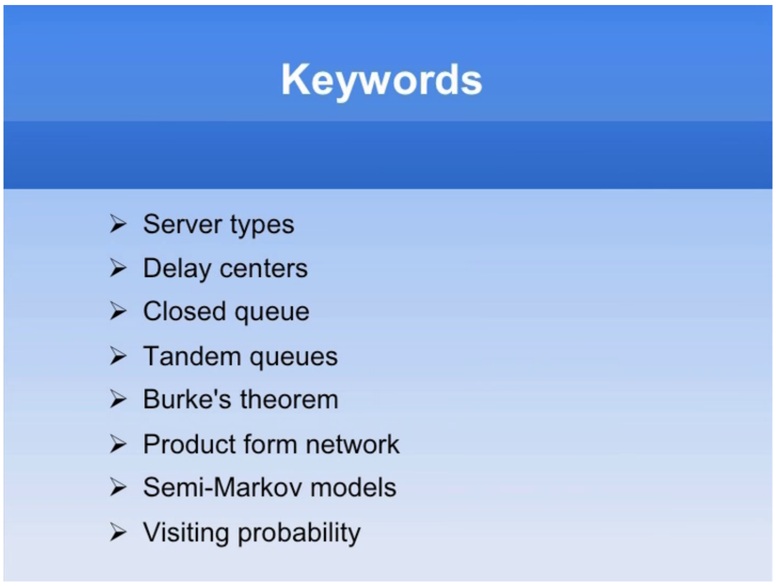
Performance Evaluation of Computer Systems

Prof. Krishna Moorthy Sivalingam
Department of Computer science and Engineering
Indian Institute of Technology, Madras

Lecture No. # 18

Queuing Networks - 1

(Refer Slide Time: 00:07)

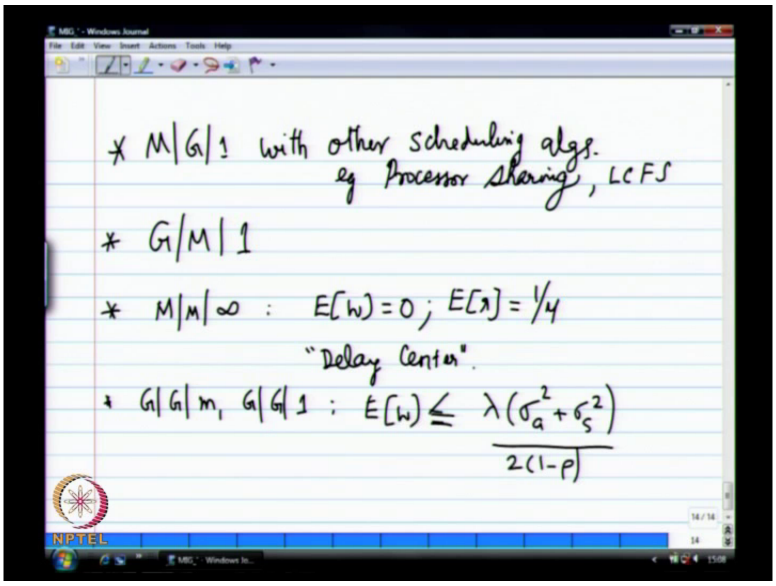


Keywords

- Server types
- Delay centers
- Closed queue
- Tandem queues
- Burke's theorem
- Product form network
- Semi-Markov models
- Visiting probability

Now, there are lots of other variations of this M G 1. We saw with first, this is basically first in first out FCFS, we never seen anything any other discipline, but there are results for things like processor sharing, we did in g p s in g p s, too many g p s. So, process sharing is another sharing discipline. Now basically round robin, with very small time quantum. There also you can emulate, So, you can with get results for M G 1 with processor sharing, We can do M G 1 with last come first out. There are several such variations which we will not get in to. If ever come across those you can simply look at those results and then try to derive them.

(Refer Slide Time: 00:59)



* M|G|1 with other scheduling algs.
  eg Processor Sharing, LCFS
* G|M|1
* M|M|∞ : $E[W] = 0$; $E[\lambda] = 1/\mu$
  "Delay Center".
* G|G|m, G|G|1 : $E[W] \leq \frac{\lambda(\sigma_a^2 + \sigma_s^2)}{2(1-\rho)}$

So, you have M G 1 with other disciplines, other scheduling algorithms. Not all algorithms will achieve get it is close from results, but you do get. So, those sum of this results are summarized in (( )) p f p s. For example, is some let us example is process sharing. So, round robin that we use is like an M D 1, you got it, round robin is basically example list but, we are time form becomes very, very negligibly small, then what happens you never finish every packet. You have to go on serving the packet multiple times that is different between M D 1 and with process sharing and with round robin.

Round robin will it take a packet, if the technically finished it, it goes back to the queue that can be modeled as a queue network and so on, where as in the process sharing you serve bit by bit or by micro seconds by micro seconds. So that packet by always in the queue (( )) last come first serve these are there. Then you have results for G M 1, where the arrival is based on some general distribution, but the service time is not program, which again I am not going in to.

Then there is one special case to of this is M M, M system called the M M infinity system. So, M M infinity means that whenever the customer arrives there is always server waiting for them. So, this is special as it is basically a fixed delay center, right because the customer comes in, what is the waiting time, always 0. You always will get a server, what is the response time, 1 by mu. Therefore, in some of this queuing networking (( )) model system as a collection of queues you simply use a M M infinity to what is called the delay center, to modular so called delay center. Which means there is always fixed amount of delay, the delay there is no queuing delay in this system, right.

There is no queuing there is always a server available. So, in some cases we use that (( )) infinity system. In fact, if we look at this M M infinity system again may be queues final example you can show that the number of customers in the system at any point in time right is a poison distribution. So, p n will be a probability for M will be a poison distribution.

So that is not, so that is generally true for M G infinity, not just M M infinity and then you have G G M, G G 1 etcetera.. So, if you look at G G 1 system there are some bounds in terms of the waiting time. It is not, you do not have close form solutions, but there are at least some decent bounds and tell you let it delay will be less than this is. So, for examples for G G 1 the waiting time E f w is bounded by, so if you know the standard deviation of the arrival process, arrival the inter arrival times and the standard deviation of the interval departure times of this inter service times, service then you can say this is upper sorry what is this, it is a upper bound.

Anyway the derivation is there, if anybody wants some of these M G 1 and all its variants or in our data networks this book and this chapter this all section 3.5, the lots of all other results, reservation, exhauster, exhaustive, gated, polling all those things are there, which lets, more useful if we go to a networks only course where you want to look at reservation also. So, I have a reservation based maxes protocols, where there is the reservation phase, units make their pack customers request for some certain number of slots, then there is the reservation, then the actual data transmission phase, right. So what is the delay in such a system compared to t d m or (( )) and things like that.

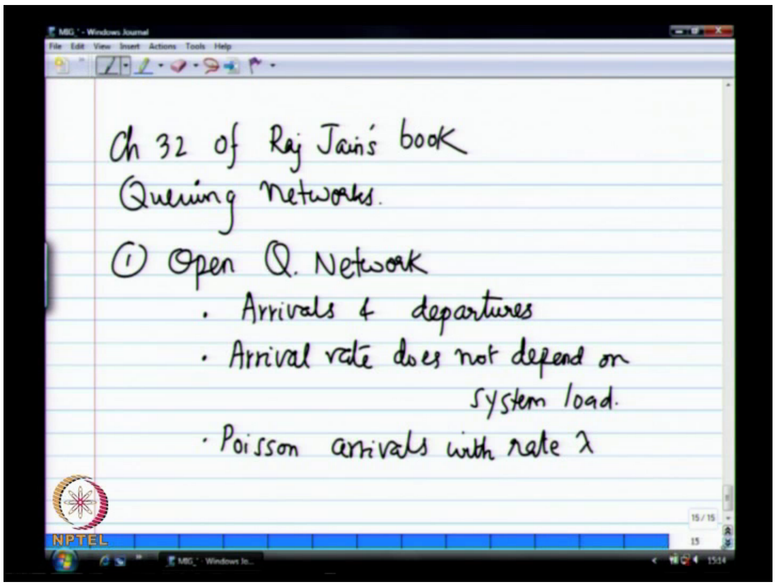
So, the next assignment that you will see soon will be related to simulation of some of this basic union system. We have seen a theory now, so I will (( )) time will present one (( )) simulation packet. That is, it will come up after I do make queuing networks for may be between I will try to talk about that. So, that way we will actually try to implement some of these queues and then various, it is various parameters. We can do lot more variations in the case of simulations base system for example, arrival rates, I can change that I can set each arrival rates whatever I want to and I can do for queue arrival rates. Here, we should assume that all arrival rates are the same in the multiple queues and so on.

Now, we have think pretty much done with system, we started off with random processors, variables, marco change, and then finally, and with how marco change can help in queuing systems in a way actually went to M G 1 were there is no real marco change at all. We can run it M M M 1 with marco business, but any way I can think useful tool to you know. So, we know how do handle single queue more else, but real system is like systems are never just a single queue is always a connection of queues, flow packets flowing from one queue to another queue and then finally, varying the system it is a simple transaction processing.

For example, if we make a want to do on online transaction with the bank, then you will be put it one queue very well there is in one system that is taking care of your request then you go on to other queue that will try to access the account, where is the queue that is involves because there will be several request for the account information queue. Then there will be another system there is validating verifying and so on.

So, we can authentication, getting the account, then do the actual transaction. So we have a series of such queues the table packet actually has to, every transaction request has to go through. So, that is our next topic right. There are two kinds of networks of queues, one is so called open network, other one called as close network. So, we will try to look at both right, so from this point onwards I am getting in to our Raj Jain's book, this is done, I am coming to Raj Jain now and Raj Jain we are going to look at 33 34 and so on. So, this is actually chapter 32 gives the introduction. So, we will start with 31 is all these queuing results.

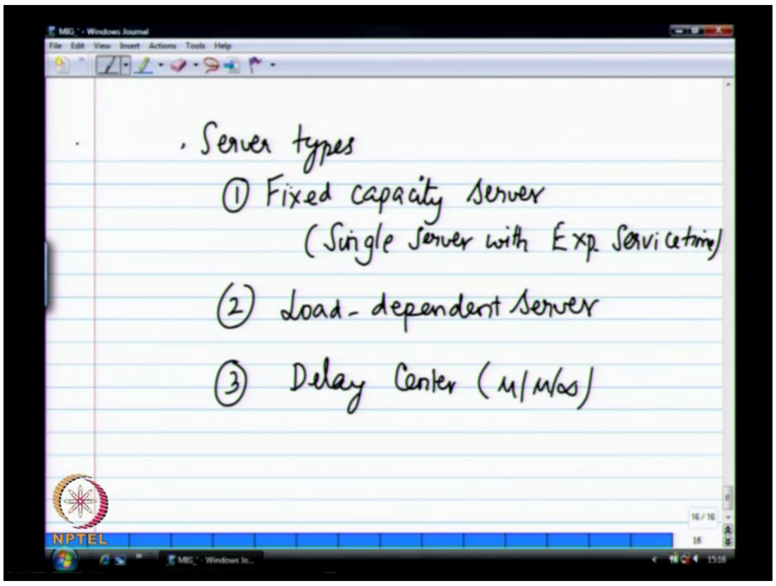
(Refer Slide Time: 08:08)


Ch 32 of Raj Jain's book
Queuing networks.
(1) Open Q. Network
· Arrivals & departures
· Arrival rate does not depend on system load.
· Poisson arrivals with rate λ

So, 32 is what we now get started with, queuing networks or networks of queues. So, the first classification is a open network. Some definitions and then we will go to some derivations later. So, this is systems where there are arrivals and departures, and you might wonder why we are saying arrival and departures but look at the close system where there is technically no departure, you are struck. Once you have the system for ever going between queues. In this case there is an expressive arrival, there will be an in or out for the queuing network, and the arrival rate, it does not depend upon the system load.

So, the arrival process is independent of the system load. So, which we in the other case the queues there, queues that we saw discourate arrival was there was which means that depends upon the system load and number of customer. So, in this case assume that it does not depend upon that. So, just independent set of arrivals set keep coming in at some with (( )) and we are also again for the sake of convenience, assume that these are poison arrivals with rate lambda.

(Refer Slide Time: 10:05)

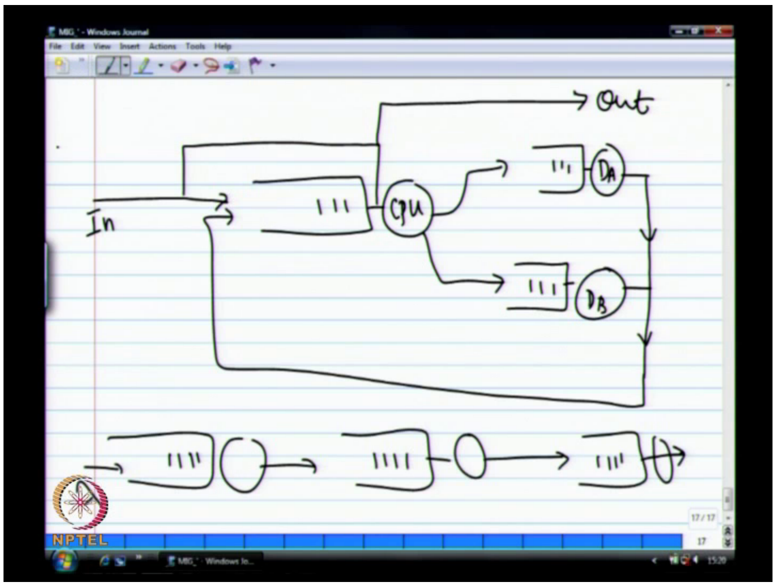


- Server types
  1. Fixed capacity server
     (Single Server with Exp Service time)
  2. Load-dependent Server
  3. Delay Center (M/M/∞)

And the servers we classify them in to three types, one is called as fixed capacity server. This is basically a single server with exponential time, single server with exponential service time, or we also called as a load dependent, there is another category called load dependent server. The load dependent server means as the number of customers increases the service capacity with the system increases which is basically where M M 1 is in different way, there is only one customer, the service rate is mu, whether two customer service rate is 2 mu, 3 3 mu and so on, up to M mu.

That is basically categorizing and that is way it is called load dependent as a number of customers it increases technically there are all these in queues server is available, but the effective service capacity is not constant. It varies up and down with the number of customers being serviced that is they called load dependent. Load dependent is orders we want kindly look at that and then other so called delay centers. This is a as queue in the system which was basically tech theoretically infinite servers, but practically it is simply delay center. You go in and out that is the constant amount of times it will spend a teacher or the service time, the one over mu will be the average service time (( )) each of this servers. There is no queuing in any of these places.

(Refer Slide Time: 12:07)



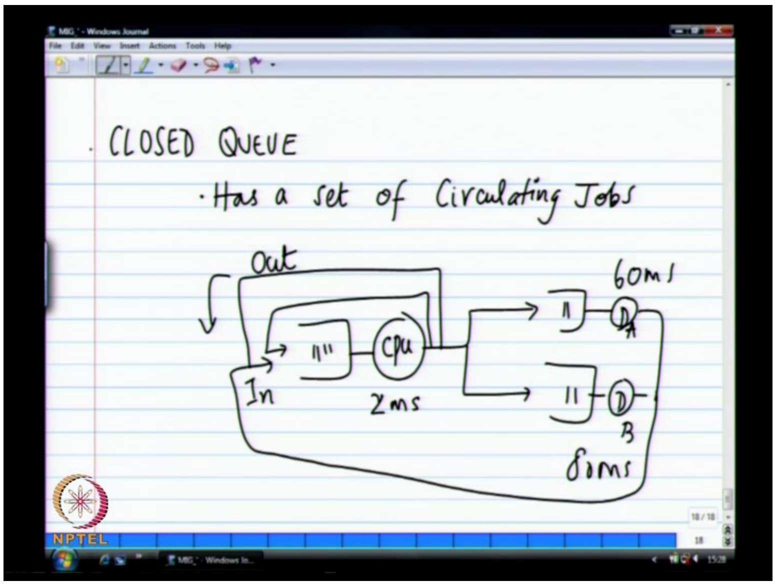
So, the network will have whole bunch of inter connected queues. This system we kind of have seen before. So, there is a cpu, two disks and then there are certain number of jobs lined up. Each of this disk including the cpu. This is our classical computer system model, actually the codes is called performance ratio computer systems. This is your we star right what we really are trying to look at.

What is this missing so I have cpu queue, jobs arrive from outside, then they can go to one of these two, something as (( )) like missing to work everything is ok. Then, what are the two missing lines, usually we put right you like to assume that just cpu, job on the cpu takes multiple iterations or multiple quantum of service before it can finally leave. So, therefore that is that line, or it is possible that it finish the cpu service and departure. So, in the queues what we saw is the risk anything, but there is no queue there, there is only the state of a process going to here getting service to cpu or waiting for a figuring service to the disk A, disk B and so on.

So, that is in an example of an open queue. So, why is it open? Customers come in they get serviced and there are these loops that make you go back and forth, do this and then you finally leave this system, so that is the typical system. So these are all the various processing stages which you go through and you might visit DA 7 times, 8 times it really depends upon the nature of your each job that is joining on the cpu. Sometimes you will you see your disk intensive, sometimes you are only in cpu intensive. So, cpu bound, io bound all that stuff comes in. So, this is an example of the open networks. So, the question is how do we solve this? How do we determine the know through put response time all those things in such a network. The simplest open network is simply a tandem of queues, where you go from one server to another server.

You arrive with rate lambda, you go to another server, spend time there, but there is no feedback in this system as such. Sometimes you go to for getting permission to travel abroad, you go the hod, hod's filed and then it goes to dean a r, the dean a r filed, then they will forward it that to dean the financial saying pay this person, then it goes to the dean finance queue. So, there you are not really going back and forth, sometimes you will, but normally you do not, you will just linear paper gets to you at the end so you are done right. So, for such a system again can we find some closed form solution? So, there is the example of open system.

(Refer Slide Time: 15:45)



Then the next is a closed queue, where your system has a set of circulating jobs. So, the jobs are permanently, technically they are not permanently in a loop, but this is for example, we look at the system with multiprogramming. So, let us see with multiprogramming what happens. So, we again have the cpu queue, there are some jobs waiting here, you can go to either one of these two queues. I have seen specified the transition probability right what is the probability this jobs will go to, like we saw in our marco change. Right, let we will come to later on.

So this is one disk, other disk and after the disk service, they always come back to the cpu and then there is a, (( )) you come, so this is, So, from the cpu then you know you can you come back to the cpu and so on, but what we have to designate is, we have to designate couple of lines in this close networks as in and out. So, basically this case what is it that your trying to find out, we are trying to find out, we know some properties how long will it take, how long will it spent in this D A? How many times will you go to each disk size? This type the process will take 10 disk accesses or 20 disk accesses and how much time will does each disk access take, with all of that. Then this, the through put is defined as a rate number of jobs (( )) from out to in, what is the number of job set of entry.

So, the assumption is that multiprogramming, if one job finishes, there is always the another job to replace it. Assume that you are always having a steady set of jobs to be, to enter the system. So, as soon as one job finishes next job is basically replaces the old one that is all.

That is why your technically it is look, it looks like a close system. Any finishing job that leaves the cpu is replaced by another new job that is waiting to be run in the system. Now, what is special here? So, in the case of multiprogramming we know that I can have only four jobs, let us say only one job is running in the entire system. What will happen? It comes to cpu, goes to disk, goes to disk, runs n number of lines, but there is no queuing there at all.

So, refer the delay is simply the waiting time and the service time are reach of those center. Then you will add in one more job, even then you might not see any queuing delay, because when one job is on cpu and other job is on the disk, that is your multiprogramming right when the cpu idle, let some other job run on the cpu. So, then your utilization will slightly be more. Then as I keep on increasing and what happens is that slowly this disks the queue starts filling up and then the queuing delay starts dominating your performance.

So, if you put 50 jobs, what will happen, all user sitting in this queues, there is only a steady, only about say that 5 jobs can be finished per minute. So, there effective throughput is only the number of jobs finishing per unit time. That is because each jobs goes through several of this queues and finally, it finishes after says several measures disk a and several measures disk b and then request cpu in troubles it finishes. So, we look at the effective of the throughput of the system, what is the system were by delay it is going to be bounded. So, initially throughput will increase up to some point, then after that point your throughput whether steady or actually start coming down. When we start think about talking about crashing and so on when the more jobs in the system then you simply waiting at these queues for service, no jobs effectively finishes.

So, it take a long time for every job to finish in which case throughput systems starts going down. So, next 5 jobs per a second will have only 1 job per second because you are all sitting this waiting for service. So, that what is the speed, I guess what is the optimal point in terms of the number of jobs that can be in the system? That is what we tried to find out with this kind of close network analysis. So, that is we look at some mean solutions for that how can we do that, that is why closed to systems come, so that understood. So, there is always is some new job to replace an old job, that is the basic idea here right. What is effective number of job completions per unit time? That is we are trying, we are interested. From the users perspective it is not like your job is stuck forever, it just from system perspective there is always a job that replaces another job. It is just like your IIT system students leave, but then we always have your hostel room is filled up. So, there is a permanent perpetual occupation of the hostel rooms.

So, it is technically we can state all the fourth years as first years students or next year, first year students or next batch of incoming first year students. So, usually you fix n for given n what is throughput so you will, you will given n number of users as an input what is the throughput of the system? I put in thousand customers in the system, will I only see 5 customers finishing per unit time, so what is optimize. So, let us say put in only 1 customer, I will have 1 customer finishing per one unit time, I put 2 I get 2. So, this keeps on increasing.

So, when I go to after 6 customers, I can have 6 customers finishing per unit time, but when we will go to 10 customers I come back to either 6 or say 7 or something like that, in which case I am no longer optimal, I am having more customers, what happens the queuing be delay simply increasing beyond the point. So, that is what we want to find out this optimal n for operation of a given system. How many jobs should I keep circulating in the system such that optimal throughput of the system is achieved and the queuing delay is basically minimized.

So, there is some point when which queues starting building up, what is the value of n. I should not operate at that level otherwise what happens is that every jobs response time, whole response time will be very large. We will put in twenty jobs in the system. Let say we take our computer center, you will submit jobs for running for that it will take some minutes, several hours together, everybody submits the jobs they have to limit the number of jobs they are currently running under the system, right. That is, it is batch system, where in you take batches of job at a time, but what is the optimal n? You do not know. So, you try to measure that based on some long rounds, long times statistics, how long does each job take, how long this, how many accesses that each job requires on average based on that you tried to optimize your multi, multiprogramming. That is one example, in the case of computer system that is.

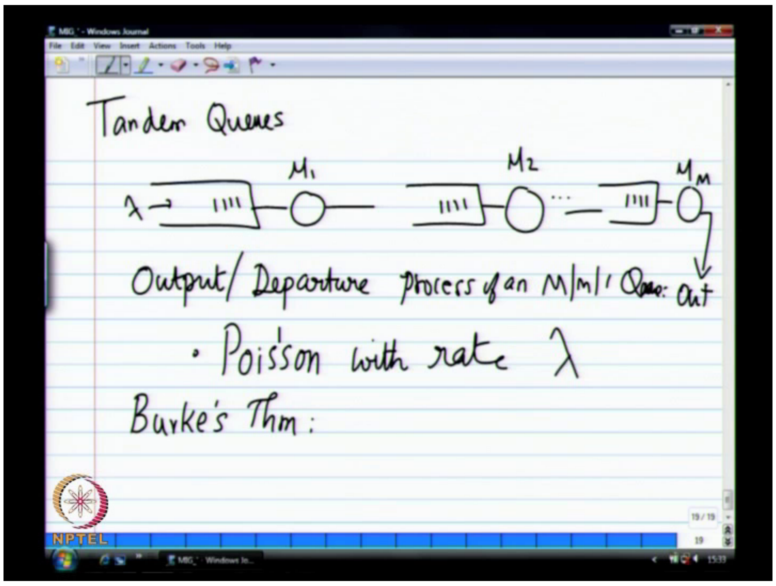
(( ))

These are three separate queues. So, what is your like to do is what is effect queue, the number of packets or number of jobs that are technically travels in this out link going in to the in link. That is the defined as the micro throughput of systems. So, how do I find that out, given that each cpu I am spending say you know 2 milliseconds here and the disk I spend the 60 milliseconds, on this disk I spent 80 milliseconds.

So, the job could be writing some data write in disk A then will write in to disk B for some other applications and so on. So, given this kind of requirements, what is the effective number of jobs per second that they can sustain in this system A and system B. So, here, that is, technically that is self loop of back going to the ready queue is not here, as if that I go here I do these things, I keep repeating like this then I finally, leave this system. The extra we could have an extra loops also, but that is not the outline, you have defined one of this lines as the out in line, and then say on this line what is the rate of charge for (( )). Only when a job, if we have another extra line here, which is simply bringing back to the cpu queue itself, I define this as a line where I would exit this system when the jobs finishes that is basically travels in the outline.

So, what is this from here to here is because this is my replacement rate right, depending on the number of jobs I finish I will able to (( )) what is that rate, that is what we are trying to get to. That we will see then we get to the (( )) itself, analysis then we will get a better class for that and then we will also implement these things, for (( )) and once we get basic single queue done in networks of queues is very simple in a stimulator, then we can run it and then see. What we get it. Other questions, before we start looking at close form solutions for this system.

(Refer Slide Time: 24:23)



Tandem Queues

Output/ Departure Process of an M/M/1 Queue:

• Poisson with rate λ

Burke's Thm :

So, let us look at our first the tandem queue and see how we can derive that probability of occupation. I have, I will come back later to this so, I can also have hybrid queues some parts of this systems is closed, some parts of it is open, let us not confuse our self right now, we will just leave this as open and close. So, another is tandem here. Now we will look at the simple tandem queues M M 1 queues. So, the series of M M 1 queues, each of them is having this service later. So, customer our packet arrives gets service to the first queue with rate mu 1, second mu 2 and so on.

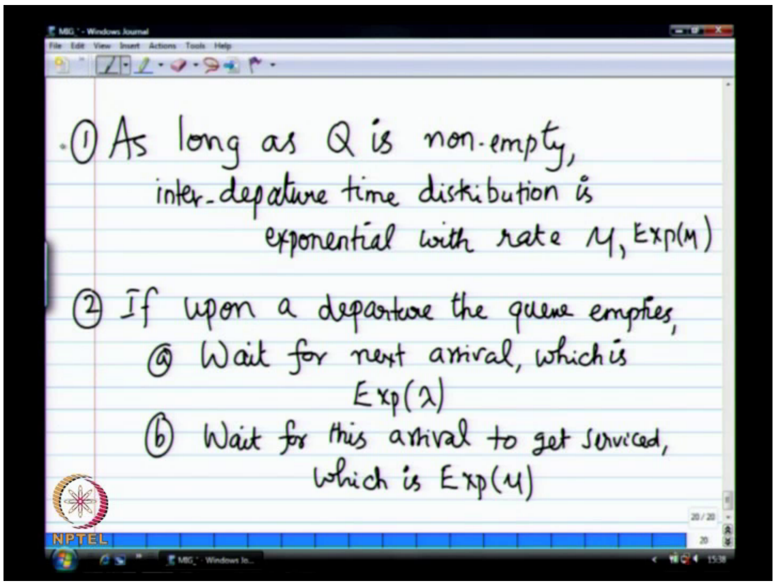
And there is no other external, you know, input of this system, only entry point is that lambda, at the first queue, no cross queues. So, to analyze this and so basically we want to find out the probability that there are n 1 customers in queue 1, n 2 in queue 2, n 3 in queue 3 and so on that is once we get there then I can derive the number expected number of customers in the system and all my response to the everything (( )) So, the first thing it people have to figure out is what is the output process of a M M 1 queue what is the (( )), packets are departing the queue, what is that process because that process is the input of the second queue.

So, what is the departure process of a M M 1 queue. Of course, this queue has to be stable. So, otherwise not attend this question.

So, what could this process be? M M 1 also poisson with rate mu 1, say I am only a generating mu 1 is a 1000 million packets per second, only 10 packets per second are arriving entry in the queue. Will my departures process be a poisson pass with rates 10000 or 1 million packets per second something like that. So, your input equals to your this other steady state, number of customer entry equal to number of customer leaving per unit time, and it so happens that it can be shown to be a poisson process with rate lambda. So, this is what is called as Burkes theorem. So, this is the classic result. In this person was also in a t n t, a lot of the results queuing results it came from a t n t folks because a t n t labs, it is very clean and getting the layers and so on, the telephone network. So, this is one of the earlier services.

So shall we try to prove this? So, how do we prove that? So, we will look at two cases, one on the queues is the never empty, the queue is always full, what will be the departure rate. The queue is always full, the queue is never empty. What will be the departure rate? What will be the inter departure time, service time, inter departure time distribution? Let us do this one at a time as long as the queue is non empty.

(Refer Slide Time: 29:08)



① As long as Q is non-empty, inter-departure time distribution is exponential with rate μ, $Exp(\mu)$

② If upon a departure the queue empties,

ⓐ Wait for next arrival, which is $Exp(\lambda)$

ⓑ Wait for this arrival to get serviced, which is $Exp(\mu)$

So I am looking at the departure process. So, I want to know the inter departure time distribution. So, inter departure time distribution will be what, in other words the time between two customers departing. So, this queue is always having somebody to serve. (( ))

One by service time, it is simply exponential with rate mu. Now, it is a exponential with rate mu, when the queue is always full, the (( )) each customers comes, get serviced, what is the time, one hour mu, the service time is exponential with service with rate mu. So, between successive customers departing the system the distribution will simply be exponential with rate mu.

(( ))

No, this is just the process, is just the process, sorry only one queue. We are not going to your forget tandem no no let us not before I go to my tandem I first I need to first of all I am looking at this system at this point here.

We are just looking at single M M 1 queue, 1 M M 1 queue, to solve this I need to know this results then I can solve this second one after that. So, what is the process coming out of the first queue because I need it as the input process for the second queue, I do not know that, I am stating that it is poisson. Only if I know this is poison then I can solve all of this independently, but I need to the first of all prove that it is poisson. So, this if the queue is not empty inter departure time distribution is exponential with rate mu. Now, let us look at the scenario where one customer has just now left. The last customer in the queue has been serviced therefore that guy has left. Now, we need to find out the next departure. So, one departure that is the last departure has happening, you waiting for the next departure to take place. So, when will the next departure happen, right now the queue is empty.

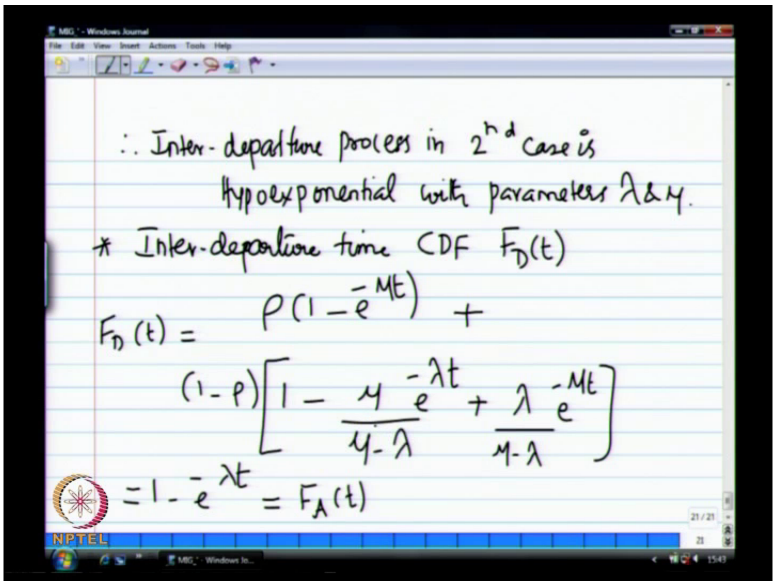
So, will write this, if departure the queue empties. So, we will have to wait for two things, what is that, wait for the next arrival. So, what is the time for the next arrival that is the exponential. So, the inter arrival time is exponential otherwise it is memory less. So, whatever point in time I say what is the arrival distribution, it is always this inter arrival with exponential with rate lambda. So, wait for the next arrival, which takes with this we say that exponential lambda. Then so, for this arrival to come I have to wait for EXP of lambda and for this arrival to leave. So, this come to empty key that is why we will get to service all already. So, therefore the time to next departure, so I will also wait for the, this,

(( ))

It is poisson with, yes poisson, but the inter arrival is exponential, when the inter arrival time. So, poisson process, arrival process, the inter arrival time is exponential with rate same lambda. Wait for this arrival to get serviced and depart. And what is that. So, when the queue is empty we have to wait for one arrival, then that arrival has to depart.

So what do I have now… So, what it is this A and B combined defined. It is a hypo exponential, there are two exponential processes back to back, there is no queuing at all. There is one of the rate lambda, one of rate lambda, one of rate mu therefore what is the so what is the mean Now, the total response time the total waiting time for the next departure. Now, this is my inter departure time. Inter departure time when the queue is always full is exponential with rate mu. When the queue just got empty it is hypo exponential with this lambda plus lambda mu combination. So, now let us do that.

(Refer Slide Time: 34:46)



$\therefore$ Inter-departure process in 2nd case is hypoexponential with parameters λ & μ.

* Inter-departure time CDF $F_D(t)$

$$F_D(t) = \rho(1-e^{-\mu t}) + (1-\rho)\left[1 - \frac{\mu}{\mu-\lambda}e^{-\lambda t} + \frac{\lambda}{\mu-\lambda}e^{-\mu t}\right]$$

$$= 1 - e^{-\lambda t} = F_A(t)$$

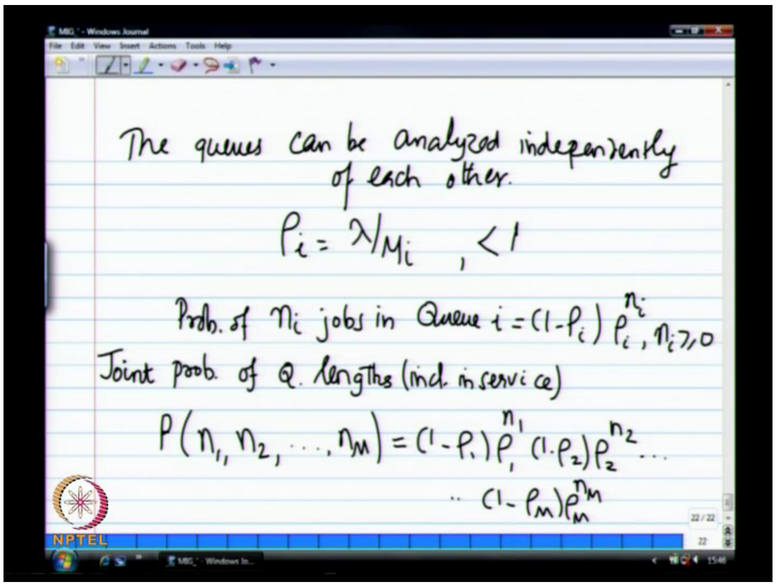
So, therefore, the inter departure is hypo exponential. With parameters in the first case is lambda, second is mu. Now, what is the probability that. So, there are two cases. Now, one is exponential mu, one is hypo exponential lambda and mu. So, what is the probability that queue will encounter exponential mu, rho. Rho is the probability utilization, probably the queue is always occupied, 1 minus rho is p naught probability that the system is going to be empty. So, therefore, effective inter departure distribution is simply probability of q being empty into this f of. So, the inter departure time CDF, we call it F D of t is given by rho into 1 minus.

So, this is the first part, 1 minus e power minus mu t, then this is 1 minus mu into this will have to go back in, so, 1 minus lambda 1 divided by (( )) lambda 2 by lambda 1 plus lambda 2, e power minus lambda 1 of t and so on. We can go back and (( )). So, that is 1 minus mu, lambda 2 minus lambda t plus lambda by mu minus lambda e power minus mu t. This is the F D of t, the queue the CDF of the hypo exponential. And this we have not done the computation, but we can do that later and then actually done the worked out, but finitely it works out to 1 minus (( )) lambda t.

So, before next class verify this because the fact that rho is their right. So, the rho is rho lambda by mu. So, I have not checked this, I just found this proof on the web and I said, well it should be correct but, looking at a time kind of now I am worried weather this is going to work out or not. There is 1 minus mu that is there and then I guess, we can make mu by mu mu 1 minus, I can take out mu there and make it 1 minus 0 and here also, well we will check, take it out. So, this is your F D of t 1 minus e power minus lambda t and that is basically your arrival process. If F A of t is arrival process arrival inter arrival process CDF therefore your departures are also a poisson with rate lambda.

So, this result is known, this particular derivation I have to check. So now, I know that whatever the process departure process of the first queue is also a poison with rate lambda it goes to the second queue. Now, I can analyze all these queue independently. So, what is the total time spent is the time spent in the first queue plus time spent in the second queue. So, these queues are basically now going to be independent; again some of the proof. We have just sort of stating (( )) we can check we will, couple of minutes or so.

(Refer Slide Time: 39:40)


The queues can be analyzed independently of each other.

$\rho_i = \lambda / \mu_i \quad , < 1$

Prob. of n_i jobs in Queue $i = (1-\rho_i)\rho_i^{n_i}, n_i \geq 0$

Joint prob. of Q. lengths (incl. in service)

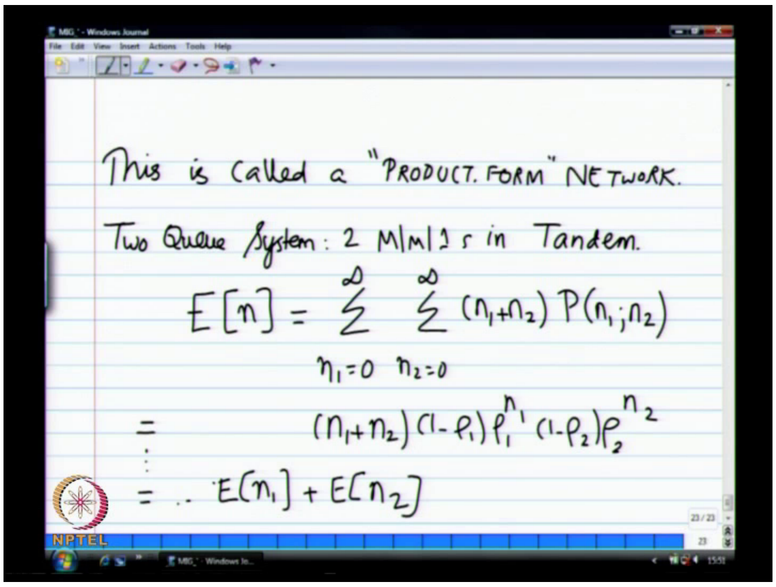
$P(n_1, n_2, \ldots, n_M) = (1-\rho_1)\rho_1^{n_1}(1-\rho_2)\rho_2^{n_2}\ldots$

$\ldots(1-\rho_M)\rho_M^{n_M}$

So the queues can be, so that result is what, let this analysis that they can be analyzed independently of each other. So, if I define rho i is lambda by mu i, the probability of n i jobs in queue i is the same thing, rho to the n to 1 minus rho. So, it is 1 minus rho i and will assume that this is, you know, less than 1. And then, if you look at the joint probability that there are, you know the joint probability of queuing lengths. Queuing length is basically a number of customer not just the length including. That is basically, what is the probability that if there are n 1 customer in the first queue, n 2 customer in the second queue and n m customers in the third queue, it is simply 1 minus simply the product of all the P m's number of customers, they are independent of each other.

So, when there are 10 customers in queue 1 it does not matter that there might be 20 customers. The probability of having 10 and 20 is independent in each of those queues. If it is dependents it becomes a problem. That is why this M M 1 assumption is useful. That, we can show that they are dependent of each other and therefore the joint distribution of having so many customers (( )) is simply the products of all the, so this is what is this, this is number of customer having n 1 customer in queue 1 into n 1 n 2 customer in queue 2 and so on and simply a product. So, this is what is called as in queuing network results is called a product form distribution. This is called a product form network.

(Refer Slide Time: 42:22)



This is Called a "PRODUCT. FORM" NETWORK.

Two Queue System: 2 M|M|1 s in Tandem.

$$E[n] = \sum_{n_1=0}^{\infty} \sum_{n_2=0}^{\infty} (n_1+n_2) P(n_1; n_2)$$

$$= (n_1+n_2)(1-\rho_1)\rho_1^{n_1}(1-\rho_2)\rho_2^{n_2}$$

$$\vdots$$

$$= \ldots E(n_1] + E[n_2]$$

So, whenever I have a network which has this product form nature I can analyze the queues independent of each other.

(( ))

Summation of all this, summation of all the delays; here, there are we see a tandem queue such as, this is values have a since m p l s path, only thing there is a some cross traffic also coming in, there is extra lambda is feed in, but in this case because your sharing the link your links being shared, you know. So, if you can somehow abstract rate away if you doing for examples conjunction controlled analysis is going through several queues t c p s for example, goes to through several queues, no one to find out the end to end delay. Some more approximation have to be done, but this is usually some for starting point for simple things go back and selectively repeated and so on you can use this as for this initial set up.

So, we will do one small example and then we will stop. So let us look at the two queue system. So, two M M 1 is in tandem. So, what is the probability, what is the E of n. What is E of n, E of n is a total number of customers in the system, total number of customers in the entire, in the tandem queue system. What is that going to be? From basic we did E of n 1 we said, what so this is basically the, what is the there are n 1 customers in the first queue, n 2 customers the second queue, then n 1 plus n 2, it is the probability that there are. For every n 1 what is the rate given, what is the value of n 2 for some value of n 2 n 1 plus n 2 multiply probability of (( )) we did n p n for the basic case. So, this is simply n 1 plus n 2, p of n 1 and n 2. And what is that, now I know that this is product now. This is the product form solution therefore, I can replace that with, this is 1 minus rho 1 rho 1 n 1 out n 2.

So, how do you fact solve this, for the so, what will we get the end of it. E of n 1 plus E of n 2. So, that I will let it work it out. So, first case make n 1 is a constant it is some some over n 2, then keep n 1 constant you will find out that you will end up with something then sum some it over the outer index n 1 and then finally, end up with E of n 1 plus n 2. So, that is, the show is we can simply add up all the. So, once again we get E of n the E of r in each queue and to get the total delay is simply do a E of r 1 plus E of r 2 and so on that is all. So, this after derivation should come to questions. So, in general what we will see, you know there are some classes of networks for which this product form the network conditions apply and that is why people have tried to do.

So, if I just take a single tandem. Yes it is a product form. What if I have a arbitrary set of you know open networks, open queuing networks is that close form or this product form then that is also a product form or arbitrary network of closed queues for all of these the exponential service time. So, arbitrary network of M M, even M M M is also okay. Mu network is also a closed form. So, we say we started with this tandem is closed product form, open network with M M 1 queues is also product form open and close with M M 1 is also product form.

So, these are been results that got proved in the 70s and 80s as people try to expand on where this product from can be applied. Then there is the classic papers which is 75 or so that says there are say, even for other types of disciplines, not just a FCFS even with multiple classes of traffic you can have all of these things done. So, we will start of, just go through those main results then we will look at actually techniques as to how to solve some of these and get throughput values that we want in the system.