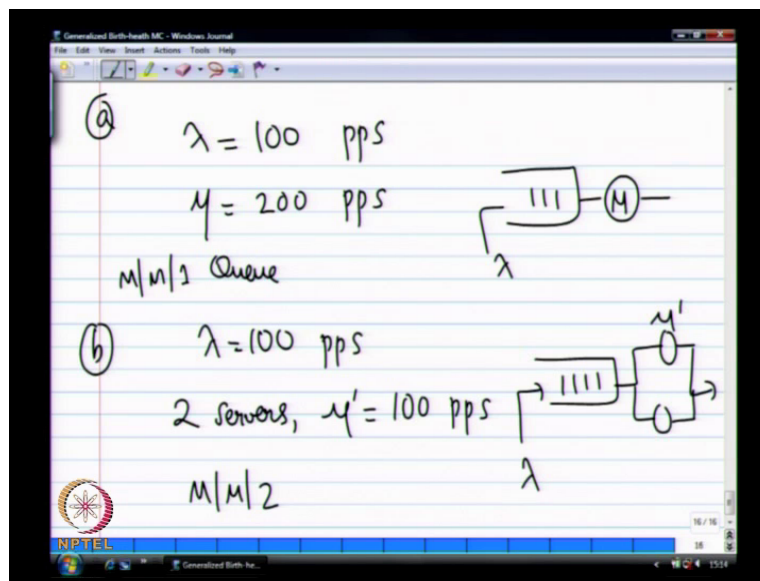


Performance Evaluation of Computer Systems
Prof. Krishna Moorthy Sivalingam
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture No. #14
Queuing theory – III

(Refer Slide Time: 00:22)



Let us do a numerical a break, that monotony of deriving. Now, there is a system. There is currently, this is your example. So, lambda is, you know, 100 packets per second. This system, that is, there is a system arrival and I have mew of 200 packets per second. This is one system. So, this is an mm 1 queue. This is one system that we are trying to; our alternate system is, I still have the same single queue set up but, with 100 packets per second as arrival.

But, now I have two servers and my mew is 100. Say, have two slower servers and one faster server or one fast server but, only one common queue. So, pictorially we have this. This is lambda. This is like mew. Well, I should probably say mew dash to differentiate these steps. So, this becomes your mm 2 system.

Now, let us compute the basic; first order performance metrics, the waiting time, delay, everything. So that, you will compute and let me know. Compute all the four. u of n u of n q a u of r n ability Say, mm 2 requires one more to extra computation. So, bring out your trusty calculators and then, tell me the answer.

(Refer Slide Time: 03:09)

The image shows a handwritten note on a digital whiteboard. At the top, it is labeled 'M/M/1'. Below this, the arrival rate is given as $\lambda = 100 \text{ pps}$ and the service rate as $\mu = 200 \text{ pps}$. The first calculation is the expected number of customers in the system, $E[n] = \frac{1}{100} = 10 \text{ ms}$. The second calculation is the expected waiting time, $E[w] = 10 \text{ ms} - \frac{1}{200} = 5 \text{ ms}$. At the bottom, there is a calculation for the expected number of customers in the queue, $E[n_q] = .005 \times 100 = 0.5$, followed by $E[n] = 1$. The whiteboard also features an NPTEL logo and a slide number '17/17'.

So, I will just write down the answer. Compute M M 1. Compute that. You want, suppose you need a calculator for this, so, E of r is 10, E of w is, do not forget your units as you go along. E of n is 10, E of w is 5 milliseconds. That is, there an E of; in that case, E of n q , that much and E of n is 1. 10 into 100 so, waiting time is 5 milliseconds. Response time is 10 milliseconds. So, fine; that was simple.

(Refer Slide Time: 04:58)

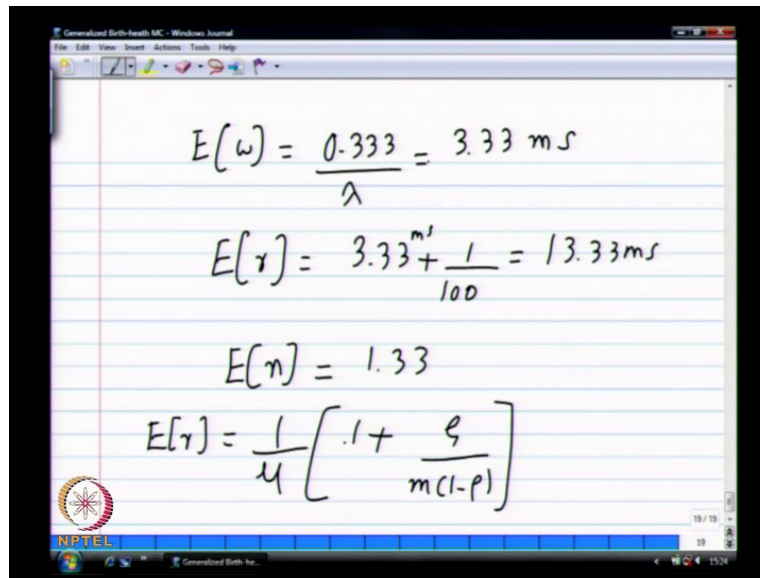
$$m|m/2 \quad \lambda = 100 \quad \mu' = 100$$
$$\rho = \frac{\lambda}{m\mu} = 0.5$$
$$m\rho = 1$$
$$p_0 = \left[1 + \frac{1}{1!} + \frac{1}{2! [1 - 0.5]} \right] = \frac{1}{3} = 0.333$$
$$q = 0.333 \quad \therefore E[n_q] = \frac{0.5 \times 0.333}{0.5} = 0.333$$

Now, for $m = 2$. So, we will compute. So, λ equals $m\mu$ **sorry** ρ will be 0.5.

Same here. So, $m\mu$, this 100, ρ is λ by $m\mu$, which is 0.5. $m\rho$ is 1. So, given you a very simple calculation, $m\rho$ equals 1. It should not be that much difficult. So, p_0 is 1 plus $m\rho$ by 1 factorial. So, 1 divided by, but, this is 1 factorial. 1 factorial is going from 1 to $m - 1$. So, it is simply $m^0 m\rho$ by 1 factorial plus $m\rho$ to m by 2 factorial. So, 1 by 2 factorial into $1 - 0.5$, which is also 1. Say how? It cannot get any better than that. So, 1 by 3 and this is simply, this last term last term into p_0 is your ρ . You do not think much.

The third term into p_0 is basically our probability of m . So, that is also 0.333. So, 0.5 into 0.333, divided by 0.5, this is also 0.333. Do not expect all q 's to have the same value. So, let us stop here for a minute. So, expected number E of n_q , let us compute E of w also and then, we will do something.

(Refer Slide Time: 07:44)



The image shows a digital whiteboard with handwritten mathematical formulas. The formulas are:

$$E(w) = \frac{0.333}{\lambda} = 3.33 \text{ ms}$$
$$E(r) = 3.33 + \frac{1}{100} = 13.33 \text{ ms}$$
$$E(n) = 1.33$$
$$E(r) = \frac{1}{\mu} \left[.1 + \frac{\rho}{m(1-\rho)} \right]$$

The whiteboard also features an NPTEL logo in the bottom left corner and a Windows taskbar at the bottom.

So, basically 1000 by 10, so 3.33. First one is better. So, E of w, yeah so, waiting time is 3.33 milliseconds here. How much is in other side? It was 5 milliseconds. So therefore, definitely waiting time is less here, which is again intuitively ok because, would expect that the two servers being there, at least I will be able to get one server faster. But, if you look at the total response time, that is plus 1 over. But, the dominant factor is 1 over mew which say 10 milliseconds in this case.

Previously, it was 5 milliseconds. Now, it is 10 milliseconds. It is only 100 packets per second. I will end up getting 10 milliseconds. So, E of r is 13.33. So, clearly that one is better. That only got 10 milliseconds and then, E of n is 1.33. So, so what this says is, having one faster server is better than having two slower servers even though their combined capacity is the same.

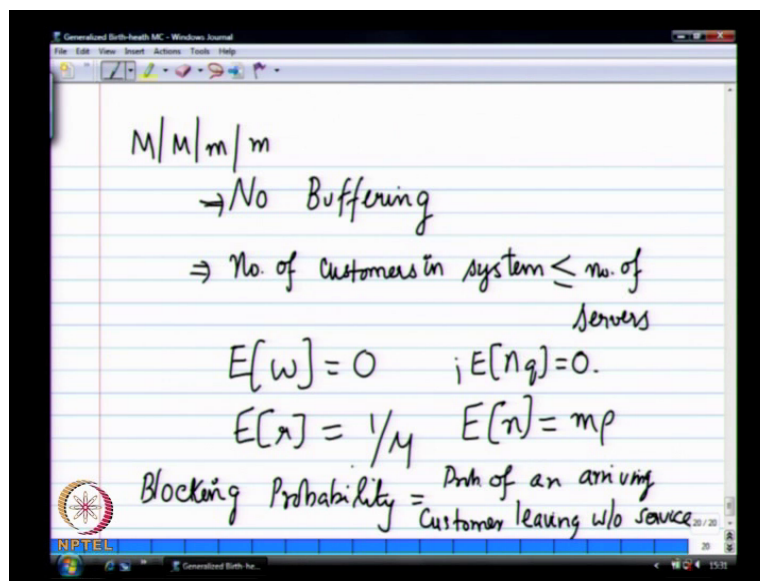
That is all and if you want to look at that closed form expression for E of r, then this will become evident. So, what you see is that, I have not derived this. This is in the book itself. What you what is, 1 over mew is the multiplication factor. So, for one system, 1 over mew is basically, even though m equals 1, mew was 200.

Other case, m equals 2 but, still your mew is 100. So this, unless, so this is 1 plus some other smaller factor. So let, how do we rationalize this? So, mew is definitely larger in the case of the per system and unless this multiple tends to be larger in the case of mm 1, this will not be

the case. So, you have to somehow prove that this rho by m, at this p q by m is going to be less in the case of mm 1. That we will have to sit down and see if you can work it out.

For this example at least, the intuition is that, faster the service time, your m 1 spent by the customer getting serviced is what everything is proportional to your total response time. So, even though wait, so when you compare to systems again, that thing, that you derived home is that, simply do not compare the waiting time alone. The total response time is what will make a difference. So, we just compare one metric and then say I am done with that. If I ask a question, compare E of w and say which is better, which system is better, then you should not follow that. That is not a good metric. You should say that, that is not adequate. We should also look at E of r and then say which one. So, that is one small example for mm 2. Any questions before I move to the next derivation, next two derivations. We can. How fast we are. Oh yes. Is it? Oh, is p of E q into is a number of rho, but 1 minus 4 is the number of customers in mm 1 system. Using that, here also but, E of n is not rho or 1 minus rho. We will, I will talk to you and see the derivation.

(Refer Slide Time: 13:02)



So, very quickly how can we generalize these two? A system where there is no queuing. So, this the M M m m. So, there is no buffering. So, the maximum number of customers, so, the number of customers in the system is less than or equal to the number of servers. So, there are m, so, m is the last parameter. By the way, is the number of maximum number of customers in the system. Population size, we went in and figure out. So, population size is the actual

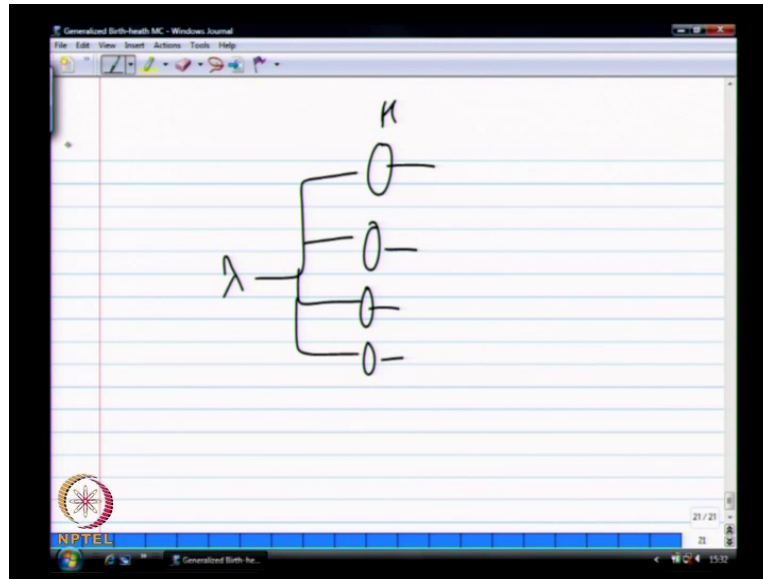
total possibility. So, let say, this, the computer center example. What they are giving there? Computer center has 100 seats but, the student population is 10,000. So, you have this set up.

At given point, only 100 customers can be in the system but, the total population of customers possible is 10,000. That is what the population. If you want to use that as the, so, this is system with no queuing. So, in this kind of system, what will be the waiting time? Waiting time will be 0. Because, you come in the system, either there is the server free or simply leave the system.

So therefore, E of w is going to be 0 and E of r , 1 over $m\mu$, this is going to be the very first derivation. E of n is λ by ρ or λ by $m\rho$ and E of n_q equals 0. So, what is it we derive? What metric are we missing? The probability of drop. All dropping that is probability of a customer arriving to the system and then not getting serviced. It is like you do not want your customers to leave your system. If you go to a mobile shop to buy a cell phone, there are four counters for servicing people. Every time you go, they are all busy serving some other customer. You do the service and you leave the system and go to some other shop to buy your cell phone. Therefore, it is a big loss for this shop to have customers leave. So, there is a penalty associated with call dropping. So, we therefore, derive this so called blocking probability. So, that is the only metric that is of importance. Nothing else in the system. Everything depends only upon λ and $m\mu$. So, blocking probability is what we want to define. So, this is the probability of an arriving.

It will be dependent upon. You have more servers and there will be less dropping. λ is fixed. Assume that λ is fixed and $m\mu$ is fixed. m is what we have to control over. So, this service; no, this is not population. m is the number of service. I just whet back to your old question about population size. Here, m is the number of servers in the system and the final m is the number of customers also in the system. In the system, this $m-1$ is like, b is a buffer size. There is no queuing in the system.

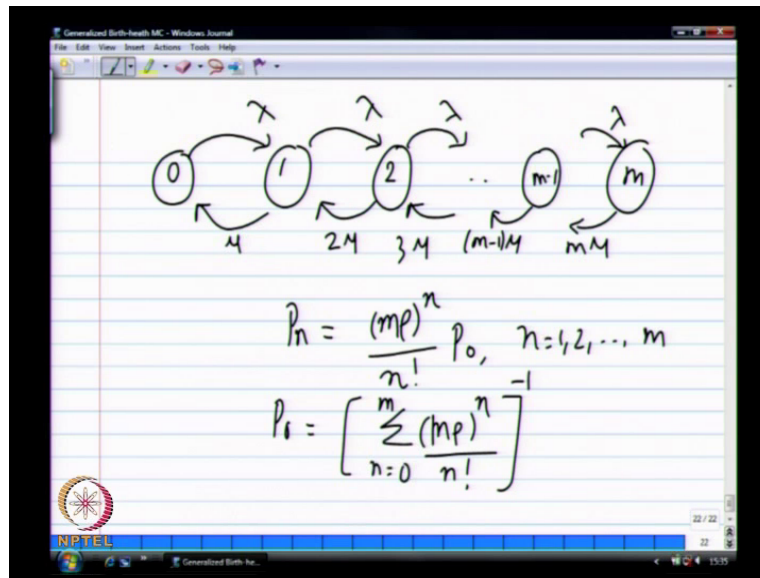
(Refer Slide Time: 17:28)



So, if I want to represent that system, it is a series of servers. So, customers arrive and they simply go to one of these guys and then leave. This is λ and this is μ . But, there is no way for them to leave the system. It is like making a cell phone call, right. You will need one channel or a pair of channels to make uplink downlink calls and when you arrive, all channels are busy. You cannot be given a channel.

We saw this in Tedium also. Because, in a way of 24 slots in a tedium frame, then all slots are used, you cannot be given any other slot. So, you want to minimize. So, the goal is to derive what is the best m . So, this is very important for its telephone networks. All this, they came about when designing telephone networks. What should be number of this, way back in the 1920s, 1900s and so on, where you need to decide some good value for m . How many circuits should I have in a system, such that, the customers do not get annoyed. So, how do we derive that?

(Refer Slide Time: 18:19)



So again, if you go back to our birth death process, it is very simple. Either 0 customer, 1 customer, 2 customers and I do not go to infinity. I simply stop with m customers in the system and everything else is the same and so on. So, only finite some number of states. So, what is the p of n equal to? We derived this before. The same p of n that we saw for m equals to or n equals to m minus 1. So what was that? It is simply m root to the n divided by n factorial.

In the last experiment that we saw, we had m plus 1; m plus 2 are slightly different. But, m and m plus 1 because, if we look at m, there are m **sorry** 4. So, for m, it is 1 to m for m minus 1 and so on. Therefore, my p naught is simply sigma n equals 0 to m. Is rho have to be less than 1 or anything like that? Does not matter. Lambda can be as large as you want. Does not matter because, beyond m you are not going to grow. System size will not grow infinitely.

So, it is a stable system. Whatever is the value of lambda because, if you look at this summation, there is, if you can simply, it is just a summation. So, we can plug in whatever the value of rho that you want and you need get some answer at the end. You are looking at as a simple summation. So, this is the expression for p naught. So, once I get this, what is the probability of a customer getting dropped or call getting dropped? Simply, p m. When the system is in state m, there will be no service for incoming customers because all servers were busy. Therefore, the connection, incoming call will get dropped. Therefore, blocking probability is x. So, blocking will take place only when the system is in state m.

(Refer Slide Time: 21:39)

The image shows a digital notepad with the following handwritten text and equations:

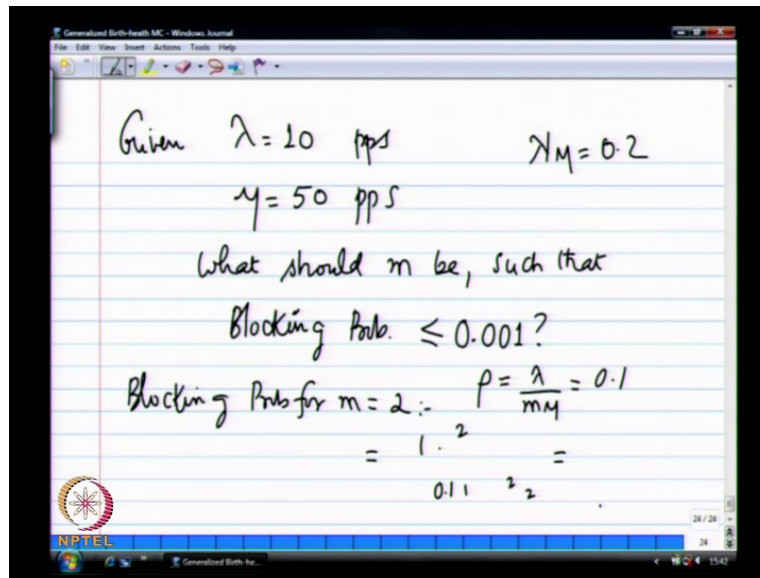
$$\begin{aligned} \therefore \text{Blocking Prob} &= \text{Prob system is in state } m \\ &= P_m \\ &= \frac{(m\rho)^m}{m!} \\ &= \frac{\sum_{n=0}^m \frac{(m\rho)^n}{n!}}{\sum_{n=0}^m \frac{(m\rho)^n}{n!}} \end{aligned}$$

Erlang's B formula

The notepad also features an NPTEL logo in the bottom left corner and a status bar at the bottom with the text "Generalized Birth In..." and a time of 15:37.

So, when I arrive to system with m customers, it cannot be handled. I have to leave the system. So, that is simply p_m and that is nothing but, and this is also attributed to Erlang. What this formula called as? It is Erlang's e B? Which one is this? e folks will know which one is Erlang B. So, usually the question is the other way. So, given λ , given μ , what should be the m so that, my target probability is less than, say 0.001. Usually keep that very low like this. 0.99999 percent should be the acceptance rate. So, 10^{-7} , for example, is a call. Around one in a one in a crore customer calls can get dropped. That is what you have to design for. You have that many customers, that many servers represent the system. That is what you use this formula for. Questions? So this, we want to calculate, so here, I do not, let us try this.

(Refer Slide Time: 23:27)



So, given lambda equals, you know, 10 packets per second and mu equals 50 packets per second. Only one in 1000 calls will be lost. So, you can do a closed form solution or we have to be iterative. We have to be iterative or even binary search will do. You need some random large m and then keep reducing it or iterative. Start from m equals to 10 and so on and see where it finally hits.

That is close to that, so that, we can go more closer. Just start with m equals 2.

Blocking probability, that is way of and then we take so many iterations to get the number of iterations.

The number of iterations will be equal to the number of servers which you are looking for. That will be the worst case. Try to reduce that. Just the linear search will solve.

It is better than.

Close this thing and then we can.

But, you need an initial guess and then move in the one of the, one reduction of the other. But, this case, where, where what will be the initial guess? Let say m equals 2. What if m equals 2? What is the blocking probability where m equals 2? That you can find out. So what if m equals 2? So, may be will get lucky here. So, what is m equals 2. So, we know that

λ by μ equals 0.2. So, what is this going to be? This is going to be 0.2 square by 2, no, m rho, so 0.4. Say, m rho is 0.4 or λ by μ , sorry, no, 0.1, will be 0.1.

Let us abandon this for now. Say, let us just do rho equals λ by m , so 0.1. Therefore, blocking probability is, so there is enough to calculate and what is that 0.36? Very high. Let us, m row is 0.1. Actually, m root does not matter. We will abandon this all over again and go to the next page.

Actually, m rho will never change because, that is going to be fixed because λ by, no, yeah you can.

Get a good initial guess.

M rho will be always λ by sorry.

We can actually put the denominator which is going from n equal to 0 to m as from n equal to 0 to infinity then, it will be become like E power m bar. There you can get a guess for m and then start from that you will **will** reach first.

You get E power m rho. But then your numerator could be some **some** part. You will have your, denominator is going to be E of m rho something to the m , m rho to the m by m factorial into E of m rho equals this one. Then, you can solve for m with that as an upper limit and then try to lower limit and then start from binomial.

That is right. As m rho increases, m rho will be, rho will decrease and m rho will stay constant. So, is 0.36 correct for m equals 2? So, when you come back on Friday, let me know what they answer for m .

It is 0.004.

Yes, that is, that is the other alternative. Just as you said, approximate that. Denominate that E of m rho and then, try to solve that to get a starting point. You do not have this but, if you have a computer, you just a like

How much like, m equal to 5 equal to 0.0048.

0.0048? So, at n equals to 6, then you should or 6 or 8 should be able to. So, usually you try to estimate this. You never really you know what is the λ in the real system. Use that as

an approximate method and then, as your, find the blocking probability is increasing, simply add additional servers. If you look at today systems, with the packet switch network, we always add additional servers as delay goes up. In this case, simply look at the, because we never know when a customer comes and does not get serviced in the system.