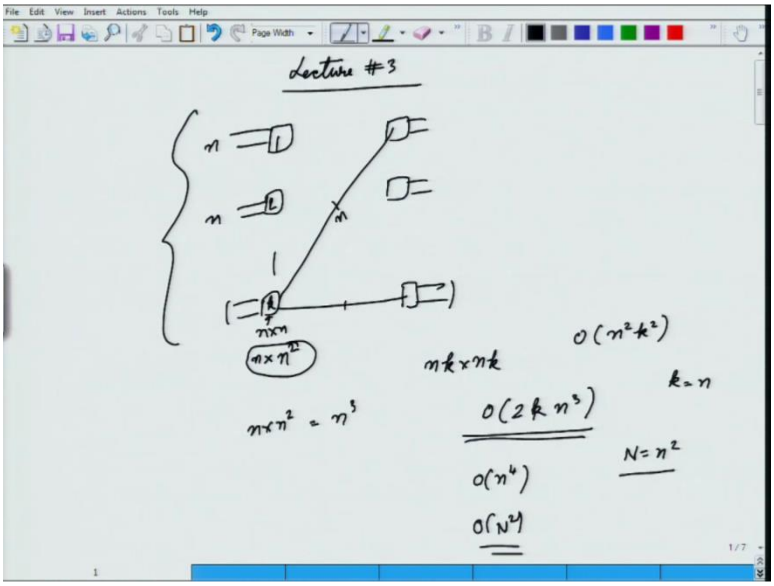
Digital Switching
Prof. Y. N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture - 3

So, we will continue from where we left yesterday, in the previous lecture, what we did was, I was talking about basically how to build up a larger dimensional switch or larger size switch using a smaller size basic switch elements. And in that sense, what we want is the preservation of the certain property, which we called as strictly non-blocking property. And of course, the question was whether I can reduce my cross point complexity in some way or not. First thing was use of creating larger size by using a smaller thing, and further deducing the cross point complexity.

(Refer Slide Time: 01:01)



So, what we did was I took a case, where we were trying to build up using two stages. And I took some switch with certain basically kind of a small N as the input ports, and with them we build up a inter connect by using, if I am going to use N by N I said that one can go to 1 link can go to each one of the switch in the output stage or in the second stage. But this was leading to a blocking situation. So, if certain inputs are free, and as well as output is free I cannot setup path between them in certain situations.

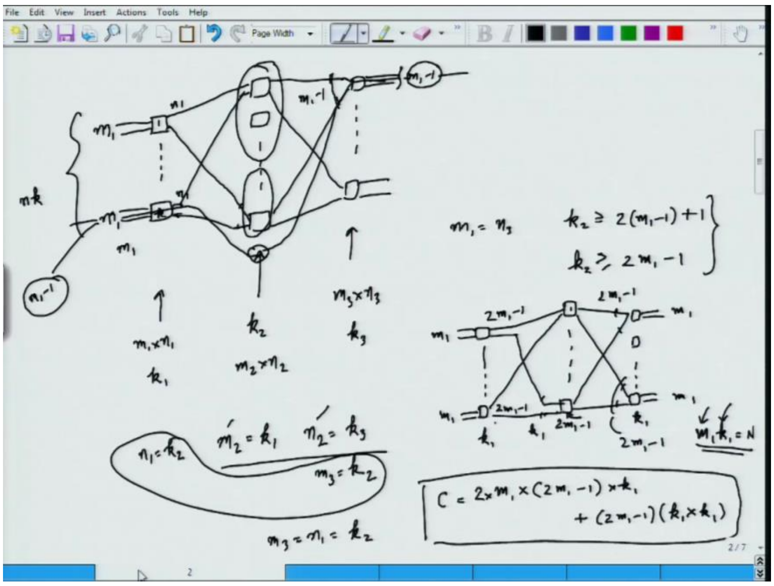
So, in order to resolve that, what we did is we actually start using N by this N by N square, kind of switches where actually used. Instead of N by N, and in that case this there were actually N wires which were going out, and this actually means, I need to put N wires to each one of these. So, all these N can be connected to any one of these N, without any blocking to happen.

So, for this number is going to be smaller than N then, it will not be possible. Because, whatever is left over and there is one, which is freer cannot be connected only part of the connections can be made. And of course, we figured out that the complexity still remains, because if you have only there are k such elements. So, you require you have switch size of N k by N k, which usually will require if I build up a single process bar will require N square k square.

In this case, I will have a switch of N by N square, which becomes N cube complexity, and I have actually two k such switches. So, I will be requiring o 2 k N cube kind of thing and then of course, if I actually take from here, you can actually clearly see that, if I am actually making kind of k will become equal to N in that case both will be o N rise power 4. Since your capital N is nothing but technically equal to N square. So, this is nothing but complexity of o N square only.

So, that is what we did last time, and then I thought, we can go to three stage inter connection. And I defined a clause network in that case, and I also defined, what is going to be strictly non-blocking structure? What is reasonably non-blocking? What is wide since non-blocking? And what is going to be blocking structure?

(Refer Slide Time: 03:46)



And then, of course, I defined the clause network, which was you will have some input. So, example N input ports, and there are k such switch switching elements. So, total number of incoming ports is N into k, and we were connecting in the middle stage many such switches. And we have to find out, what we will dimension? Now, each switch should connect to each one of those switches.

So, I will be actually having something like this. So, I might end up in choosing some number here some middle stage. So, in fact, the way I defined it was it was not m it was m 1 m 1 this side was N 1 this is also N 1. So, all switches where which were here was m 1 by N 1 switches. And I was using k 1 of them then I was using the middle stage k 2 switches, which was of m 2 by N 2 size.

And then, there was a third stage configuration third stage switch, where I will be using m 3 by N 3 and there were k 3 such switches. Condition is that only free incoming ports are available only in the 1st stage, and only free outgoing ports are available only in the last stage, which is the 3 rd one in this case or output stage. None of the incoming ports and outgoing ports in the intermediate stage should be left free, they have to be connected.

So, in that case the number of because I need to connect everybody here. So, this number, which actually means that m 2 has to be equal to k 1. Because, k 1 actually lines will be coming and connecting on to this. And similarly here, the number of inputs everywhere will be m 2 will be equal to k 1. And similarly now N 1 has to be equal to whatever be the number of switches in this case will be equal to k 2. And this also implies, because same structure will be applied here.

So, you will have N 2 will be equal to k 3, and m 3 will be equal to k 2. So, you can see this condition. So, N 3 will be always equal to N 1 is equal to k 2 this condition will always be satisfied. And I said this actually switch can become strictly non-blocking. And I took a very simple example this is this will be later on proved as a clause network this is actually, a clause network. We will use a clause theorem to find out, what will be the condition for strictly non-blocking nature of this particular switch.

So, if m 1 minus 1 lines are already busy occupied here, the same is true at this place that m 1 minus 1 lines are occupied here. There is only one line, which is left in which I am interested there is only one line which is left in which I am interested. I want to connect these two, in worst case scenario now this is out of these lines, which I am putting here m 1 minus 1 are already occupied.

They might be using certain middle stage switches for connecting to some incoming ports. From this side in worst case scenario these can be using a different sort of switches there is no overlapping between them. If I had one more extra this can always be used to setup the path between, this free input and output link. And switch will be always a strictly non-blocking switch, it you no need to disturb any existing connections, which actually implies that I need to have if I am going to take a symmetric case where by N 1 is equal to N 3.

So, you will find that I need k 2 should be equal to or greater than or equal to 2 into m 1 minus 1 plus 1, and that is the condition which will come from this thing. And this is nothing but 2 into m 1 minus 1 this should be the condition and then, k 2 will actually decide what will be the value of N 1 and N 3. So, strictly non-blocking switch will look something like this, it will have m 1, which is going to come in.

And you will have these ports which is 2 m 1 minus 1, and this will be going to various switches here, m 1 ports there are total k 1 such switches. So, again this dimension will be 2 m 1 minus 1, total number of switches will be 2 m 1 minus 1 the number of ports here will be k 1, and since I am actually taking m 1 is equal to m 3.

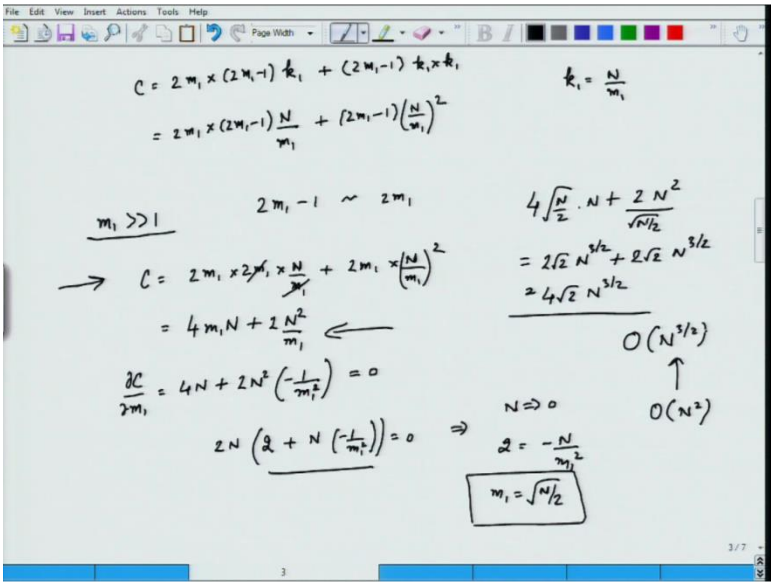
So, I will call it again m 1, this will be also m 1. And since you will have in this case again these lines have to be 2 into m 1 minus 1. So, number of middle stage switches will be these many, and these of course, because I want symmetric thing. So, I will be also taking this thing as a k 1, k 3 will be equal to k 1 that is the symmetric case, and this will be the condition which will be satisfied. So, m 2 will be equal to N 2 in this scenario, this switch also everything is actually symmetric in this case, and this will be strictly non-blocking switch.

Now, what will be cross point complexity for this particular switch? So, if I want to find out cross point complexity number of cross points, I can very well estimate that for this particular switch I will be requiring m 1 multiplied by 2 m 1 minus 1 number of cross points. And number of switches, which are here is k 1 and exactly similar stage also exists in the third one.

So, I have to multiply it by 2 and then, what I will have is; how many middle stage switches will be required is 2 into m 1 minus 1, what is the size of the number of inputs? Which are coming into this? This will be nothing but k the number of inputs which will be coming in, and k 1 will be going out. So, this is what will be the number of cross points.

So, we can try to optimize this, we have to first of all find out, what is going to be the value of because the total number size is actually now N. N 1 into k 1 is, what is yours is a constant, and I need to find out what are the optimal values for this. So, I will write everything down in form of m 1, and we will try to optimize with m 1. And then, find out what should be value of m 1 for optimality? Basically the minimum size of the number of cross points delta c. So, for this let me now first of all repeat whatever; I have written.

(Refer Slide Time: 11:32)



So, this is the total number of cross points. So, I will now replace all k 1's by N by m 1. So, that I have only one variable sitting in there. So, then I can take a derivative. So, k 1 will be N by m 1. So, this what will get, we can also actually assume that let m 1 be much larger than 1. So, that I can now make an approximation 2 m 1 minus 1 can be approximated as, 2 m 1. This actually simplifies my life.

So, approximately cross points will be now, varying as 2 m 1 into 2 m 1 into N by m 1 plus and now, solving this. This gets reduced I will have 4 m 1 N plus 2 N square by m 1. Now, taking derivative delta c over delta m 1 for the optimal value of m 1, and then we will put that m 1 value and find out the total number of cross points approximately And what is going to be their complexity.

I can now actually keep this particular assumption true, and then use this expression for finding out the number of cross points. So, once I take the derivative this will turn out to be 4 m plus 2 N square minus 1 over m 1 square, and I can make it equal to 0. And this will give me… In fact, I can now reduce this will be 2 N I can take out.

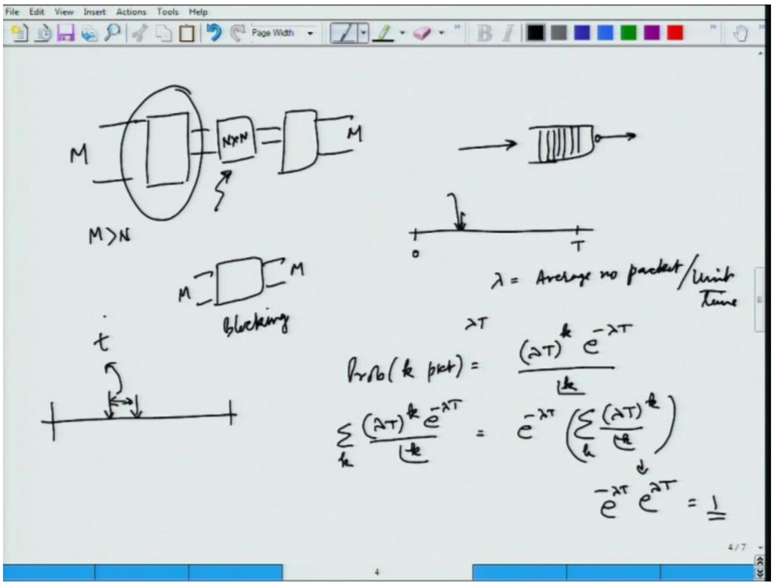
So, this will be 1, I will let me keep it as it is. So, this actually implies either N is equal to zero. So, which is the condition I am not looking for. So, I think this argument if this is 0 this is, what will give me the answer. So, this is to be 2 plus. So, 2 will be equal to minus m by m 1 square. So, m 1 will be nothing but N by two root. So, this will be the answer. So, I can now put this thing in the node to find out the number of cross points in this expression. And this will turn out to be 4 root of N by 2 into N plus 2 N square by root of N by 2.

So, this will turn out to be 2 root 2 N 3 by 2 plus again, from here 2 root 2 and this power 3 by 2. So, now, the complexity is now of the order of 3 by 2. So, we have now able to go from N square to this, this is lower complexity than, what we had in a single cross bar. So, I am able to reduce my number of cross point.

Of course, only thing is that now, I have introduced complexity in terms of the routing algorithm, which is required, because you have to search for a path and then you have to set up a path, but path can always be set up. So, this is a pretty efficient system now, before actually moving forward I need to understand, now this kind of switch requires large resources.

Now, usually network operations or whenever we do a practical design of a switch we will never build up a actually a restrictive non-blocking switch, because you do not require it. If I it can induce a small amount of blocking probability say 5 percent or 10 percent and have much less hardware, probably I will be able to provide an acceptable service to the users. So, it is always a good idea that or I can make a very small size restricted on blocking switch.

(Refer Slide Time: 16:48)



And then I actually use what we call expanded cell compressors, which basically is a special kind of a switch where large number of incoming ports are there, on one side than smaller on this side, with the assumption that, all users will not be active at any point of time. So, chances you will be blocked is very small, but because of which my hardware reduction is much drastic, which will help in deducing the call cost.

So, I will appreciate that. So, usually it is you will put a compressor like this then; it is a strictly non-blocking switch of N by n. So, you are using some capital N which is going to be larger than n. So, you will be using this kind of configuration for a switch, where this will be strictly non-blocking switch or you will itself will design a m by m blocking switch, with less number of cross points.

Because, every cross point require a management. So, in fact, cross point need not be actually physically made in this section I can build a restrictive non-blocking switch, even using time slot interchanger, which I will be covering sometime later N the lecture. And we need to estimate what is going to be the blocking probability, if I build up a 3 stage structure, because I need to have a blocking probability estimation, if I am going to design a blocking switch. Now, that requires some understanding of queuing theory.

So, I am just actually digressing from here, and moving to the what we call the probability, what we call state probability estimation, in case of a switch. But, before that let me do a very simple cube, because the this need to be understood, and I will use a similar procedure now, for what we call simple m by N. I will call this as a composite switch m by N composite switch, and will try to find out how you will estimate the blocking probability for this thing.

So, for the numbers of calls are going to be less than N, which are passing through this switch this will remain in non-blocking state. The moment you get a call, which is going to be number of calls will become equal to N after that any call comes in cannot be made through. It is then going to be in blocking state, we need to estimate that blocking probability, and for that we need to know understand the queuing theory basic fundamentals of that.

So, in this case it is very simple, the way we actually model it. Let me go to simple q and then I will come. So, what happens? q is always represented in this fashion. So, there can be tasks, which can come into the q, and they can be served. So, what we do is; how the task arrival task will be arriving randomly actually there is no deterministic thing? How you can actually model this particular task arrival process? So, what we do is we assume it to be a Poisson's distributed phenomenon.

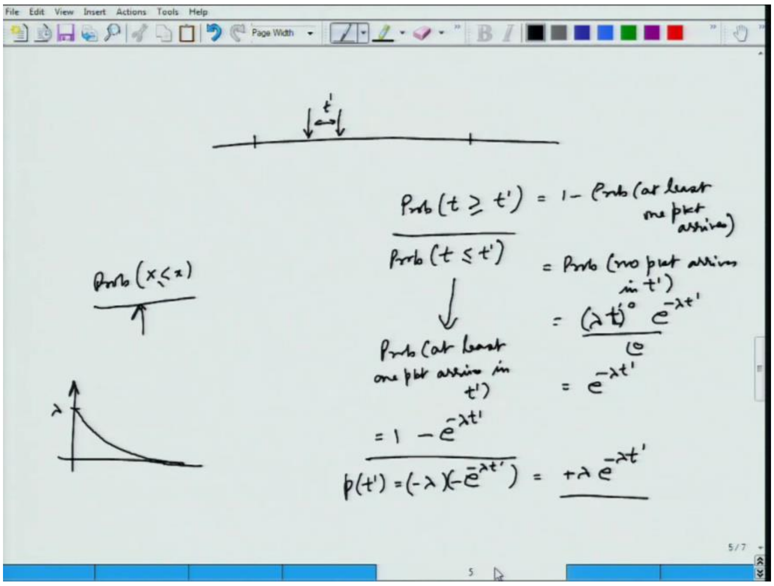
So, what do we mean by Poisson that if you take a time interval. So, you are starting from time 0 to capital time T. So, how many number of packets will be arriving in this time interval T. So, how we will what is the probability that k packets will be arriving. So, packet arrival is instantaneous process in one single instant of packet come. So, two packets can come very close to each other.

But since it is a point process two packets cannot come at the same instant, there will be some infinitesimally small gap between them. And we define a parameter called lambda here, which is the arrival rate. So, this random number of packets, which are going to arrive per unit of time. So, that is I think something, which can be done. So, in capital time T on an average lambda t packet will be coming.

Now, the probability that k packets will be there in the time t, probability for this will be defined as lambda t rise to power k, e rise power lambda t divided by k factorial this is a probability distribution. And of course, one can verify if I do summation of this, if I actually do sum of this over k. I should actually end up in getting equal to 1, which of course, is obvious because e rise power lambda t can be taken out, and this summation as all f us do know, over k this summation is nothing but e rise power lambda t. So, e rise power lambda t and lambda t will give you 1 always.

So, this actually is a probability distribution, and we call it a Poisson statistics. And of course, we can look at the particular same distribution in another way that, what is the chance? That packet arrives and the next packet arrives, what is the distribution of intra arrival time the time gap between them. If I want to find out the distribution of this then, what should be the distribution? This actually can be derived from here, we call it exponential distribution, and the derivation is again simple elementary, what I can do is; I can now take let a packet arrive.

(Refer Slide Time: 22:00)



I can find out at least packet is not arriving in this time period, I can find out this probability. So, you find out at least one packet arrive this is not the packet does not arrive at time t, this is means that probability that arrival time is greater than or equal to whatever is the t prime, which I am taking. So, remember this is a cumulative distribution function, once I take the derivative I will get the pdf of this probability that the arrival time is greater than this actually means that, 1 minus probability at least 1 packet arrives, in this time distribution or probability that no packet arrives in this particular time distribution. So, that probability I can get from that Poisson statistics.

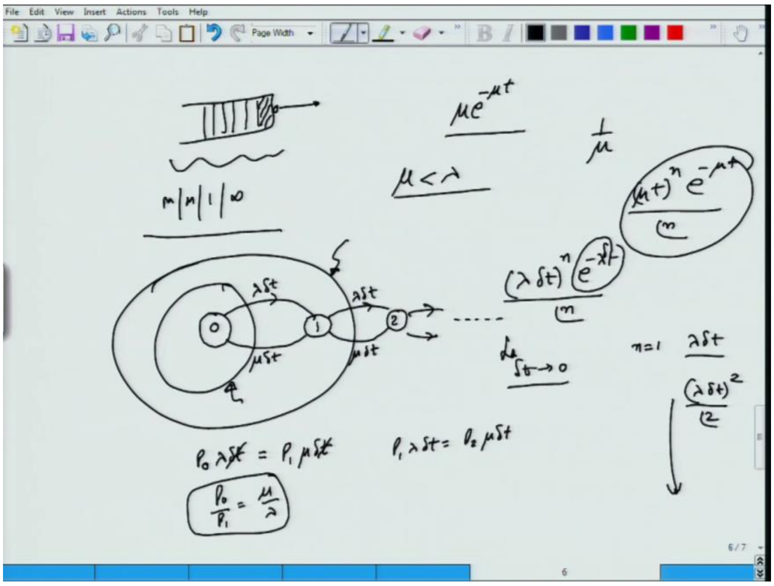
So, this will be 1 minus e rise power 1 minus. So, probability no packet arrives in time t prime that is what it means, and this will be I just used the Poisson statistics. So, lambda t prime will be 0, e rise power minus lambda t prime divided by 0 factorial, it will be e rise power minus lambda t prime. So, that is a probability or alternatively we can say that the probability that inter arrival time.

In fact, c d f is always defined that, probability of x being less than x, that is what is c d f and then we take the derivative. Here, I have taken it on the higher side the t is greater than t prime, I should actually take when t is less than t prime. So, this implies that at least 1 packet should alive arrive probability that at least one packet should arrive, I should have derived at this way at least one packet arrives in time t prime, which is nothing but minus no packet arrives.

So, it will be this thing. So, this is, what is consistent definition with the cumulative distribution function? So, this is what is the c d f, and once you take the derivative of the c d f you will get the p d f. So, once you do that take the derivative. So, p d f of t prime distribution will be e rise power minus lambda t will remain as it is, minus lambda will come out on this side, this is minus. So, this is what it will be at t is equal to 0 this will be lambda, and then it will fall down, and we call it as an exponential distribution function. So, that is the inter arrival time, which is going to be there between the two.

So, coming back to this queue. So, this arrival process is characterized by lambda, we also define an equivalent process, which is for departure which is technically nothing but again if there are always packets number of packets, which will be departing if packets are continuously there in the queue, that will also follow a Poisson statistical process. But, we cannot do it because; if there is no packet in the queue that time the packets cannot go out. So, the outgoing process cannot be represented by Poisson statistics, because that is dependent on the arrival process. So, what we do is we define a complete packet processing time.

(Refer Slide Time: 26:09)



So, if a packet is there in the queue. So, packet the moment you start serving it how much time does it take to serve the one full packet, how far it is transmission. So, that we essentially now we assume it to be exponential distributed, and we define this distribution as this. So, average value of the distribution time or the time average time, which a packet takes for moving out is 1 over mu actually in this case.

And usually what will happen is; if the packet the process if the packet if this queue is going to be empty for at least some fraction of time, if you observe for a longer duration then the queue is stable. So, only when certainly large packets will come then the queue will fill up, but all the time packets are coming and you are not serving at faster rate it means that mu is going to be smaller than lambda then there is a problem actually. If mu is smaller than lambda then queue will keep on building to infinity, it will be unstable thing we will actually prove it that it is what happens.

So, this is also technically is Poisson's process, but only if the packets are continuously available. Because, packets will not be available some once in a while we cannot use this kind of thing that number of packets going out is good is equal to this, there is going to be true, if packets are available all the time. So, technically it is also being derived from Poisson statistics.

So, what you do is, now we have to now build up what we call Marko chain for this, and we are going to assume a study state condition. So, we define the state of this queue is being defined by the number of packets, which are there in the queue. So, the first state will be the state 0, when there is no packet. So, if I take a very small time. So, now, the time of observation is very important, if you are in state if I am actually having a time interval delta t, and I will actually make this delta t go to arbitrarily 0 values.

But, ultimately this will cancel out in the balance equations. So, the probability that the packet will arrive is defined by delta t by N factorial, and if delta t limit going to 0 then of course, for value of 1 for N is equal to 1 the probability will be lambda into delta t. For higher order higher probabilities this value will, because now this argument will become 0 actually. So, this will be 1 and for higher values the probabilities will become extremely small, it will become lambda into delta t square by 2 factorial and so on.

So, these values can be neglected they are very small. So, I need not consider them. So, at any point of time when delta t is infinitesimally small, only one packet can arrive. And if that happens I will be coming to state one. So, you will never be going from 0 to 2, that is not possible in this case. So, similarly a packet goes out that probability will be mu into delta t, and you will come back to state 0. So, this happens at any point of time when you are observing. So, similarly from 1 to 2 you will get lambda into delta t, and that will be the probability, and this probability will be mu into delta t and so on.

Since, I am actually taking infinite size queue. So, this can keep on happening in. In fact, call it a markovian arrival, markovian departure, single server there is only one server, and infinite buffer size and m 1 infinity queue we call it. Now, under steady state condition the probability that you will be in certain state will be constant, which actually means a probability that I will find this system in this state 1 or state 2 are going to be constant.

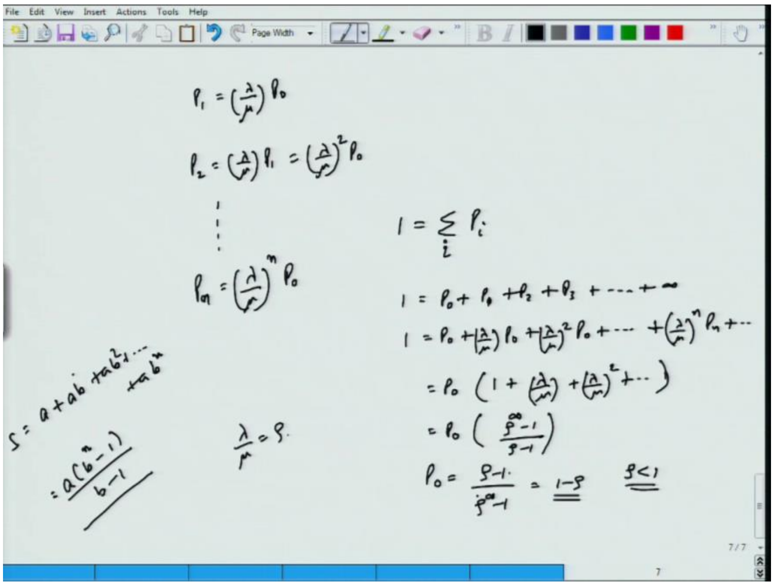
So, the probability if I make any close surface if I make any close surface the chances that you will exit out of the surface, and you will enter into the surface should be exactly same, that is the balance condition. So, this balance condition if it is satisfied will give me the steady state of actually now, I can use this particular thing to identify the state probabilities. I will exactly use the same thing later on to comp make an estimate for my m by N composite switch.

So, in this case if the p 0 is the probability steady state probability of being in state 0, an I make a surface something like this, which actually means lambda into delta t these are transitional probability. Then, I am going out of the this surface this particular surface, and what is the probability of coming into the surface that I am in probability of being in state 1 into mu into delta t. And I can see actually sorry, I have this delta t cancels out.

So, delta t is actually immaterial. So, p 0 and p 1 they are related by ratio of lambda and mu that is what is more important. Similarly, if I look at this particular thing, I can they same thing I can do it p 1 into lambda delta t is equal to p 2 into mu delta t. You can do whatever surface you can will be able to make if they are say k states possible states will be able to build up k minus 1 equation.

And for k for state probabilities you want to estimate require k equations, the last equation will be the first axiom of probability that sum of all mutually exclusive equations the probability of those has to be equal to 1. We will use that axiom as the last equation, and then we will to get all state probabilities for a steady state system. So, again delta t goes out, same condition holds true. So, I can keep on extending this.

(Refer Slide Time: 32:12)



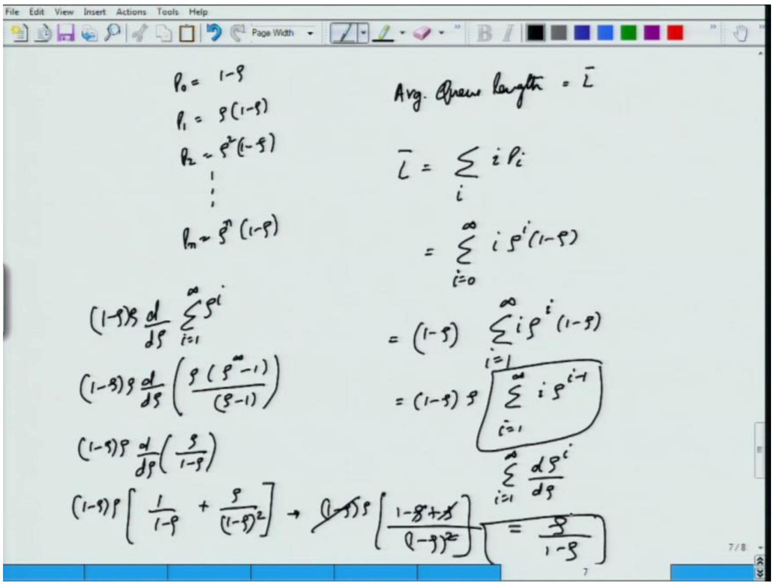
So, which actually means I can write p 1 as lambda by mu, p 0 p 2 I can write as lambda by mu, p 1 which is lambda by mu, square p 0, and tan p rise power N will be lambda by mu rise power N, p 0 and N will go till infinity actually. So, now, almost all equations I have all probabilities, I have are not been represented in terms of p 0,,but what is p zero? So, if I do p of I summation over all possible i’s they should be equal to 1, which implies that p 0 plus p 1 plus p 2 plus p 3 and so on, till infinity. They should be equal to 1 and I can write this as p 0 plus lambda by mu, p 0 plus lambda by mu square p 0 and so on, lambda by mu rise power N p N and so on.

So, this should be also equal to 1. So, this is nothing but I got a series, and this is a geometric progression. And I can very well actually solve it, I can call lambda by mu as rho. And this value will be using a series thing. So, series say that... So, once I want to solve this particular sum. So, series will be nothing but a b rise power total number of terms. So, that what should be the sum?

So, I am just going to use that same formula, and I will end up in getting is, because a in this case is 1 rho rise power infinity minus 1 rho minus 1. And interestingly what I will find out rho rise power infinity can be 0, only if rho is less than 1. So, p 0 will be rho minus 1 divide by rho rise power infinity minus 1. And if rho is less than 1 then only I will get a solution which will be 1 minus rho some positive value.

And a probability of course, has to be less than 1, which will always be guaranteed if rho is going to be less than 1, if rho is 1 p 0 will be 0. And of course, rho is 1 p 0 0, but 1 rise power infinity is also going to be undefined in that case. So, this becomes undefined. So, rho has to be less than equal to 1 for a stable situation. So, once I know p 0, I can now actually find out all state probabilities with that.

(Refer Slide Time: 35:18)



So, I have now got by p 0, which is 1 minus rho p 1 which is rho into 1 minus rho, p 2 rho square 1 minus rho and so on. And hence forth, I can find out what is average queue length of this packet. So, average queue length of course, I need not I need only the state probability thing. Because I will be using similar expression similar procedure actually now to for m by N composite switch.

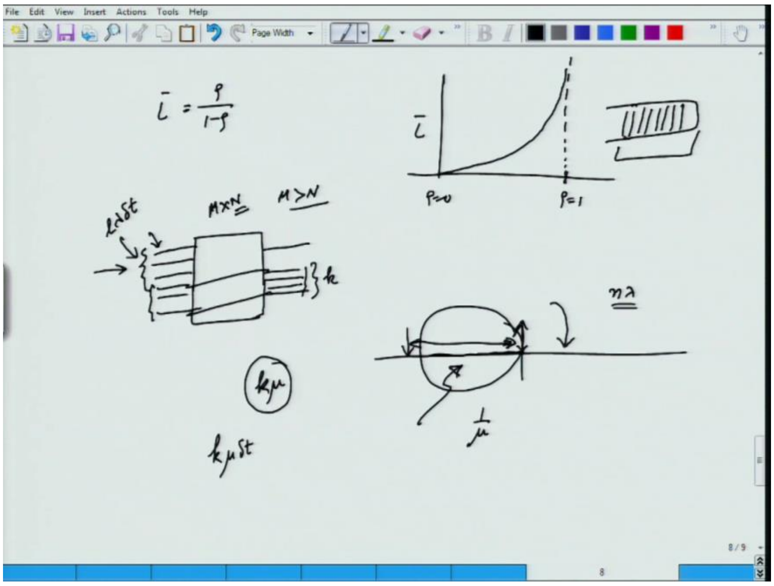
So, average queue length in this case, because this will be required. So, I call it l bar this will be nothing but summation of number of packets which were there in the queue I, and what is the probability of being in state i. So, this for all i's. So, this will be for l bar. So, this can be estimated again as, i rho rise power i 1 minus rho, and i goes from 0 to infinity.

And of course, you can solve this, and you should get a value of rho 1 minus rho, I am not solving it I am just writing down the result directly the easier way to solve it actually take the derivative take 1 minus rho out take the derivative, and then of course, replace that is the best way of doing it. So, I can I can try doing that it is not. So, let me do it here slightly. So, 1 minus rho comes out straight, i is equal to 1 2 entity for i is equal to 0 does not make sense, I have rho i 1 minus rho. I can take also a rho out and of course, now this thing can be always written as, and summation and derivation both are linear operators. So, I can always exchange them.

So, I will end up in getting d over d rho summation of d rho rise power i, i is equal to 1 power infinity. Now, this summation is again nothing but a geometric progression with the first value is going to be rho in this case. So, rho rise power infinity minus 1, rho minus 1. So, rise rho rise power infinity minus 1, this thing actually will turn out to be nothing but derivative of rho 1 minus rho.

That is what you will get remember; whenever you are taking 1 over 1 minus mu derivatively minus 1 over 1 minus rho square, and minus rho again to become positive actually. That is how it will be and I can solve it 1 minus rho plus rho this cancels with this I will end up in rho over 1 minus rho. So, that is what was the result, which I wrote here. So, this gives you the average queue length.

(Refer Slide Time: 39:46)



And you can actually now, plot rho into 1 minus rho is equal to l bar, and that is when rho is equal to 1 and rho is equal to 0, this goes from 0 and then this explodes and goes to infinity. So, queue becomes unbounded it will not be bounded in length, and it will be unstable at rho is equal to 1. So, we will use the same now technique essentially this is the basics of, I think everybody in queue would have done this thing first. Now, we need to do it for m by N composite switch now here, what is happening is; a call actually arrives the call request arrives, and those arrival will happen.

Now, you have to understand if whatever has a free port only call can arrive on those, once a call arrive that line is now busy. So, if you look at a telephone line if a call arrive call will now remain hold, and now line is now occupied till the call is over, the person is not talking for this period. And when call is over again line is free then only the new call can arrive. So, in this period no arrival is possible in the line actually.

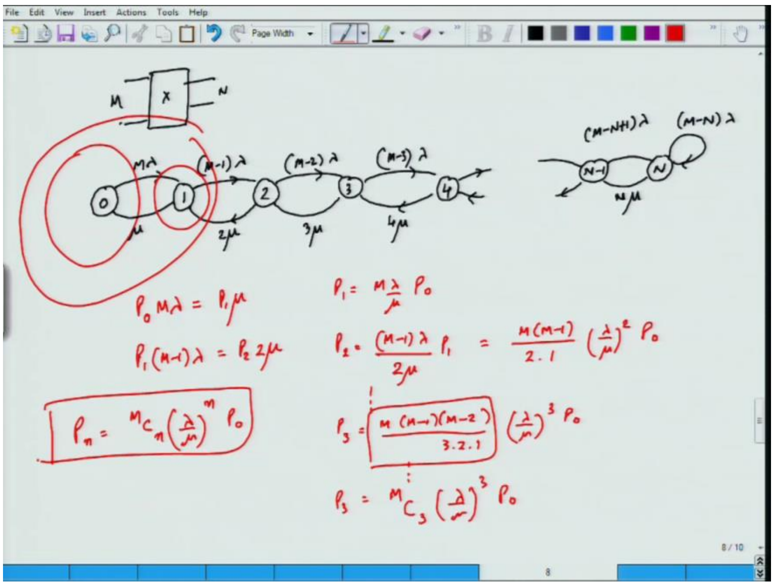
So, arrival is only possible for the free lines, so but for a free line when a line is free, what is the arrival rate. So, I can take some lambda. So, if there are N free lines my arrival rate will be N into lambda. So, lambda is arrival rate on single thing now, once somebody is talking usually what I am assuming is the person, what he talks; The duration of his talk is now exponentially distributed, which actually means this call completely takes on an average 1 over mu time.

So, I what I will call is a call departure rate is mu, for every line. So, if there are k lines, which are occupied, which are busy which are which are been talking. So, k here also going to be occupied connected to these. So, k into mu times will be the departure rate in this case. So, k into mu into delta t is the probability of coming from state k to k minus 1. Once you go to k minus 1 it is k minus 1 into mu into delta t, and similarly arrival rate is going to be whatever; is the number of free lines.

So, if number of free lines is l. So, it will be l into lambda into delta t now that will be variation here. And the state of this m by N composite switch it is a m by N cross bar technically, and only l and N connections can be put on the outgoing ports. So, the state of this particular switch is being governed by number of lines, which are occupied here, at the outgoing port you cannot have more than N lines that is not possible m, because m has been taken as larger than n.

So, it is not possible to have more than N line getting occupied, and after that if it did not even if a call arrives it has to be dropped it cannot be taken up. So, it is a blocking system and it do not have the way we had in a queue, you could have infinite packet queued up here that is not possible you can only have till N. And after that even if call comes, you cannot hold it actually it is gone. So, you are in the blocking state your queue will have only the capacity of N here. So, that is how m by N composite switch can be modeled with this particular thing. So, let me now build up a Marko chain for this entity.

(Refer Slide Time: 43:28)



So, your m by N composite switch is going to be… So, there is no call there. So, the arrival, which can happen will happen with the rate, m lambda delta t. Remember, delta t I N is going to cancel, when I am going to have a balance equations to be built for a steady state situations. So, I need not I can actually remove delta t. In fact, earlier time the delta t cancels from both sides I need not keep it so.

In fact, when I have built the Marko chain I could have removed the delta t without any problem kept, now that is what is actually is being showing in most of the places. So, I need not keep I just keep the rates here, it is proportional to rates and so on. And of course, if one call arrives you will be in state 1, and then how you will come back to 0 delta t.

So, I am now going to remove delta t because of the reasons which I have mentioned I need not keep it, then if you are in state 1. So, one call is coming in and one call is true. So, only now m minus one lines are available on which the call can come, and here N minus 1 line are available. So, if the next call comes, this will come with a rate m minus 1 into lambda, that will be arrival rate.

And from state two you can transit back to state 1 with 2, m at this instantaneous probabilities are fixed, and then only one call can be completed at any point of time. So, if 2 2 0 will happen from 2 2 1 and 1 2 0 that is the way it is always going to happen, because of delta t its limit is going to 0. So, from 2 I can come to now 3 and this will be m minus 2 into lambda 3 mu perfect I can keep on doing it, but for how long. Ultimately remember, I have maximum size N minus 1 calls can be set up.

And when the N th call is set up you have had only one line free. So, it has to be lambda, and you can go to with N mu when after that see me if arrival happens you can come back to the same state, does not matter after this. So, with this transitional probability actually loop backs, m minus N lambda arrival you come back to the same thing you do not change the state.

So, this here actually now this should not be lambda, but this should be m minus N minus 1 lambda actually. I need to correct this, and you can actually now note that the sum of these 2, if you make m minus three forget lambda and mu in this case m minus 3 plus 4 this will be m plus 1 m minus 2 plus 3 m plus 1 m minus N minus, I think it should be N plus 1, minus N plus 1 plus N m plus 1. So, that is how you can verify. So, this will be the Marko chain for a composite switch.

And of course, now I can write down the balance equations, and then solve them and then get the probabilities. So, let me do that. So, I am going to build up a first of all this surface, p 0 into m of lambda should be equal to p of 1 into mu. So, which implies p of 1 is m lambda by mu p 0. That is what you will get, if I make the second surface like this or you can make a surface like this also does not matter. Ultimately, you will be actually now cancelling all the terms, and trying to get all the state probabilities in terms of p 0.

And sum of all probabilities will be always equal to 1 that is a way it is going to evolve. So, p of 2 now in terms of p 1, I can write m minus 1 lambda, it has to be p 1, p 2 into 2 mu. So, that is the transition rate at this surface, which will give me p 2 is equal to m minus 1 lambda divided by 2 mu into p 1 m into m minus 1 2 into one lambda by mu square p 0, keep on doing it.

So, p 3 similarly will be m into m minus 1 into m minus 2, 3 into 2 into 1 lambda by mu q p 0. And of course, you can clearly observe, what is this? It is coming to real m c 3 lambda by mu 3 p 0. And in general I will have p N is equal to m c is small n, n has to be actually smaller than N, lambda by mu rise power m p 0. So, now, at this point actually let me stop. And we will now take up this thing further and try to understand how to compute call conjunction probability, and time conjunction I will also define those two terms in the next lecture.