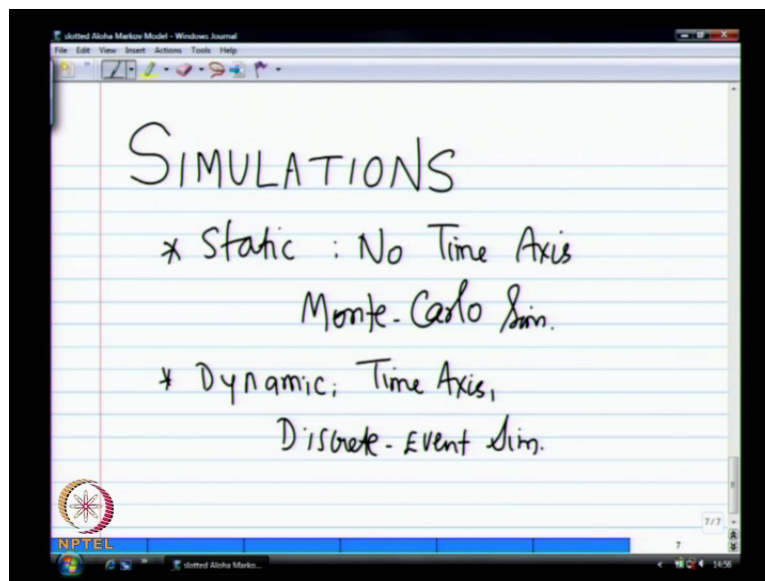


**Performance Evaluation of Computer Systems**  
**Prof. Krishna Moorthy Sivalingam**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture No. # 21**  
**Simulations - I**

(Refer Slide Time: 00:12)



So, the general topic is simulations. (No audio from 00:16 to 00:31) So, we are used primarily digital simulations today. When I was under graduate student, I had a course all simulation, but that was analog simulation. I do not remember any of that, I do not I do not remember understanding in other when studied that ((O)). Some of you managed to go through that course; that was analog simulation, I just I will not bother to go back and check up in to it, this is the whole devices OPAMP something like that. ((O)) might remember better or either Raja Raman's book was there, we went through that mechanically, but I do not recollect what that simulation meant even today.

So, we will ignore that. So, you are talking about digital simulation right. So, in that there are essentially primarily mainly two categories. So, you have so called static simulations, where there is no time access. Then you have dynamic simulations, where there is notion of a time axis, and it is we also refer to that as discrete events today. So, static simulation system is

usually used with random number generators for some sort of combinatorial, you know probabilistic analysis. For example, what we did in our emulator is basically the static simulation.

There is no notion of time. You are simply is **right** flipping the coins. Then you are saying, I am starting in state as naught or a usual state  $S_i$ . Then with, there is some probability of going from one state to other state and so on. So, you are essentially simply moving from state to state, but there is no real time axis as such you simply repeating the experiment **(( ))** times and transition to the various state to understand the behavior of the system. And then from that, we said I spent how much time in state 0; so much time in  $S_1$ . Therefore, average of that gives the probability of being in the each of those states.

So, there is no real notion of time. It is simply random number generation based modeling of a system behavior. So, that is what **what right** that is easier to do; that is also called Monte Carlo simulations. (No audio from 02:46 to 02:56) We have this birth day paradox; you remember the birth day paradox. What it is a probability the 2 people in a room will have the same, but what **(( ))**. So, that we can do mathematically, but how do validate that? So, I can validate that by simply generating random numbers for every person's birth day. And then looking at, I just have say 100 examples randomly generated birthday for each of them.

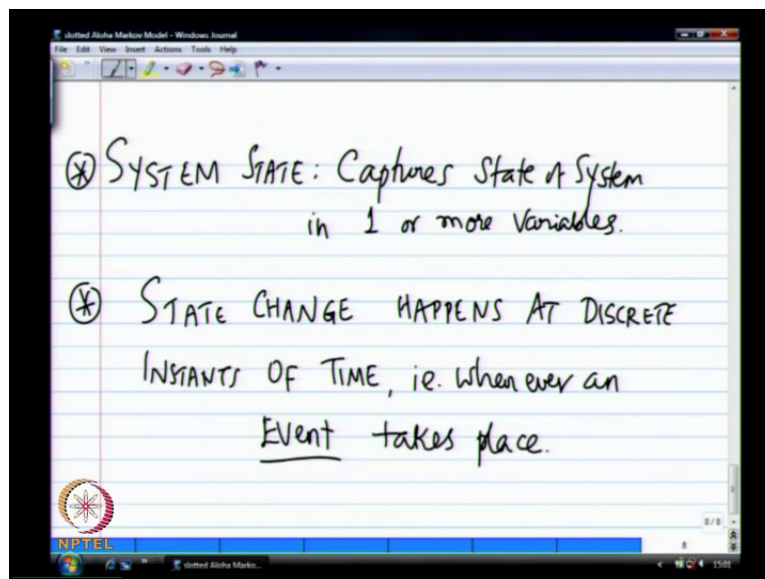
It will be 1 over 1 365 uniformly, and then simply fill up **fill up** say 100 such birthday and then calculate the probability. What is the probability at least 1 **1** pair will have the same birth day. That is done by simply looking at all possible combinations and of course, I can **(( ))** for 100 users, 1000 users, million users and so on. And then I can also try this number of trials also. I can do just 10 trails and come up with the number or do 1000 trials are come up with the other number and so on. So, there is that is the essentially a static set of simulations.

In the other context of discrete event simulation which some of you all would have done w c r. Some of you will do n S 2. It should how much of how **how** many of you do not really know the fundamentals of n S or discrete event simulation. Most of you seem to know, what is the discrete **(( ))** the tables around. Tell me, what is the? I should **(( ))** since and then we let them also they will listen and I will **(( ))**. So, what is a discrete event simulation? So, we are now looking at modeling as system behavior with time axis. There are some events happening with respect to time. There are some system changes **yeah right**.

State changes that happen as time progresses in the system and this is not only computer systems or networks or whatever it is, this is used everywhere. For our climate modeling also people try to do that, you come up with some event for your prediction in terms of whether modeling for example; you look at what happens with time. Here are some models for it and then you introduce time in the picture and then see how (( )). How the system will behave over long periods of time. So, when you look at the system representation as such why is it called a discrete event simulation?

Suppose event we are not like the (( )) random time every time it is different values and data. So, first you represent the system state using some variable. So, we need to first of all define system state. So, that is the first just like we did it in the Markov model. You capture something as the system state in the case of the M M 1 queue, and use the Markov chain. The number of customer in the system was my system state and I modeled it I simply model that behavior of that system state change with respect to time.

(Refer Slide Time: 05:45)



So, you have a system state which captures the state of the system in 1 or more variables. So, if I have a multiple queue system right. So far network of queues, then the system state is basically number of customers in the first queue, second queue, third queue and so on. That is what, is capturing the system state. So, that is. So, at an instant, this is the state of the system. Now as time progresses, the state of the system changes. Now, the state of the system actually

will change on continuous basics literally. I mean which possible that it is for example, if you look at the queuing system.

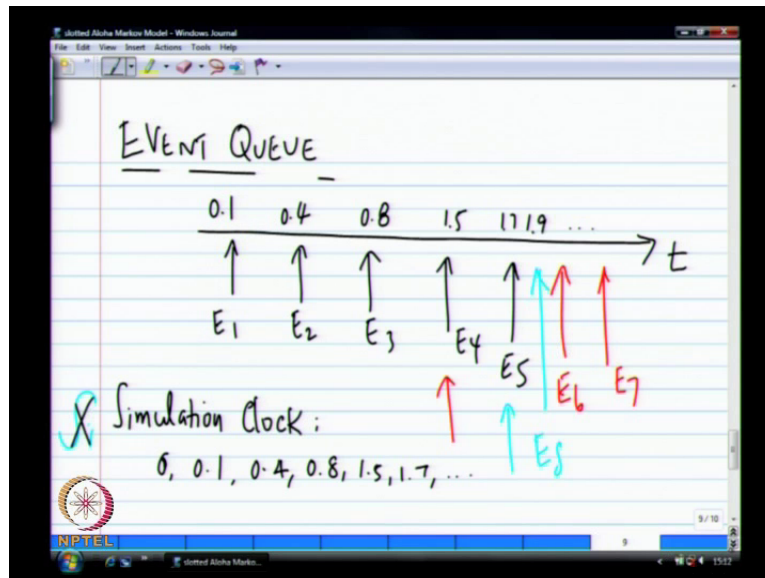
But every instant of time, the number of customers one customer can arrive, one customer can depart and so on. So, really it is the continuous; the state changes with respect to continuous time. But continuous time is hard to model in a **in a in a in a** real system. So, what we do is we only look at the system **strange** system state change at discrete events at discrete instant of time. So, you say there is an event taking place only when there are events occurring, thus the system states change. So, you model the behavior of the system as a sequence of state changes, where the changes occur at specific instants, which we call as events.

So, some event takes place to changes your system state. It is like interrupts in your p.c. Interrupt arrives, something is done, everything is interrupt driven, if you look at the system with the timer interrupt changes, your process that is standing on the CPU to some other process. If it is a hardware interrupt, you go on process that hardware. There will also invoke a change in terms of the system. So, your state change (No audio from 07:34 to 07:44) happens at discrete; so that is whenever an event takes place. (No audio from 07:59 to 08:12) Every event will trigger additional events also in future.

As when an event occurs, you will be scheduling some other events in the future that, you know that can **(( ))** something else and soon. So, with this you start of your system with at least one event, one boot strapping event; that will then spond other events as it guess. So, every event will occur, that event has to get processed. So far every event, you have to associate an event handling function. So, then event handling function's job is to handle the event whatever do processing and then in turn generate future events that will be again sold in the some.

That is going to be stored and then that will be processed. So, that is all. Your system now a state of the sequence of time ordered events. So, every event has the time associated with that and you simply process this so-called event queue in time **yeah. (( ))** that we do not consider the state of the system, why that event is occurring busy at the beginning of the event and at the end of the event **yes ,yeah yeah** at the beginning of the event, you will change the state, and then you say I am done. Then the next event is processed **right**.

(Refer Slide Time: 09:25)



So, your system is... So, essentially you have an event queue. So, event queue is that tells you the set of future occurring events at any point in time. You may have some 20 events in a queue. So, let see if this istime axis. So, there are events that occur **that that occurs** at various point E 1, E 2. (No audio from 09:49 to 10:03)

And let say that, the current time of the system is E 4. So, each at this is the next event that you are processing. You are processing event 4 that is taking place at some point in time at t 4 whatever that.

So, then when you process E 4, already there is anevent E 5 that is supposed to take place at some point in the future, that is the already in the system. But as the result of processing, you might **you know** and up creating couple of other events E 6 and E 7, that occur in the future. So, that is **that is** how your events keep on getting generated for future. So, this is your **your** you know that as the result of for example, what is the whatis one eventspondingon other event. So, if you look at the M M 1 queue, event is sever fix up the packet for service; customer for service.

So, when you pick up the packet for service, you will simply schedule an event that says current time plus service time. Service time; let us say 100 milliseconds current time plus service time, I will schedule the packet completion event. So, is the packet transmission event essentially when you process that, you also schedule the packet completion event for some other point in time in the future. That packet completion time might do again something else.

But in normally, what you have is you have you have you have usually two separate processes, I will you give one example of this M M 1 system as you go along, but that is the basic idea.

One event when finally can generate other events to model, and how is that generation; that is your logic comes, you write the logic for generation of events and where does your logic come from? Your logic comes where you understanding of the system itself; that is representing the system behavior; that is model of the system itself. So, then **then** after E 4 is done, then you say process E 5. So, then the system moves on to processing E 5 and usually, you do not try to schedule events before the current time. In theory, you could because then you are because you are simply maintaining a ordered less.

You simply, you could say the E 5 go and process something that happen before E 5. That is not what we normally we do; what I can suddenly do something like this. I can create E 8, which is ahead of E 6 and E 7; some other event that is supposed to be processed before E 6 and E 7. No, **no no** every time you have to create some events, **you your** as part of your logic, you will say this you start up with say only one event. Event is let us say, the first arrival in the system. And then the first arrival this system will process will spend the next arrival in the system next arrival in the system and soon.

Case I will 2 and so on. I am just saying the states. But in between, there may be other arrival set; other events that can take place, but still I move only states. No, **no no** your state is not the issue. I will show that example. Your **your** each of the you events, there several possible events, that can happen. Only thing we are doing is time ordering. When you do you will not know all the, but you know all the possible state. But when those events are going take place is going to be random time say some random generation involved there **(( ))** and same time your arrival coming yeah.

So, that **that that that** will not happen, because that **that** is why, we will have to see the let us do that some example **(( ))**. You can use the events that are already generated, but ahead of it time. You will generate an event, when you are processing in event. Suppose now we are processing E 5, I am generating new event. So, can I use E 6 and E 7? No, E 6 and E 7 No you do not know. They are simply sitting in the queue; that is all. They are supposed to happen at that point in time. But E **but E EE** 5 processing cannot; you can if you know the exits, you can delete that event, but it right and you can delete some future event.

For example, timeouts in a packet transmission in a link; you will schedule time out interval, timeout event always every time you send a packet. If you get the acknowledgement and you know when this particular event is going to occur, you can delete that event; that you can do, because that is **that is** what we do in our real life too. You always cancel the time. Time of the event is exactly that we can use it **(( ))**. If you have handle for that time also handle for that. Usually, every event will be given an event handler, some pointer will be there. Some sort of representation will be there for it to handle.

So, you can use that to do the processing if you want. But you cannot essentially trigger that event ahead of time; that event **you if you** if you want, you can reschedule that event to some earlier point. It is like essentially cancelling that event and then starting a new event ahead of time. E 3, E 4, E 5 ordered of schedule 1, and then E 5. So, we started off with E 1. Let us say that is what happened, E 1 is the first event in the system. We have to schedule the first event. E 1 spond E 2; E 1 processing to E 1 spond E 2; E 2 spond E 3 and then soon. So, we are now at the stage, where E 4 is the current; E 5 was already spond by; E 3 spond E 4 and E 5 with respect to time.

And E 4 handling created these two events and thus, then the system say what is the next item in the queue? After handling E 4, next item in the queue is E 5 and E 5 is basically sponding another event called **(( ))**, which is at this point at time. In the system **yeah** this is independently. So, you can execute that independently event will handle, which one? The E i; E i for every event you will have corresponding handler; whatever is handler will be invoked. So, **it is for a** it is like a packet sequence in the UNIX system for every system you will have corresponding handlers.

So, here you will register for every event, you have to say **what a handler** what the handle code is going to be... Then more than one can focus simultaneously. It can the system will have to do some sort of priority, then you are end up with the race conditions. If depending on which of those get selected first, you **you** have that the behavior can change. So, unless you have a way of distinguishing events like the priority or something else, system can simply do first in first out and if there are multiple events, it can randomly choose one and execute.

So, when you write your code, you have to be really careful about which one; if the order of execution of events changes which are scheduled for the same time, will it make a difference to your program; that is where, some time errors happen. So, debugging discrete event

simulation code is always ordered, because of the some of these unknowns. Because there are randomness in the system and same pattern might not be repeat the next time, you run the code, you run it once, the system will just hang. Then again suddenly it works beautifully by because this two events happening at the same time did not happen.

Because the way your random seat changed. Therefore, different set of random values generated. So, first have now what we understand is, the system can be modeled using a sequence of events. Each event changes the state of the system from whatever set and each event can spond other event. Every event has that event handler; this is what we have. So, if you going to write your own discrete event simulation package; so now, one more thing is that, in thissystem there is a notion of simulation clock. (No audio from 17:21 to 17:37) So, the simulation clock also advances in discrete time in intervals.

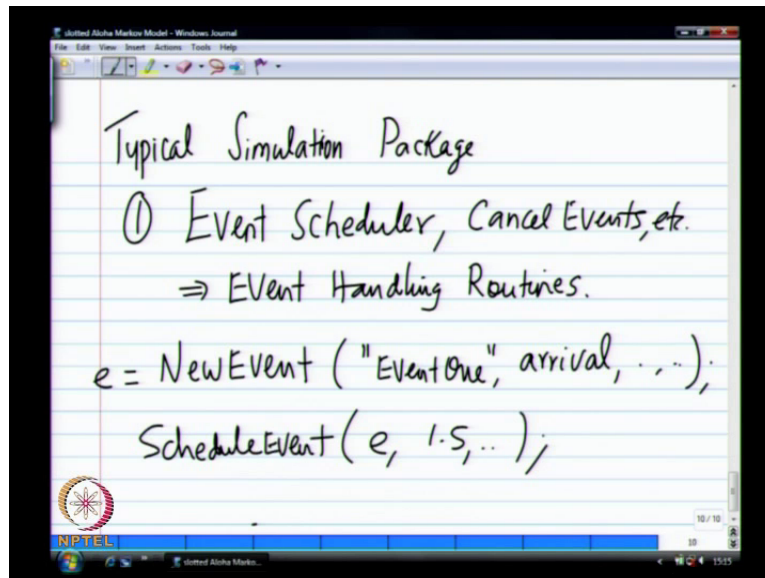
It is not like a regular clock where you have to go from 0 to 0.1 or 0 0 and so on. So, example if this was at 0.1 **right**, this is 0.4 etcetera. So, every time your system your simulation clock will be set to the current event time. So, when you schedule an event, you will also tell what time that event is going to be having. So, you are simply your clock will jump from one event to the next event and yoursystem simulation clock will start off with initially will be 0. And then when I handled the first event that becomes 0.1, the second event the clock becomes 0.4 and so on. So, it is a discrete.

(No audio from 18:19 to 18:29)

So, that is the simulation clock. So, in every package, you will find simulation time; get simulation time or time now; that will be there to tell you the current time for whatever delay calculations and all those things. So, this serves your system clock, basically with these two things, you have a discrete event simulation. Scheduling an event and handling that event, then the clock that is moving along with the based on the event set of **((C))**. (No audio from 18:53 to 19:03)



(Refer Slide Time: 19:04)



So, typical simulation package, so if you are trying to learn a simulation package, the first thing is you will ask for are these things. First is, how do I schedule an event (No audio from 19:24 to 19:33), and there are different ways to schedule events. The simplest thing is schedule an event 10 unit from 10 time units from now; that is all in the future. Then if you look at this (( )) was slightly more sophisticated on you can schedule based on a resource availability. But the simplest is, I am now at 0.8 time units schedule an event for 1.2 time units that is alright.

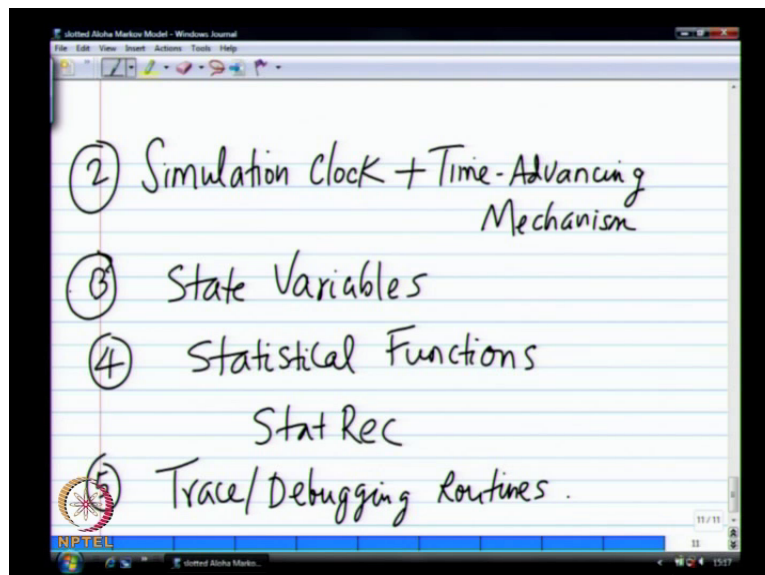
So, that is a event scheduler and then also have to cancelling events **cancelling event** etcetera. Creation, deletion, modification, rescheduling (( )) you know variations. But the simplest thing is create an event, and then we also have to correspondent **have to** the event handling routines. So, event handling routine is basically your core logic for the program; that you have to right. But you have to know how to associate each event with the corresponding event handling. So for example, in (( )) you have something called new event. You will give its some name; event one.

Some string there is for mostly debugging proposes. Then you will say arrival process say arrival; arrival is simply a name of the function point; points to function call arrival which we have to define somewhere else in your core, and some other things. And then you will also say the time into the future. This will be simply the event creation itself. Then you will have

schedule event, for example. So, this is defining event; you might have some `(( ))` and then you would have something like `you know` schedule event.

Schedule event is not any `(( ))`, but the approximately what do you say. So, schedule event eat `you know` 1.5 something. So, it is relative time; it `it` will tell you whether the syntax is relative time; absolute time. But it is always `you know` that from now for example, you are scheduling a packet for transmission. You know the length of the packet. So, you know the time it is going to take to complete packet. So, simply plus comma the service time; thus going to the time into the future; you will schedule the completion event only start.

(Refer Slide Time: 21:55)



So, that is the first and then this notion of simulation clock. Because all your delay calculations depend upon this clock interface that they give plus the time advancing mechanism. (No audio from 22:09 to 22:25) Then how do you represent state variables? If it is C, then it is a bunch of C data structure `structure`. So, that is starting there; the rest of this is about it the lots of other things `you know` listed. How do you get all like all the input stuff? So, then in addition to this, what you would need is some sort of statistical interface; some statistical functions.

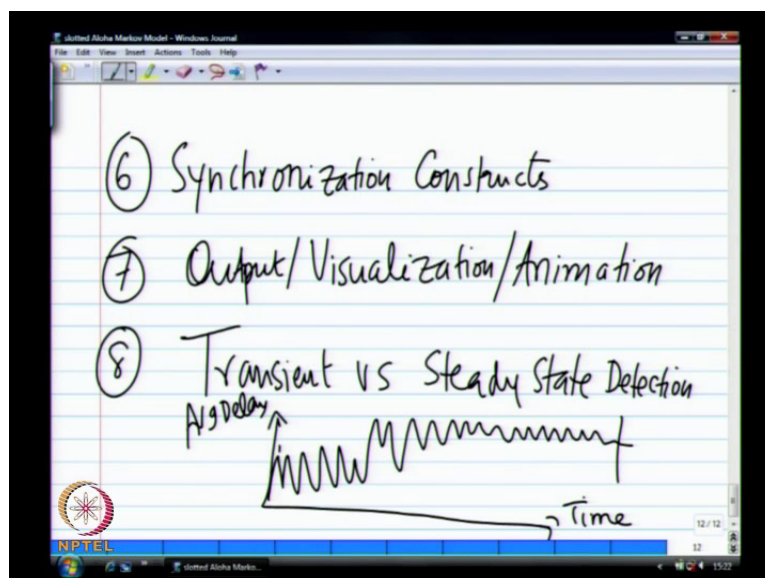
These are not compulsory, but most of the times you know, because you do not want to always keep on calculating the mean and standard deviation. All the system can simply do that for you. Say every time you say finish a packet transmission, you know what the transmission delay was... You simply want to say, here add this to the correspondent

some statically variable that you create. You simply said dump it on this variable. Let that take care of getting histogram or mean or whatever it is. So, the statistical function if they are there, it makes your life easier.

So, you have to look for what are the different statistical functions? So, in the case of (( )) there is something called stat rec. From the stat rec, you can get a lot of. So, there is a statistical record is one; is function; it is a variable and for this variable, there are a lot of functions that let you get the histogram mean, and all those things. And there are sophisticated ways of actually computing; this means point mean other things. Then it should also give you trace and debugging. This is particularly important, because it is not sequential code. There is a lot of time randomness synchronic here.

And **and** some time we will see if it is, it can also be essentially, it is parallel code within the same simulator, I can actually have just like UNIX system; I can have multiple processes joining. And then those processes just like UNIX process can share variables, can have semaphores, barriers all those things are there in system like (( )). Others might not have, but you can also essentially write your code as the sequence of interacting processes or as the set of interacting process. And then you will have to worry about all the synchronization constructs. So, which is also you know typically available. So, we will list that also.

(Refer Slide Time: 25:00)



Synchronization constructs (No audio from 25:02 to 25:25) and then one very important thing **I am sorry** graphing your statistical packages. Some of the packages give you graphing

utilities. But you look at n S2, n S 2 give as gives you well there is name for example, sometimes they have animation. Because if you simply run the simulator and then look at the numbers and now the tracefile will come out, which you have to sit in process. So, there are may be some... Fine, we will say the output visualization, animation. So, when slots that are come, we has to do a demo.

You simply get an opponent and then use opponents. Say you can run opponent show that as we develop the opponent and then show the one's simulation of you know vehicles moving around in all the demonstration. So, go it is spacy thing, but that is usually only for user only for this demo process and besides that, there is still one important, documentation **documentation is right**. Documentation is one pdf file, let us about all the tools see for documentation. In some example files, that are there which I wrote some time ago; one very important thing all the how will. So, you are talking that is a good point.

When do you start and when do you stop? Usually packets do not tell you that; you have to improvement your own routines for example. Because your system will always start in a transient state and then it reaches steady state. And later on, we will see how do you know the system is passed the transient state? Sometimes people want to analyze only the transient state behavior; because that is very important. Sometimes you only want to analyze steady state behavior. So, you have to have some ways of specifying transient versus steady state, which I am not sure of all packages that you do that.

Because it is dependent upon the variable you are trying to monitor; like some variable might reach steady state quickly. But some variable may take steady state variable take a very long time, depending on what you that are machining. Transient versus steady state detect; because there are two things; you initial data in a simulator will always be not useful for if you looking at steady state results. And, when do you know when to stop the simulation? that also package will never tell you. In fact, this package **(( ))** does not, but there is some extension, that you will see called new stat recand soon which will try to do batch replications.

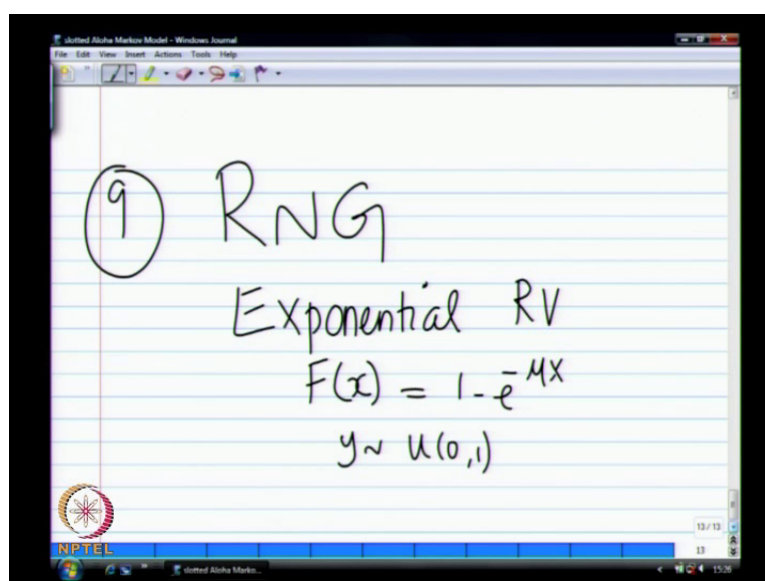
And then automatically let us say you are trying to converge on delay. So, it keeps for every 500 customer, computes the delay, computes the delay and so on. And then finally find that, there is no point running the simulation any longer; because delay is now more or less stabilized. So, if you look at your system behavior, let us say delay is what your average

delay is what you are trying to plot with respect to time. So, first customer delay will be something. Then your delay will go up and down like this. But finally, at some point in time, your delay values because this is the long term averages. So, it will finally stabilize.

And if this is a steady state, then you want to know that beyond a point running the simulation another 2 hours will not make a difference. So, when do I stop my simulation? Many times people simply say 10000 whatever 5 minutes of simulation time **right** manatee n S 2, whether the system steady state is achieved or not. We do not really care try 5 minutes; because that takes 2 hours. So, I cannot in any longer. So therefore, that is the end of the... Patience is over this is the numbers; take it or leave it. And sometimes, you are very keen on how long it takes to reach steady state for example or what.

Because queue behaviors change during this appearance, the steadying queues for example. The steady in transient state might be more useful. No, I am still looking for one more very important thing. So, sometimes packages might give you ways of automatically, there is auto terminate, you will see that function in **(( ))** which will say have to terminate and it will work only the system will reach stable condition. The system is the particular way we will looking for does not really converge. Then the program will run forever unless you give some upper limit on time. One very important package is missing, we are taking of time, we are talking of discrete event simulation.

(Refer Slide Time: 29:50)



In random number generators, you need random number generators not just uniform. How do you generate the random number in Linux or Windows? D random or L random right. So, you have some RNG's in the system and if you want double values, you want double. Random is integer, you want double Drand, or you also go to mat lab for what did not say. So, you need some very good random number generator. This book has the chapter on RNG's which lets you know there are two. One is mathematically you should prove that, this is really got randomness property and the cycle is very long and things like. Then there are cryptography random number generators.

But generally you need a good random number generator not just for uniform random; what if I want to generate exponential. How do I do that? I want to generate an exponential random variable. I want to generate a Gaussian random variable. I want to emulate Poisson arrival. How do you do that? You have done RNG's. Take the inverse of the function you want and you get the yeah, one of the ways. If you know this is not readily available, then you will have to go for other technique.

So, usually we can do these ourselves. If I simply give you uniform random and random number from 0 to 1, I can translate that into an exponential random number. If the inverse is is easily if I get a figure, but in some cases inverse is not available. So, again there is a chapter on how to generate random variants, which we will not get into right now. If you are really interested, you can look at that chapter on how to generate random number variants. Exponential is easy because we know that  $E\{x}$ , we know. So, because of the fact that,  $E\{x}$  is sorry  $E\{x}$  not;  $F\{x}$ .

(No audio from 31:42 to 31:52) this is the cdf right. So, cdf is  $1 - e^{-\mu x}$ . Now, what I want to do is given  $\mu$ ; I want to generate  $x$ , which over the long run will basically emulate  $\mu$ ; will emulate this exponential  $(\lambda)$ . So, you what will do is generate some  $y$ , which is uniformly distributed from 0 to 1, and then will essentially try to get your value for  $x$ . So, given some number  $F\{x\}$ , let us say  $r$  is the number that you randomly generate. Your  $(r)$  into this one, then will do your simple  $F$  inverse of  $x$  is what you will try to do.

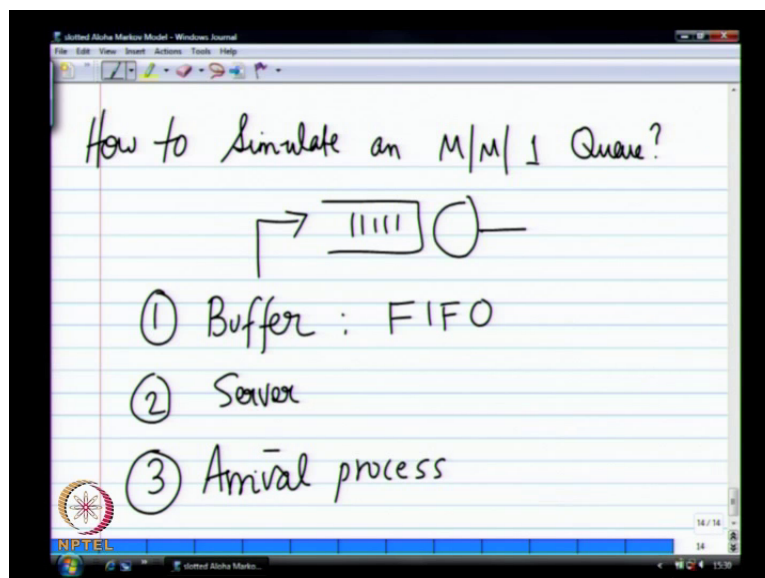
So, that if the system gives you this function readymade, it is good, and then we can essentially start off with your random number generation. So, these are some of the main entities that you look for internally to actually, if I say build your own random number events simulation discrete event simulation system. It will not take much time. All you

have to do is be able to maintain an event queue. How do you maintain event queue? Simply ordered list linear time ordered list will do; but if your list gets very large, then your cost of maintaining that will be high or you can use the heap.

So, that is the heap is also possible or you can have buckets is so called calendar queues, where you have set of queues and then sub queues. Again I am not, since there is not directly related to us, we are not really building system. But there is a small section of how calendar queues can be used. So, an index linear list. So, this is these are all things achieved essentially need. So, as when you start writing your simulation model. So, you figure out what other system variables, and then how do you capture the logic of the system?

And then two important things that which are not really sometimes available, which you mentioned before is how to terminate this simulation and how to delete the initial data? It is called initial data deletion, and **(C)** and terminating the simulations. So, **will** I will come back to that after next set of classes. So, questions on these basics so far? So, ready for waiting M M 1 core with pseudo interface. One should how **how** would M M 1 system look like if I want to simulate in M M 1 queue? What should the code look like? You will see the actual code on Wednesday.

(Refer Slide Time: 34:39)



But you will understand the code better if you kind of **right** see, how the code will be developed? How is being developed?(No audio from 34:45 to 34:56) So, to develop a queue model for a system, you first identify the entities of the system and which is going to

capture the state of your system and soon. So, what are the entities in my system? What do I have to essentially build? I need a buffer. I need one buffer structure that will store. So, I need some sort of any number and this is the simple 5 4 buffer. I can make this more complicated if I want to, but let us say it is a 5 4 model.

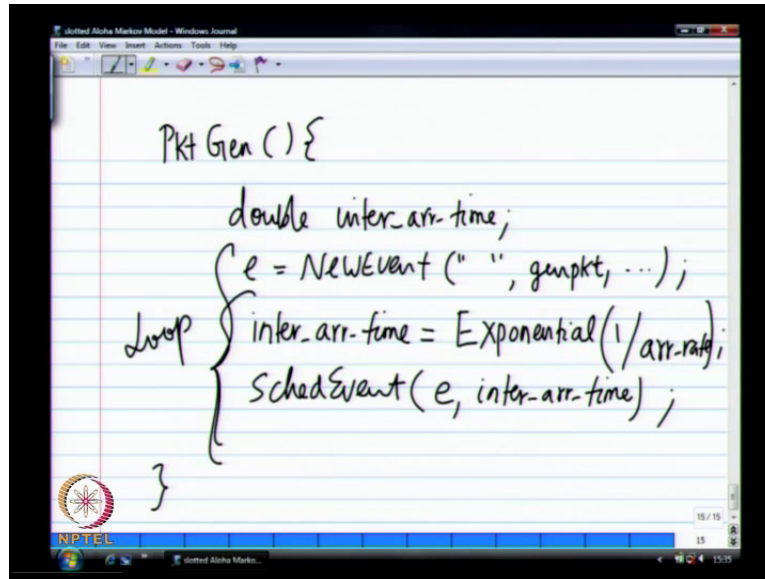
So, that you can write in node time, simple linear linkedlist will do or doubly linked list whatever it is. You are adding elements to the list or queue basically if you unless doubly linked list, if you want to do something like priority base scheduling, where you do not maintain the list in r i. So, if you randomly or I do the random scheduling, but queue structure will self loop. How do you, you can write a queue to add elements at the end of the queue and delete elements from the beginning of the queue. So, that I can do; then, how do I then what else?

I need also to emulate the... I need to have something like a server. I need the server. I need something to emulate the behavior or to model the behavior of the server, wherein the queue is empty, this **this** server will go to sleep; whenever the queue is non empty, the server will pickup packet; then pretend to be busy for the duration of the time that it is servicing the packet then **right**. So, you have to have some server entity that is going through the state changes. It is either idle or servicing and does it as you finish your servicing one customer; it looks at the queue as the queue I can work it out.

So, we need some have to model that server itself. Then the next thing is actually how do you fill up this buffer? How do I emulate my arrival process? How do I handle the? That will be Bernoulli, but here I am saying I want exponential. We can what you did there in the first assignment? **Yes**, we can say in every slot arrival where flip a coin and then disable whether there is a packet or another packet. So, that is Bernoulli is easy, but I am saying now I have scan of simple. I really want to emulate exponential.



(Refer Slide Time: 37:20)



```
Pkt Gen () {  
    double inter_arr_time;  
    e = NewEvent (" ", genpkt, ...);  
    loop {  
        inter_arr_time = Exponential(1/arr_rate);  
        SchedEvent(e, inter_arr_time);  
    }  
}
```

So, the way we do that is. So, let us say that you know I have something called packet gen. This is the it is say process or some function that is going to be you know this is the when you start your system, you start say implement start with this packet. So, I have to essentially simulate the first arrival. So, what you do is inter arrival time or let us say double. (No audio from 37:48 to 38:04) So, we can say  $e = \text{NewEvent}(\text{" "}, \text{genpkt}, \dots)$ , where this is some name for it, and let us say you know generate packet. The actual code will be slightly different, because this is roughly what you can do; think of.

So, I am creating an event; I just this only creation of event is not scheduled yet. No, I want to schedule this event for the first time. So, I use my schedule event whatever is my schedule event right so but before that, I need to find out my inter arrival time. So, I know lambda; let us say lambda is the arrival rate that is given to me. So, I simply use in the case of  $(())$  that happens to be one exponential generated. Since I know that arrival process is Poisson and I know that the arrival rate. So, arrival rate is given to you. So, lambda mu are specified, that is about all that you need.

In this infinite case buffer case, if you want I can do finite buffer also; no big  $(())$ . Infinite buffer chain to finite buffer is trivial, so I generate this inter arrival time and then whatever I do; then I can simply say scheduled event. (No audio from 39:31 to 39:43) So, I will simply schedule this event into the future inter arrival time and soon. I have scheduled the first event.

So, from this instant, I am generating the next event. Then what happens at that time 0 plus whatever current time plus inter arrival, this gen packet is going to get invoked.

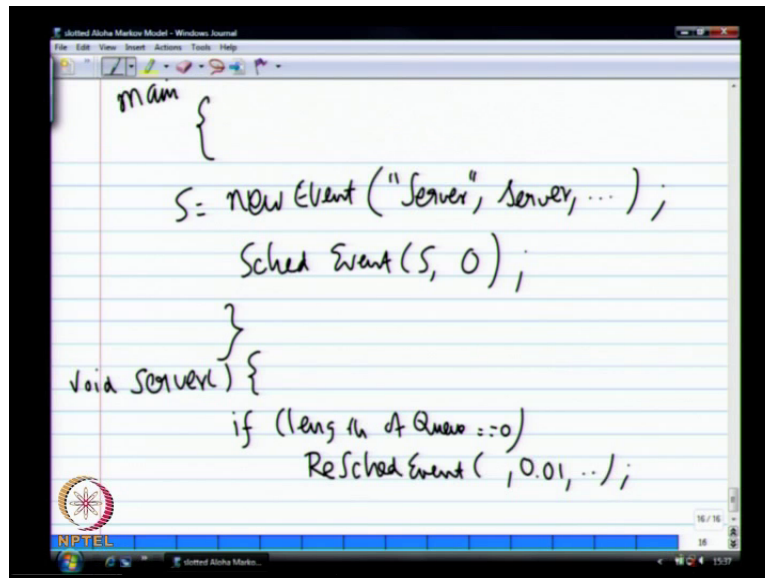
So, gen packet will simply generate the packet; some packet error such you will define; fill in the fields of the packet; add that to the queue. So, fill in the fields of packet; add it to the queue. Then gen pack will then essentially have to trigger itself, again saying calculate that next inter arrival time. So, this could be within a loop, if you really want to do **right** where I have a loop that will simply generate the packet, fill up the fields. And then you will re schedule the next event based on the next inter arrival time. So, this is something, which you would try to do this whole thing you could put in a loop, if you want to.

Go back and forth when you handle this next event - the first event, you will generate the next packets arrival, which will again be based on same inter arrival time. It was exponential way  $1 \text{ over } 1$  by arrival rate. So, you have this simple loop; it keeps on repeating, where you constantly generate packets. So, this is the separate set of event that is happening. Packet generation is like one separate set of event that is happening on its own. It keeps **keeps** on filling the buffer. On the other side, you have to have this server that picks out the packet.

So, you have to have some mechanism now to have the server process to be scheduling. So, how do you do that with events? I have to have an event going to be, I do not know when that event is going to get triggered; this uncertainty in that event. So, what we will do? Some sort of notification or I simply create one event called server schedule at time 0 and we say server wake up every time instant. So, simply reschedule server 1.0, queue empty; if not empty, go back to sleep, **sorry**. If not empty, handle that customer; if it is empty, go back to sleep **right that is all**.

So, there is initial waiting period where you keep on checking whether the queue is available or not; where there is a packet in the queue is ready to be serviced or not. I can make that as my event. So, I start with one event called server event, which is at the beginning of the simulation that initially checks every slot. I can make that unit to may event 0.01 slots, because simulation slot does not really matter. I can do every 0.001, keep on waking up and check. Let us see, if I can write the pseudo code for that.

(Refer Slide Time: 42:22)



```
main {
    S = new Event("Server", server, ...);
    Sched Event(S, 0);
}
void server() {
    if (length of Queue == 0)
        Resched Event(, 0.01, ...);
}
```

So, I have this in the beginning. In main, I have created an event called say S. S is the server event and I have scheduled S 2 start of right away. Server gets to work instantly as soon as the system simulation starts, so you what you will do? You normally look at the main program will have the sequence of these scheduled events for all the boot strapping events. Then you will say drive or run, which will basically start the simulation, you set up all the events, then you say start. You could have seen that in n S 2 also, **right** n S 2 start and stop.

So, my server is now simply a function; normally, it is void function; where you Len check the length of the queue. So, if length of queue equals 0, then in (( )) there is a way of rescheduling in event without having to create that event. Again you can simply say reschedule event at say you will simply say reschedule. But whatever (( )) that you are going to wake up and check 0.01 time units; check it again **right**. We can make of the sever rate, how the packet arrival event there is generated; there is no server. No, my server is simply a function. You are thinking of server as the process.

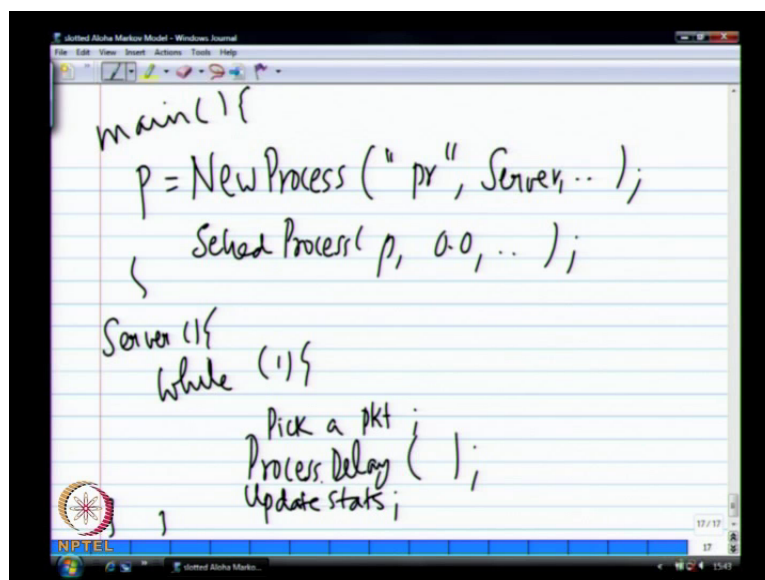
If there is a process, I can wake up the process. Here, there is no process as such. There is simply an event function that is going to be triggered at regular events of time. We will there is a notion of process in (( )) that will benext for a (( )). So, this is one way of doing that; without process, I can simply just have this event (( )). Event of packet arrival can trigger, **yes**. So, here what will happen is you are thatno actually there is no approximation; whenever

the packet is **yeah**, there is a delay of 0.01. So, there will be that that will be of it is not instantly that the server gets to work; that is, if I implemented this way.

I can also implement this with implementing the server as the process, then I have synchronization variables in the case of you can set of flags. So, whenever a packet arrives or the queue length becomes greater than 0, there is the flag that you will set. Then react some internally will go and wake up the process in the case of react some. So, there is the process model, I do not want to give too much; the process models the example will come. So, if you implement as the process, then yes you can do this.

Infact, the example that you see will you see is using something all the resource, which is there any reaction, which actually has a built in server. So, simply you taking an event and schedule it on a particular server, that is not as integrate as this one, but that is the starting point. The little bit harder to figure out what is going on, because the implementation of the resources now within the reaction itself. But if you want really emulate the server, one is to do this kind of you know periodic checking for your packet or have a process model. So, in react some we can also create a process.

(Refer Slide Time: 45:52)



```
main() {
    P = New Process ("pr", Server, ..);
    Sched Process (p, 0.0, ..);
}

Server () {
    while (1) {
        Pick a pkt;
        Process Delay ( );
        Update stats;
    }
}
```

This is almost like our threads interface. I will say P is the new process and then you will schedule you have to schedule the process also initially and you can schedule the process any time you want. It could be 10 seconds, 100 seconds in the future, but let us say you want to start at **...** So, this is what you do in main, then your server function is now a process that is

always running in react some. Now, this server you can do your, you can look for synchronization constructs; there is barriers; you can use semaphores, if you want to. You have there we can simply use simple semaphore and just check whether the values you know greater than 0 that is all.

So, you can use the same producer consumer construct **yes**. Reschedule if it is 01 with 0.01 **yeah** 0 little if we running in differently like process **yeah yeah**. You do not want to do that or if you do that again what you will do their comeback and check. But the thing is if a packet; so, what happens is let us say that the same instant. So, that two event happening at the same time, one is the packet arrival and one is this checking. If the system that react some scheduler picked up the arrival process before you, then then they are lucky. **Yes**, then the packet will be there in the queue. Then you will have length will be greater than 0.

So that, that is there; you can also do it with that. But for that at some point, that arrival process has to be selected. If the arrival process never gets selected, then that is all that event will never take place; this event will be simply checking infinitely. So, that is at capture that you have that, so now the server... You can do time plus normally it is time sorted, you can also have priorities for the processes. So, if you want to say arrival processes as high, in react immediately there is a priority for the process. So, you can say arrival process has higher priority than server **server** process.

Because that way if there is an event, two events on the same time, you will schedule the higher priority process first; in which case, arrival will be handle. So, that if you do, then you will not have this race condition, this server will be an infinite y loop. So, this will be while **you know** just this if they are wait for a packet arrive, when a packet arrives you would. So, in the case of a server, you will simply delay the server. There is something called process delay like our sleep or **right**. So, you pick up pick up packet, process delay which process delay will simply you put the process to sleep for one packet unit of time.

Then, when you wake up, you will update all the statistics that you want to do. So, this is how it works. Update stats, the pickup packet pretend to go to sleep pretend to busy basically **right**, at the end of it, this process will be woken up, you compute the delay whatever that you want throughput all those metrics in the statistics. Then go back and wait for the next packet to be serviced, that is also a model,

**(( ))**, but that will stop time progression in the system **(( ))** because if you...

Until...

See, if you consider the queue is empty at the point of time.

Yeah. So, that is fine. Time progression will be stopped, because there is no real thing happening. So, it is almost like continuous time simulation.