet, so I have to go to transmit state therefore, I go to transmit that is all.

And I look at first packet in the queue that is the packet I just first packet in the queues pointer that is all we do, packet pointer input this and no info **(())**. And those you know assert know why you using assert you are standard C programming practice I want make sure that I do know have some **(())** buffer code. So, if it is sequential let us me something on. So any

way I have extracted the pointer to the top bracket of the queue and setting myself in to no transfer state.

(Refer Slide Time: 34:50)



```

PKTptr pktptr;

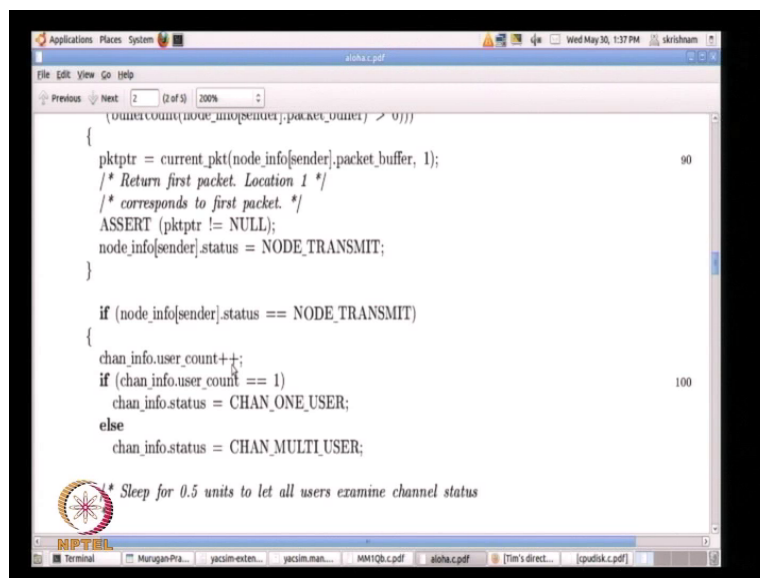
sender = ActivityArgSize(ME); /* Get Sender ID. */

while (1)
{
    /* First decide whether we will transmit in this slot */
    if (((node_info[sender].status == NODE_BACKOFF) &&
        (Uniform(0.0,1.0) < node_info[sender].backoff_prob)) ||
        ((node_info[sender].status == NODE_IDLE) &&
         (buffercount(node_info[sender].packet_buffer) > 0)))
    {
        pktptr = current_pkt(node_info[sender].packet_buffer, 1);
        /* Return first packet. Location 1 */
        /* corresponds to first packet. */
        ASSERT (pktptr != NULL);
        node_info[sender].status = NODE_TRANSMIT;
    }
}

```

Step one, then incase I did not go and no transmit state rest of slot I can simply skip.

(Refer Slide Time: 35:07)



```

        (buffercount(node_info[sender].packet_buffer) > 0)))
    {
        pktptr = current_pkt(node_info[sender].packet_buffer, 1);
        /* Return first packet. Location 1 */
        /* corresponds to first packet. */
        ASSERT (pktptr != NULL);
        node_info[sender].status = NODE_TRANSMIT;
    }

    if (node_info[sender].status == NODE_TRANSMIT)
    {
        chan_info.user_count++;
        if (chan_info.user_count == 1)
            chan_info.status = CHAN_ONE_USER;
        else
            chan_info.status = CHAN_MULTI_USER;

        /* Sleep for 0.5 units to let all users examine channel status

```

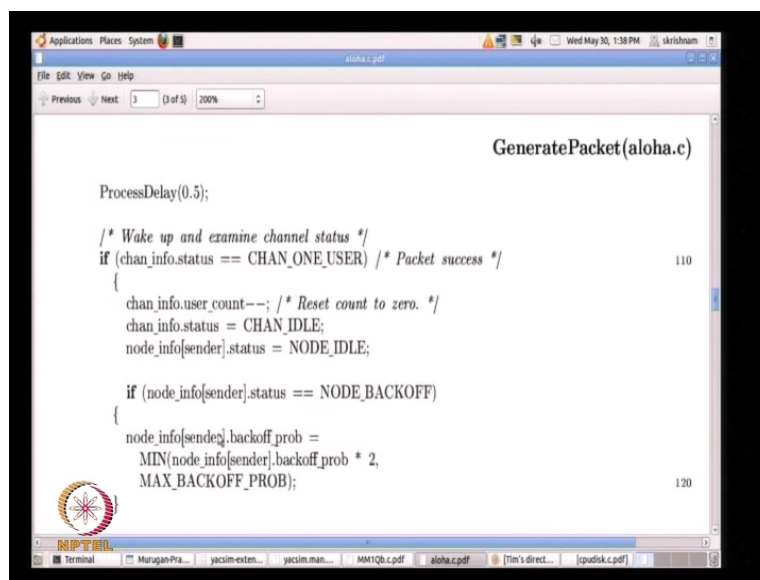
So, incase I am going into no transmission state, so what do I do? I simply increment the user count by 1. Now, here actually I should be using the semaphores for mutual exclusion I am not using that. Because I am sure that at a given point in time only one process will be executing this particular `(())`. To be correct I should be using the semaphores `(())`, let us for

time being ignore that. And if the user count is set to one, then you know because what I do is, when I increment I can always check right if it is already some other user incremented. Remember that there are 4 users, they are all executing at the same stimulation time this is like.

So, if I just said these two as the 2 variable you use this variable later on, then once I increment in the counter, I simply process delay pointer. Now see the process delay is very handy 15.0, I changed the variable user count variable at 15 point to 15.5 I go to sleep why so that other process, can then update this variable based on the logic. So, that is why I go to sleep, so the three other processor waiting to send the packet who are all also in no transmits state can updating this variable.

So, at the end of 0.5 I know that everybody who is wanted to update the variable has had chance to update, there is nothing else that can happen in the system all right. So, then I wake up and I examine in the channel status the channel status means only one user was there excellent, then you reset to 0 to, because next slot I have to make sure that is 0 right then after is what is there some other you know things that the channel is idle so on.

(Refer Slide Time: 36:32)



```
GeneratePacket(aloha.c)

ProcessDelay(0.5);

/* Wake up and examine channel status */
if (chan_info.status == CHAN_ONE_USER) /* Packet success */
{
    chan_info.user_count--; /* Reset count to zero. */
    chan_info.status = CHAN_IDLE;
    node_info[sender].status = NODE_IDLE;

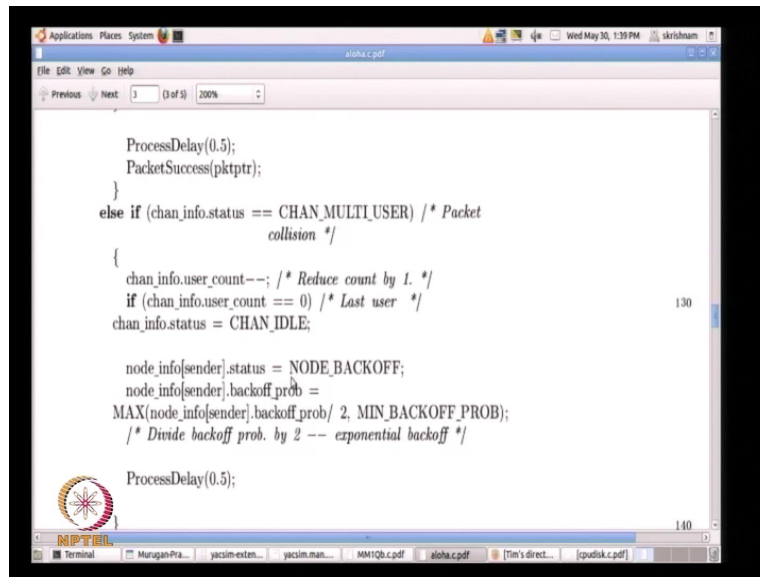
    if (node_info[sender].status == NODE_BACKOFF)
    {
        node_info[sender].backoff_prob =
            MIN(node_info[sender].backoff_prob * 2,
                MAX_BACKOFF_PROB);
    }
}
```

But, basically I have to again delay this particular process which ever process was successful, I delayed for another half unit of time, why? Let us to finish the slot a 15.0 I try to send 15.5 I checked the status, status said I am the only one user; therefore the packet is successful. So, I have to then wait, until the entire slot is finished before I can do all the packets success

processing. What is packet success processing involved? Removing this packet from queue, updating delay variables **right** all those things all the statistics has to be gathered.

So, I simply again if I am the only user right, if it is channel one user I then process delay for 0.5, and then I process the packet success at the point of time this for this particular process.

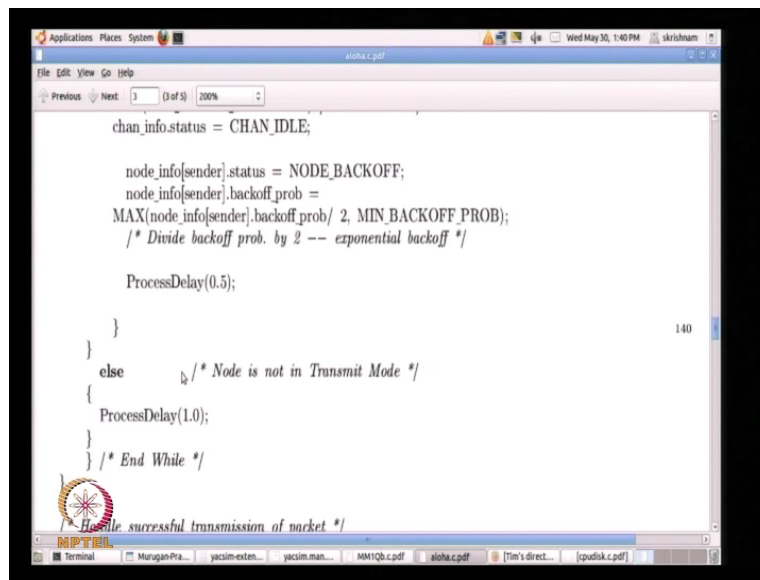
(Refer Slide Time: 37:27)



Otherwise, what we do what nothing else to done you simply set the, if there are more than one user; therefore, collision is take place is simply delay for another 0.5 units of time. And then nothing else for this process, that is all you change the state correspondingly, you would be back off state, you went to transmit state if you go to back off for; previous start up the collision nothing much can be done about the collision state that is all right. So, from node idle you got node transmit from node transmit you would go to node back off, then some point in time when you decide to re transmit you will go from node back off to node transmit.

And that is successful you go back to node idle, if it is not successful, then you keep on being node back off for some **(())**; that is what is node goes through. So, this is what we have done right every slot, we have checked whether the slot is collision or not collision whenever, there is collision I have to invoke the packet success right this is what we try to do.

(Refer Slide Time: 38:44)

A screenshot of a PDF viewer window titled 'aloha.c.pdf'. The window shows a snippet of C code for a node's transmission logic. The code includes comments in Italian and sets delays for successful transmission and non-transmission. The code is as follows:

```
chan_info.status = CHAN_IDLE;

node_info[sender].status = NODE_BACKOFF;
node_info[sender].backoff_prob =
MAX(node_info[sender].backoff_prob/ 2, MIN_BACKOFF_PROB);
/* Divide backoff prob. by 2 -- exponential backoff */

ProcessDelay(0.5);

}

else /* Node is not in Transmit Mode */
{
ProcessDelay(1.0);
} /* End While */

/* successful transmission of packet */
```

The PDF viewer interface includes a menu bar (File, Edit, View, Go, Help), navigation buttons (Previous, Next, 3 of 51, 100%), and a taskbar at the bottom with various application icons.

So, packet success, what is packet success going to do? Packet success will handle all the successful transmission right components of it. Now, remember this is big if else statement, if I am not transmitting in a particular slot then what will I do, I simply have to sleep first, this entire slot. So, I have process delay 1.0 saying nothing much to do in this slot bye **bye** I will wake up in the next slot. So therefore, this transmission process for every node will keep on making up of every one unit of time and then process it.

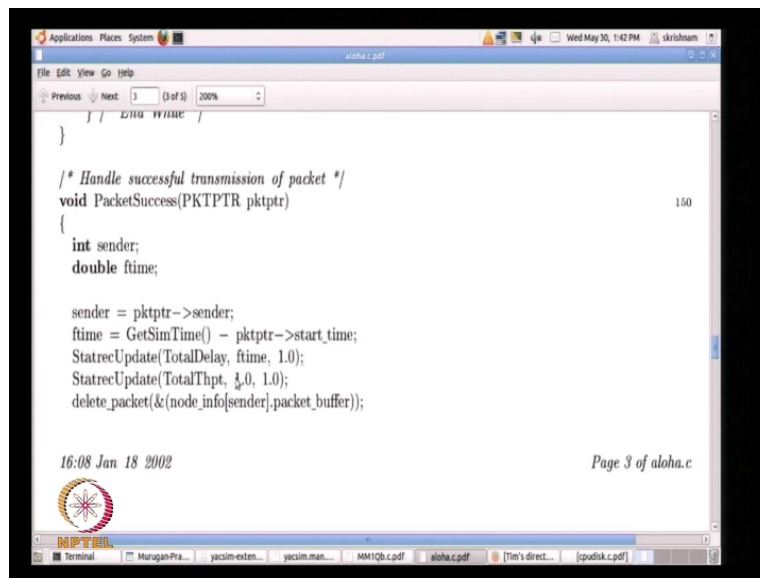
And now actually what happen is, this look very nice, but when you look at it the stimulation I have put 1000 users right, there are 1000 user doing this checking. There are 1000 process running this is not very efficient way of processing, but this very natural way of processing. I simply think about one process for every user and I can modify whatever I want. I could have replaced this multiple process with single process also because I have global variable any way. I would simply look at the status all the users, all these guys have packet to send, therefore, no point there is a collision.

I can simplify this implementation if I want to, but I have just used a little bit, easy to understand implementation, because we have to think of the node from the system, from the system perspective. How does the system behave you want to emulate that, and then you can do optimization saying wait a minute this is too complicated, because of, if you run this work for example, things like **(())** and so on, it takes such a long time **(())** events getting handled there, and you have no way of changing this. So, you cannot say will let me simplify the way

that events are been generated; and therefore, I can no and we cannot really do that, with the action we can do that because you are the one creating every event that you what, so that's what we have done.

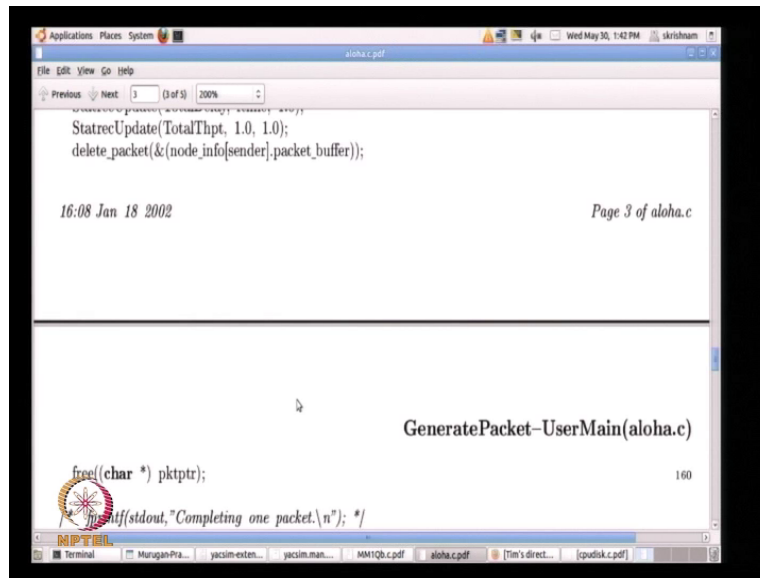
So, then successful event is updating the total delay, so you look at the time, current time minus when the packet start time, there is the time taken for the packet delay that is how long it to service this packet; because this involved the retransmission and everything and so on.

(Refer Slide Time: 40:30)



And every time packet successfully to send, simply update the through put one value, one packet has be sent that is all that is why 1.0 represents, delete the packet from the corresponding sender, and that is pretty much, all the events freeing all that right, so that is what this is the basic logic of the system.

(Refer Slide Time: 40:43)



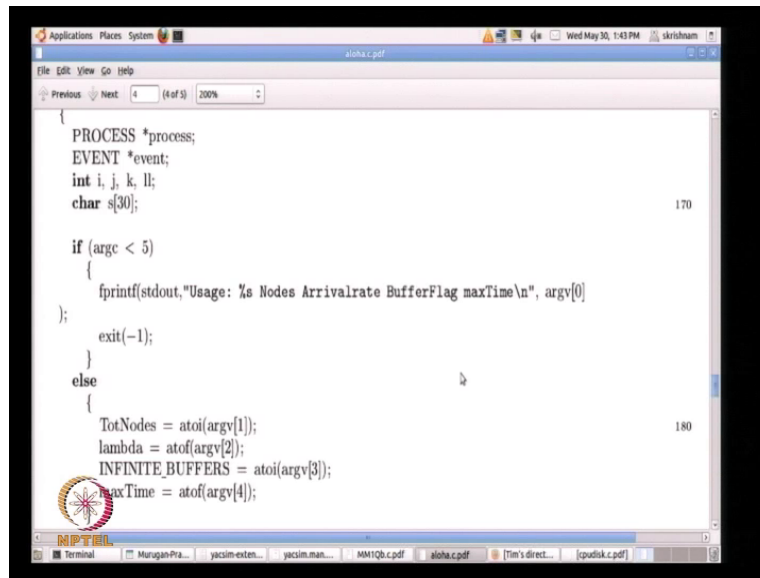
So, even the simple system like aloha requires fair amount of thinking to get the logic, and this also took me several around of getting this logic right. The way I did the back off probability that was not correct, when I did the implementation. And I compared this stimulation to theoretical research, my results were not coming correctly, and why I could figure out why because I told this in class earlier on right, the way back off implementation did not match, what I was using in theory.

I was using uniform random variable there, here I am using a geometric random variable and that will makes difference to your delay values, so that those things were also cropping up. So, you have to make sure that what you are implementing in mathematical model matches reasonably close to what you are trying to do here. So, that is the code part and the rest of it is all straight forward you have seen this **right**, so the logic clear, questions?

(0)

The uniform function part of reaction, the reaction is few default function, the source code I do not here if I look at the source code, you look at the all those function; you can go and add your own variable also if you what that is the way the straight forward delay.

(Refer Slide Time: 41:56)

A screenshot of a PDF viewer window titled 'aloha.c.pdf'. The window shows a C program for simulating the Aloha protocol. The code includes variable declarations for a process, event, integers, and a character array. It features a conditional block that checks the number of arguments and either prints usage information or sets simulation parameters like total nodes, arrival rate, lambda, infinite buffers, and maximum time. The viewer interface includes navigation buttons and a search bar.

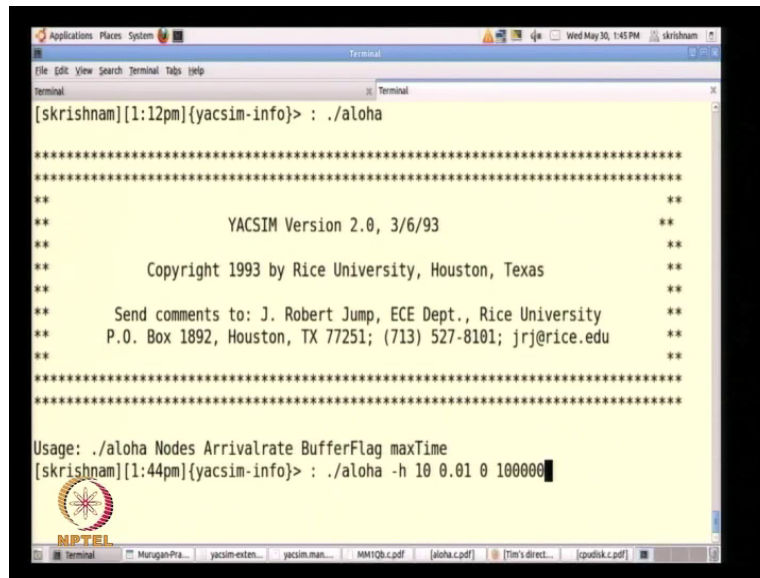
```
{  
    PROCESS *process;  
    EVENT *event;  
    int i, j, k, ll;  
    char s[30];  
  
    if (argc < 5)  
    {  
        fprintf(stdout, "Usage: %s Nodes Arrivalrate BufferFlag maxTime\n", argv[0]);  
    };  
    exit(-1);  
    else  
    {  
        TotNodes = atoi(argv[1]);  
        lambda = atof(argv[2]);  
        INFINITE_BUFFERS = atoi(argv[3]);  
        maxTime = atof(argv[4]);  
    }
```

Other thing is that should be similar to this is people have used I think sim java, which I have not used if you are java expert. The people are also familiar with python also, there is sim 5, which I have not tried, I can look at sim 5 which again, you basically depending on what you want to implement you choose the stimulator. Whether just **bare bones** discrete event stimulation gives the only events and process handling; that is all or if you want something that has all the protocols implemented, then we go for **(())** plus plus and all the packets just come within are **half net** or all the packages that comes with it, right.

But any time you go for large packets scenario you have to realize that many errors and you may not be able to understand that where they happen why they happen. It takes time to figure out changes, where there is much more control over the code definition here. So, there are no questions on the logic part, then let us try to run this. So, these are the number of nodes, so let us say 10 nodes and arrival rate what should be arrival rate be. For single channel aloha what should be maximum arrival rate before system starts collapsing standard aloha; total offered load those should be less than 1 right.

If the total offered load goes to more than 1, then you find it system throughput start s declining. So, let us look at say 0.01 this is arrival rate per node and we will use the infinite buffers sorry yeah I thing just says finite buffers max time I will say it this 10000, 100000 unit of time this is 10 users each generating 0.1 packets per second.

(Refer Slide Time: 42:30)



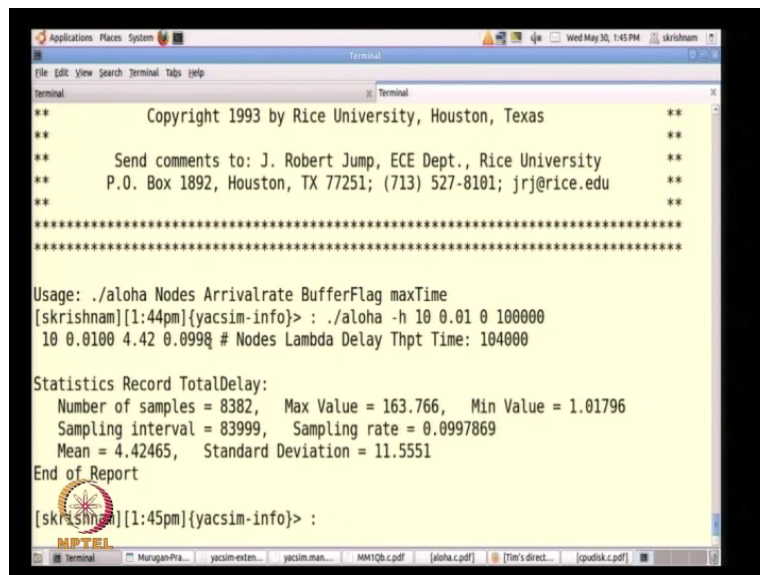
```
[skrishnam][1:12pm]{yacsim-info}> ./aloha

*****
**
**                               **
**      YACSIM Version 2.0, 3/6/93      **
**                               **
**      Copyright 1993 by Rice University, Houston, Texas      **
**                               **
**      Send comments to: J. Robert Jump, ECE Dept., Rice University      **
**      P.O. Box 1892, Houston, TX 77251; (713) 527-8101; jrj@rice.edu      **
**                               **
*****

Usage: ./aloha Nodes Arrivalrate BufferFlag maxTime
[skrishnam][1:44pm]{yacsim-info}> ./aloha -h 10 0.01 0 100000
```

You know, what should be the throughput of the system, expected throughput, is this a lightly loaded scenario or heavily loaded scenario? It is lightly loaded and the expected throughput is simply number of users into number packet generator per unit time. So, therefore, it should be about 0.1 right that should be the expected throughput let us see if we get that.

(Refer Slide Time: 43:55)



```
*****
**      Copyright 1993 by Rice University, Houston, Texas      **
**                               **
**      Send comments to: J. Robert Jump, ECE Dept., Rice University      **
**      P.O. Box 1892, Houston, TX 77251; (713) 527-8101; jrj@rice.edu      **
**                               **
*****

Usage: ./aloha Nodes Arrivalrate BufferFlag maxTime
[skrishnam][1:44pm]{yacsim-info}> ./aloha -h 10 0.01 0 100000
10 0.0100 4.42 0.0998 # Nodes Lambda Delay Thpt Time: 104000

Statistics Record TotalDelay:
  Number of samples = 8382,  Max Value = 163.766,  Min Value = 1.01796
  Sampling interval = 83999,  Sampling rate = 0.0997869
  Mean = 4.42465,  Standard Deviation = 11.5551
End of Report
[skrishnam][1:45pm]{yacsim-info}> :
```

So, here is the output number of nodes is 10, this is rate per user, the delay is the 4.42 seconds and the throughput is 0.0998 that is close to 0.1, that is kind of close to 0.1. Now, only thing is delay one has to you know verify how do you verify 4.42 is correct, we will not know. That

is when we use that semi Marko models we talked about in class, I give the semi Marko model actually the verify that the delay values are more or less matching at least 5 percent, 10 percent close to this theoretical values, so it seems to give me reasonable result for these things.

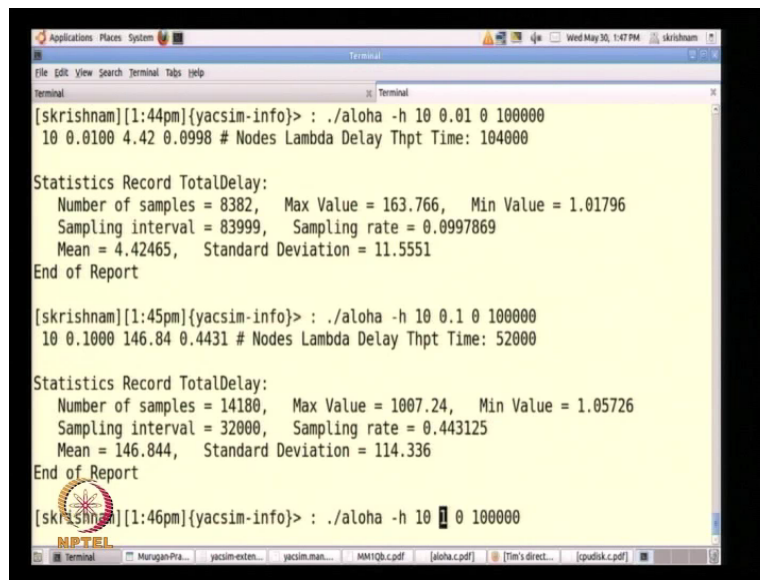
Now, if I increase this rate to 0.1 right what will be the system throughput now, what is the offered load? Offered load is 1 packet per second we know from theory that if I offer 1 packet per second, I would only get about 36 percent theoretically assuming it is all Poisson things like that. So, therefore, theoretical maximum throughput of started aloha is about 36 percent.

But, we will see the little bit more than about 0.4 also 0.44 is, what is the through put of the particular system again, right number of user 10, load per user is 0.1 delay is now very high delay is now 146. Why? Because lot of packet lot of retransmission right. I have not measured I can measure in the stimulation right, how many retransmission where there? I can simply measure number of the retransmission whenever there is collision, I keep track of those things.

I have not done that, we can very easily manipulate the code to give those statics also we can have more detailed diagnostic. Because how many times does the packet retransmitted there is nothing that tell us here. But that we can measure the stimulation you want. So, the throughput is 0.44 okay that does not match my you know system but look about 14000 samples, 14000 samples right, I am saying that, this is based on the auto termination whenever the convergence in delay the system will simply stop.

So, even though I set my upper limit of time to 100000 system actually finished 52000 units because it felt that delay value in conversion enough the samples where there delay is converging the there is no point in running the stimulation any longer. This is automatically done by the `(())`; that is there. So, then we have now I am going make this 1.

(Refer Slide Time: 46:12)



```
[skrishnam][1:44pm]{yacsim-info}> ./aloha -h 10 0.01 0 100000
10 0.0100 4.42 0.0998 # Nodes Lambda Delay Thpt Time: 104000

Statistics Record TotalDelay:
Number of samples = 8382, Max Value = 163.766, Min Value = 1.01796
Sampling interval = 83999, Sampling rate = 0.0997869
Mean = 4.42465, Standard Deviation = 11.5551
End of Report

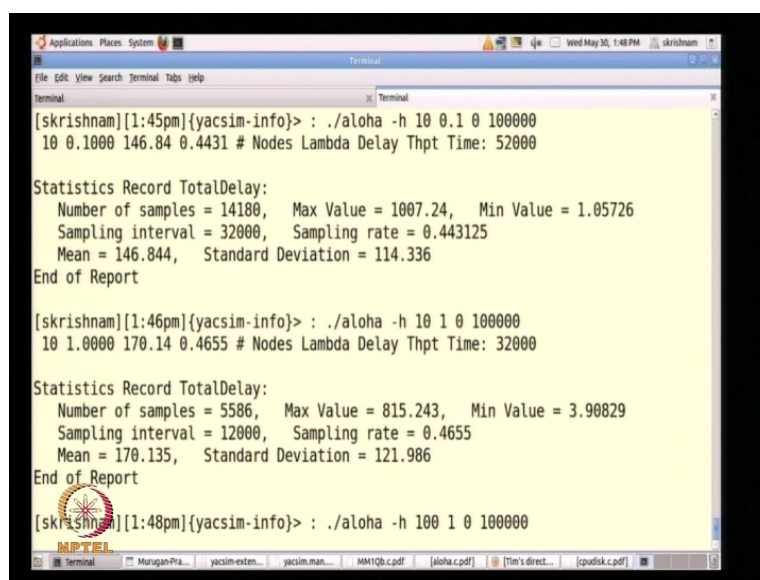
[skrishnam][1:45pm]{yacsim-info}> ./aloha -h 10 0.1 0 100000
10 0.1000 146.84 0.4431 # Nodes Lambda Delay Thpt Time: 52000

Statistics Record TotalDelay:
Number of samples = 14180, Max Value = 1007.24, Min Value = 1.05726
Sampling interval = 32000, Sampling rate = 0.443125
Mean = 146.844, Standard Deviation = 114.336
End of Report

[skrishnam][1:46pm]{yacsim-info}> ./aloha -h 10 1 0 100000
```

So, every one generating every packet every second, so what will happen, what should be the system throughput now, will be very large or small very small right. So, if the theory tells goes to almost 0, so let us hope that stimulation also tells us the same. It is still reasonably we still managing right. So, even through the offered load is 10, somehow system is still giving us, let us try this 100 nodes right.

(Refer Slide Time: 46:45)



```
[skrishnam][1:45pm]{yacsim-info}> ./aloha -h 10 0.1 0 100000
10 0.1000 146.84 0.4431 # Nodes Lambda Delay Thpt Time: 52000

Statistics Record TotalDelay:
Number of samples = 14180, Max Value = 1007.24, Min Value = 1.05726
Sampling interval = 32000, Sampling rate = 0.443125
Mean = 146.844, Standard Deviation = 114.336
End of Report

[skrishnam][1:46pm]{yacsim-info}> ./aloha -h 10 1 0 100000
10 1.0000 170.14 0.4655 # Nodes Lambda Delay Thpt Time: 32000

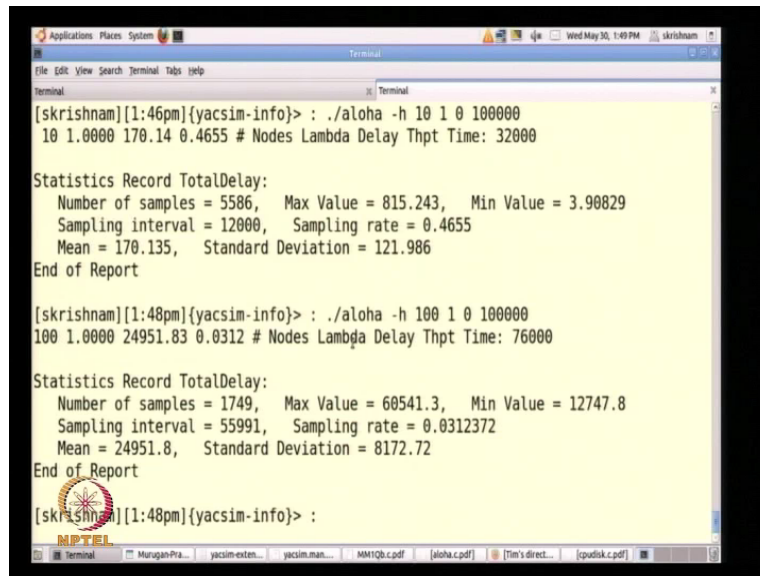
Statistics Record TotalDelay:
Number of samples = 5586, Max Value = 815.243, Min Value = 3.90829
Sampling interval = 12000, Sampling rate = 0.4655
Mean = 170.135, Standard Deviation = 121.986
End of Report

[skrishnam][1:48pm]{yacsim-info}> ./aloha -h 100 1 0 100000
```

Somewhere, I make sure the system load goes to 0, 100 nodes each generating 1 packet per second should cost lot of collisions. Now, see now it is taking long time because there are 100

process is running right, 100 events sequence finally, finished. So now, you will see this point 0.0312, so the delay is now the throughput is very, very small. I offered 100 packets and only 0.035 packets per second went through. So, system throughput is almost close to zero and delay is of course very high; but look at the number of samples collected only 1749 it saying that know probably the last data sent.

(Refer Slide Time: 47:20)



```

[skrishnam][1:46pm]{yacsim-info}> ./aloha -h 10 1 0 100000
10 1.0000 170.14 0.4655 # Nodes Lambda Delay Thpt Time: 32000

Statistics Record TotalDelay:
  Number of samples = 5586,  Max Value = 815.243,  Min Value = 3.90829
  Sampling interval = 12000,  Sampling rate = 0.4655
  Mean = 170.135,  Standard Deviation = 121.986
End of Report

[skrishnam][1:48pm]{yacsim-info}> ./aloha -h 100 1 0 100000
100 1.0000 24951.83 0.0312 # Nodes Lambda Delay Thpt Time: 76000

Statistics Record TotalDelay:
  Number of samples = 1749,  Max Value = 60541.3,  Min Value = 12747.8
  Sampling interval = 55991,  Sampling rate = 0.0312372
  Mean = 24951.8,  Standard Deviation = 8172.72
End of Report

[skrishnam][1:48pm]{yacsim-info}> :
  
```

We can actually, output that total number of packet transmitted very easily if you wanted, that we can modify. So, this is, this is live with expectation we know system through put is this low is going to be this low we do not know, you know it is low, there is some (()) behavior. Whether why should be 24000 we do not really know that we can really verify with the help of some mathematical model right; other model actually make sure is it right. So, that is the aloha model that you want to talk about.

So, any question on this? So, if you want make change the codes, changes we can change the back off probability model right; that is something it is very easy to change. That you can go in I have some sophisticated back off probabilities, you can use some other simpler back off probability you can use the simple back off model. You can use the I have here adaptive back off probability there, you can make a fixed back off probability; at you look at performance of the system.

In fact, if you look remember that ISI model system semi morel model that the backs off probability fixed it is not dynamic where it is my case it dynamic. So, dynamic back off

probability can also cost slightly higher throughput values, what happens is that a lot of collisions you back off to a very small back off probability, and as the number of collisions decreases you start climbing up to higher the retransmission property.

That retransmission property has a big impact to the system performance, and you can now change aloha for doing all sorts of things. (()) of aloha is there, instead of sending packet right away in the first time we have probability of p_i , where you decide with the probability of p_i send the packet as it an array we get the better performance with p_i percent of aloha things is like that.