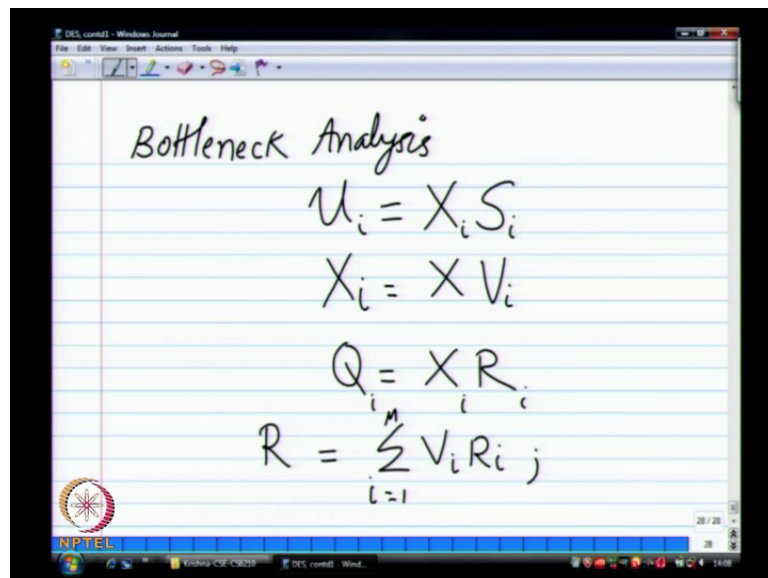


Performance evaluation of computer system
Prof. Krishna Moorthy Sivalingam
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture No. # 25
Operational Laws - II

So, what we saw last class was the set of equations mostly simple of simple once right, that are used to calculate the response time and throughput and so on.

(Refer Slide Time: 00:37)



Bottleneck Analysis

$$U_i = X_i S_i$$
$$X_i = X V_i$$
$$Q = X R_i$$
$$R = \sum_{i=1}^m V_i R_i$$

So, which of those you remember, we need that for a test, some four laws; four or five that we saw right. So, the first one is what utilization of a device of the system itself is, right, say u equals say $X S$ right. So, X is the throughput of the device, S is the service time of the device, right. This is basically our ρ equals λ by μ ; just another form right. Then Little's law right sorry, second one is not Little's law. It is your force flow, so force flow is X_i equals X into V_i , yeah.

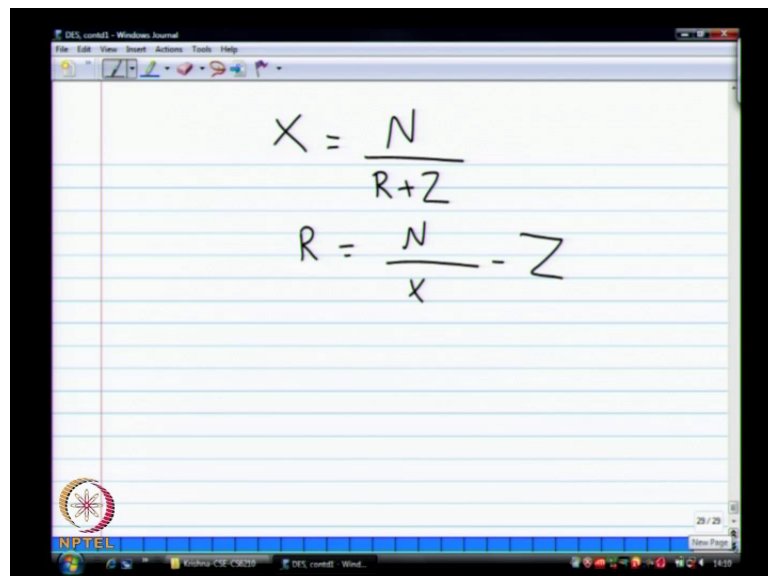
Next one also Little's law, right, so Little's law equates, say everything starts to, right, first thing you find out the X , then you can slowly start building it, right. It will be specific here, right, so you can put all the u s if you want, this is utilization. And X is

usually given in the case of an open loop system. If the balance, the flow balanced assumption is satisfied, then X is equal to λ right, then the single...

And then the next one is our general response time law. So, that is, right, so given that, you know the number of resist every device and the response time at each queue or each device, overall response time for a particular job process is just this one right. And then the case of this interactive system that we saw this specialized case of a time sharing system; so every job right, you basically have a set of circulating jobs.

The reason for looking at this interactive and general response time is, in the olden days there were only batch jobs. So, there was a batch job submitted and it went through all the queues, then it clubbed the system. But as soon as one batch job is finished, system immediately loads another batch job as long as there is one ready in the queue. It always assumes that there is always a set of batch job to be processed. So, that is your... when there is no think time. And the other option is when you had time sharing terminals, where there is always this delay between the system giving you the response and you send somebody in a new job. Essentially, it is treated as yet another job that is being recirculated.

(Refer Slide Time: 03:44)


$$X = \frac{N}{R+Z}$$
$$R = \frac{N}{X} - Z$$

So, in that case we have this. Remember, we derive this thing right; X equals N by R by Z . I will go to the next page. If there are N jobs and each jobs takes R times for

completion, Z times for thinking. Therefore, the throughput of the system is, right, this usually represented in terms of N , if it is close to the system.

And therefore, your R is also determined by this. So, this is, right, so basically everything seems to start with X . Once you figure out the V_i , V_i is almost always given or and some certain form and this thing. So it is, so that is basic start, but then we are getting little more complex stuff as we go along.

So, sometimes what we would like to see; we will mostly restrain ourselves to this time sharing system because that is generalized version. In other system, in non-time sharing system, your Z goes to zero. That is all; with the pure batch system, then Z is simply zero. There is no delay between one job, Z could be also, say new job loading time or something in olden days you have to get a new cards put into the system.

Now, maybe there is some time to setup to this job from. Even now you know computers send it today, we have cluster based computing. So, there is a cluster, the tax of jobs in a batch basis. And then you submit it, you wait for your turn whenever at some point in time they load it, they run, you get back the results. So, you have that or you just have to, right, so there is a delay there before your job actually gets run in the system. So, in that case, simply goes to zero if it is purely batch base system.

So what we try to do is, occasionally there is a hurry that you want to know. You just tell me the bounds, right; some upper bound, lower bound, where throughput of the system or the or the response time for a system. Given the set of your network queues, just want to know what is the upper or lower bound, something that you can figure out for both these things. And so we will start up with the simpler bound, then we will revise this, you know more complex bound later on. So, let us... Simpler bottleneck analysis.

(Refer Slide Time: 06:00)

Bottleneck Analysis / Bounds on R & X

Let Device b be the bottleneck device.

$$i.e. D_b (= D_{max}) = \max_{i=1}^M D_i$$
$$X(N) \leq \min \left\{ \frac{1}{D_{max}}, \frac{N}{D + Z} \right\}$$

where $D = \sum_{i=1}^M D_i$

$$R(N) \geq \max(D, N D_{max} - Z)$$

So, this is called as bottleneck analysis, but what we are actually trying to do is get bounds on R and N, no, R and X. So, we have a set of devices and every job requires D_i ; that is demand. That is made on every device as it circulates through the system. Let say average time.

So, we will find that there is always one device; it is a bottleneck device, Right. Let say there is one device, which has the highest value of d_i across all the devices that is your so-called bottleneck device. And the way it is the system designed is you find out the bottleneck device and you find ways to essentially balance that or reduce the bottleneck, where either making the device that run faster or some other way. So that, the loads, a rest, the demand is more or less balanced; which will get the best throughput for a balanced system. So, if it is an unbalanced system, let us see what happens.

So, let me write device b i can erase this. Device b be the bottleneck device, so that is, there is this D_b which is also, we call that D_{max} from now on. So, D_b is the device, the highest demand of all in my notation, the this oh these. So, yes let me put D there. Sorry, yes, next I do not need the arg there; D_b simply, so these by... So then the throughput bounds right, simpler bound.

So, the throughput will be at least these two. This is one bound. So, this is an upper bound, lower bound; upper bound. But sometimes you also want to lower bound. That comes later. And for the response time, so why I state in terms of N is, this is a closed

queuing network. That is the assumption here. There are N jobs circulating in the system. right.

So, for N jobs circulating in the system, what is the rate of completion? Again think of this as your multiprogramming system. If you feed the operating system with ten jobs, then on average how many jobs this system finished. So, that is the rate which you can actually finish and add new jobs. That is your throughput. And I do not do anything that... and the bounds for R of N . Even in this case, we have lower bounds; D_{\max} minus z . Again, for these two we will get that bound later on. These are the first set of bound derivation. This is the average response time when there are N customers in the system. Yes, this is the average response time for job. Everything is in terms of the number; customers and jobs, right, X of N is number of jobs completed per unit time; R of N is the response time for job. So, you have this job. Your C P U, you process that, runs on a C P U on the disk and all that, finally it finishes. So, when does it finally finish?

(Refer Slide Time: 11:09)

Proof

(a) $u_b \leq 1$
 $X(N) D_{\max} \leq 1$
 $X(N) \leq \frac{1}{D_{\max}} \quad \text{--- (1)}$

(b) Let $N=1$
 $R(1) = \sum_{i=1}^M D_i = D \quad \text{--- (2)}$
 $R(N) \geq R(1) \geq D$

So, proof, so how do you prove the first part? That the throughput is bounded by one over D_{\max} ; one of the bounds, so that. So, let us look at this utilization for this bottleneck device. So, u_b is the utilization for the device. I am operating this system, essentially with infinite queues. These are all... There is no capacity restriction in any of the queues.

So, what should be maximum utilization of a device? You can go up to one. Utilization cannot be more than one of any device. But with the whether infinite queues, infinite queues will restrict lambda, but in generally, utilization for a device is limited to one.

Now, this is the u. There is a u. there look like mu, now is u. so utilization of bottleneck device is less than or equal to one. And so is all the device. Because what happens when your system is fully loaded right, which device will have utilization closes to one? The bottleneck device; others will be less lower than that. Utilization for those will be less. So, we will take this as our starting point. So, what is u b then? Some other formula that we know, right; $X \text{ into } D \text{ b, right, } X \text{ into } D \text{ max.}$

So, $X \text{ of } N \text{ into } D \text{ max;}$ that we did not write in our last expression when you wrote u equals $X S$, there is also u equals... So, this we derived right from in the last class. So, utilization is can be expressed in terms of the overall system throughput and that demand on the particular device. So, this is one... I have this is one of your bounds. So, this will be achieved when N is large when N is large, utilization for this device will go close towards one. But for n is small, it will be little bit less than this. So, let see what that is. So, this is our first proof. Step one: first in general see we have right u equals $X t i.$ u i equals $X t i.$

Suppose, we want necessary you are got some demand for derive the very slow one demand is demand is more by $((\cdot)).$

Demand is a product of $V i \text{ into } S i.$ Number of visits to the device to into the service time. This service time is very low. So, that is... even, if it is only visiting it once, but the service time is very large. That is still your bottleneck right.

So, the service time and your number of visits to the device dictates the... So, this is, we will say part a. right. The proof, now let us do some other part of the proof. So now, in the system I am only allowing one job. right.

So, let there be only one job on the system. If there is only one job in the system, then what is the response time? Simply the time that it spends in each of the queues and you visit the queues multiple times. And therefore right, if there is only one job in our m m one system, what is the service time? Simply, one over mu, in this case, I will repeat, visit this queues several times. So, this is simply $\sigma D i,$ that is all. There is no

queuing delay, only response time, only the service time, right; service time into the number of visits to each device. This is again, these are averages, but this is your... right.

So, R of one equals D. This is your other, no, we said R of one, right, R of N is greater than equal to max of D, something else. So, this is a lightly loaded system. Heavily loaded system used for this derivation. This is a lightly loaded system for this derivation; R of one equals D. So, if I add more customers, and certainly R of N is going to be greater than or equal to R of one. So, R of one, R of N, will defiantly be this. Therefore, it is one.

(Refer Slide Time: 16:46)

The image shows a handwritten derivation on a lined paper background. The equations are as follows:

$$\textcircled{3} \quad R(N) = \frac{N}{X(N)} - Z$$

$$\geq ND_{\max} - Z \quad \text{---}\textcircled{3}$$

$$\therefore R(N) \geq \max(D, ND_{\max} - Z)$$

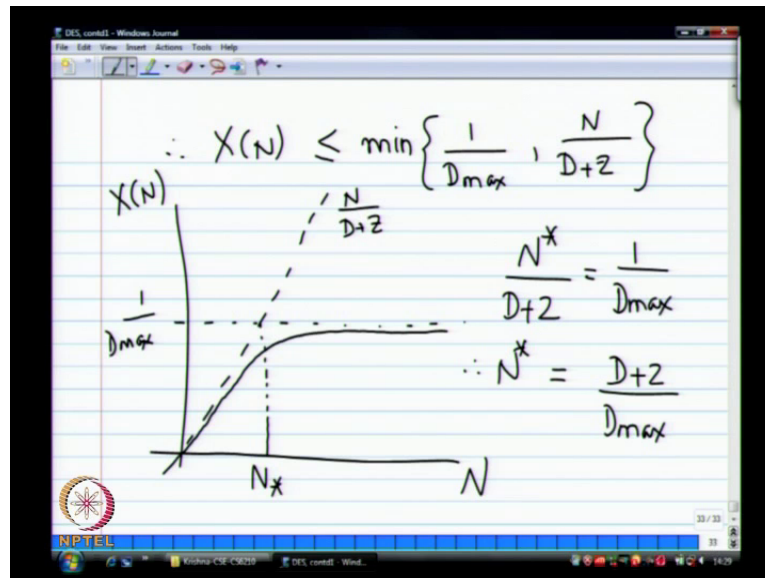
$$\textcircled{4} \quad X(N) = \frac{N}{R(N) + Z} \leq \frac{N}{D + Z} \quad \text{---}\textcircled{4}$$

Using $R(N) \geq D$

So, now I can use my next expression R of N equals... This is something we derive, right; R equals N by X minus Z. And because you know that X of N is bounded by one over D ma, this is simply, say upper bounded that is.

So, under heavily loaded conditions, your throughput is bounded by, upper bounded by one over D max. right. So, applying that, we get this other bound; R of N is greater than or equal to... So, combining these two, depending on whichever is larger, right, so therefore always... and we use the other result. Now, X of N equals N by R of N plus Z. right. This is what we derived earlier. And we know that the lower bound for that is D. right. So, R of N's lower bound is D. The least possible value is D. So, N by... So, any value for R of N will be always larger than D. So therefore, this whole thing has to be bounded by... So, therefore my bounds are...

(Refer Slide Time: 19:05)



So, these are two bounds. This one over D_{max} bound will be, I will use this when the system is heavily loaded; the other bound system is lightly loaded, that is what we will have; so that, simply N over D plus Z .

So, with this, you can try to find out approximately. So, basically you want to operate the system. So, what is the need for a multiprogramming to begin with, if I have only one job in the system when one device is being used, other devices are simply sitting empty. So, you want to add more jobs to the system such that, CPU is not idle that some disk is always sitting, used and so on.

You want to basically push all the systems to closed max utilization. But you want to find this magical point here, beyond which your system starts thrashing. Your throughput will increase up to some point. And then beyond that point if you increase, what is going to happen? You start having queuing delay. So, you want to operate the system just below that notch or the point where your queuing delay is do not start happening. If there is queuing delay what happens? Your, then your system delays starts keeping on increasing, then there is no point in adding any board jobs to your multiprogramming system.

So, let us so let us... they may plot this right; N versus X of N . So, then we have these two bounds. So, one bound this... This is the linear with N right, N by D b. So, it will have one straight line like this; which is your N by D plus Z . right. So, that linearly increases. So, initially throughput will increase with N linearly. But beyond it is going to

stabilize. And that stability; that stable point is D_{max} . So, if I simply, if I actually plot the real throughputs it will be something like this; right, keep increasing and then at some point it will start maximum beyond a point. I just cannot have any more jobs in the system because the device is saturated. One device is fully used, it simply going to queue up and throughput will not go any further. So, this is your classic new point. This is approximately, where queuing starts, upon till this point, there will be no queuing in the system.

So, you try to find out what, show what is in act right. Appropriate number of jobs I should be having in the system to avoid any queuing at all. That is what this analysis used for, so how do you find this N^* ? Just equate these two guys and you will get your N^* right. In that particular system, there is a single queue. Let say you are analyzing, queuing is there, you want to only achieve stability in terms of the number of...

Only we are saying, so we know at three point is there is so many what is this is a value of n , so we can take end value below than this then do not do anything; once it is reach this particular number of...

I know let me do our, see in the queuing network, projector that is coming up usually start with the number of jobs in the system, when we just say we start with the N jobs... right.

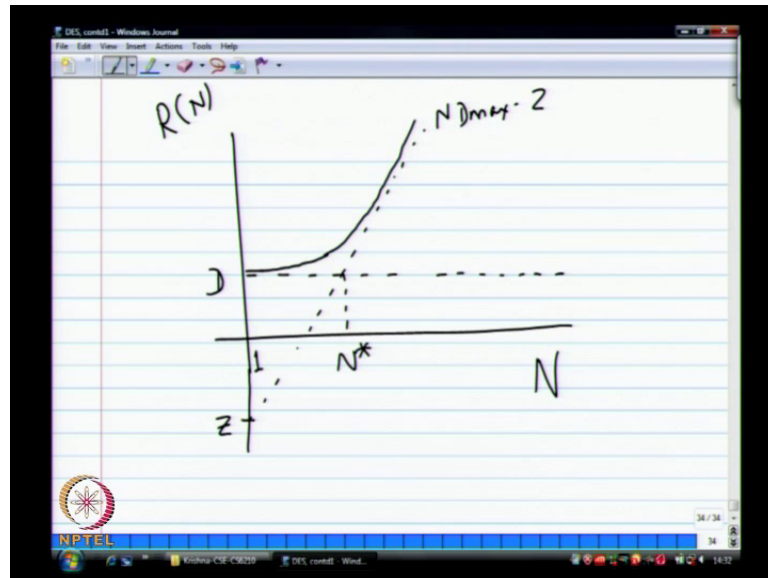
But I do not I do not think this is going to be useful for your ideas for general system or initially there are deletions for that transit. Remember, I do not, I do not see the connection between that two. We will take that after that.

So, this N^* is basically where, N^* is then simply... So, that is your throughput graph will look like as we increase the number of jobs circulating the system. This is not only CPU, this could be any manufacturing job system also. Where, you have no multiple machines that you are making a new, whatever component and that has to go through these multiple machines. And each machine you have to spend certain amount of time whether it is late or we are getting short or cut or wilder and so on.

You keep going around and then you want to see right. How do I keep my utilization of all the machines in the factory maximum, so that they are not sitting idle? At the same

time, you do not have a huge buildup of right components to be made sitting at each of the places. So, that is a basic idea right on this. Questions on this, there is no question mean, this is simpler. Then we will go faster, try to go faster.

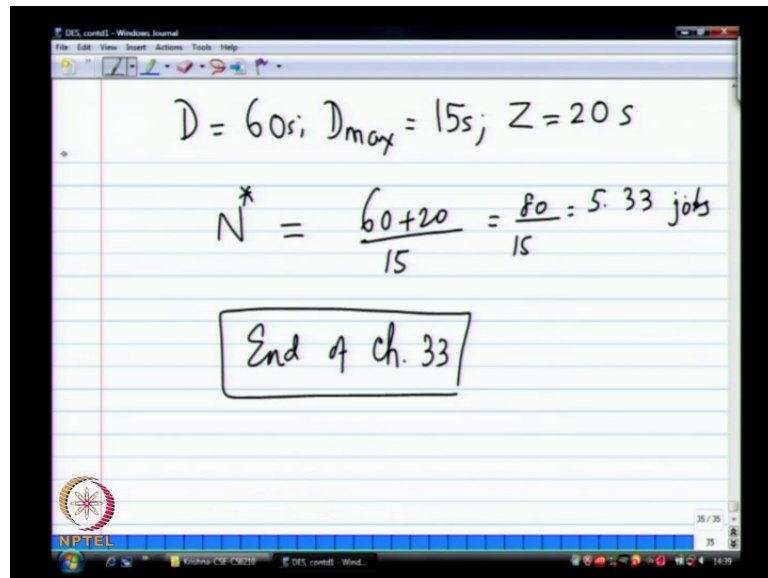
(Refer Slide Time: 23:58)



So, this is N and this is R of N . So, what are the two R of N values? So, R of N , I only have lower bounds. So, first is D . So, this goes from one to N right. So, under the light loads, ideally it should be D that is my D . For that should probably be a dotted line. Hold it like this dotted. So, that is one lower bound and the other lower bound increases with N right. So that is, so this is N into D max minus Z . So, somewhere here, right, it is here Z . So, this will be something like this. So, the slope of the line is simply D max and your y intersect is minus Z .

So now, how will your response curve look like? It is very close to D and then it slowly starts increasing. At some point, it simply starts. And it has to be greater than that. How much greater we do not know, but definitely greater than that. It should be much greater than this. So, this is your expected response time to the system. And you want to operate close to D . right. Why should I be sitting queuing in the queue? And so you want again this N star. You saw the optimal operation region for a particular system.

(Refer Slide Time: 26:10)



The image shows a digital notepad with a white background and blue horizontal lines. At the top, the text reads: $D = 60s; D_{max} = 15s; Z = 20s$. Below this, the calculation for N^* is shown: $N^* = \frac{60+20}{15} = \frac{80}{15} = 5.33 \text{ jobs}$. The result is rounded down to 5. At the bottom, the text "End of ch. 33" is enclosed in a hand-drawn rectangular box. The notepad interface includes a toolbar at the top with various drawing tools and a taskbar at the bottom with the NPTEL logo and system icons.

And again... If you want again to equate this and you get the same expression. So, if I say that, now D equals say sixty, D_{max} equals fifteen and yes Z is say twenty. All this is in seconds.

Sir, yeah...

These two graphs, steady state throughput. Yes. Throughput will stabilize because you can have any more, yeah. But if you look at the response time, it is actually, that will increase because that is queuing, yeah.

So, when your, when your queue starts building up, there is no benefit to you at all because number of jobs completing is still going to be fixed. I just, all the jobs are sitting in the queue. That is all, if you put right, you put more than the number, this N^* jobs, then your queues are all pretty much operating at close to their or other queues; might be less than hundred percent utilization. But your bottleneck will be hundred percent utilization. There is fully utilization; all other just sitting around, right, doing nothing or then I have given jobs, you are not there may would not be ideal for larger fraction of time.

So therefore, your response time means that you are right, you are now, what happens if you have hundred jobs? You will be sitting in front of 35 jobs right, by the time you get to the particular server. Therefore, wait time increases in the queues.

So, that is, that is why queuing delay right. And much later, if we have time we will actually look at how we can use this for analyzing our T C P, right. T C P congestion control, the window based congestion control is basically the same system, I am having a packet that goes through a sequence of queues, goes to the receiver, I get back my acknowledgement. Acknowledgement comes back by window moves forward. And that is essentially a closed queuing network there.

So, I want to find versions after value of N , given the delay in the system, given propagation delays and given... Well, we actually ignore packet drop. You can just right, what is what is after window size should have? That is where given T C P right, will be using something similar to this. So far now this system, what is N star?

Sir, N star basically bottleneck device can take only...

If because of **yeah**; it is a bottleneck device known your N star customers will divide across the multiple queues, but there are all sitting at the bottleneck queue. Will be other, they will be visiting other queues too, that is idea of the... Is all the time get and N star always, so that we can have be fixed amount of queuing delay as well as the...

Yes, example of real life system, if you are looking at this machine or this batch system, you approximately find out from analysis, right, what should be the N star. Then you never have any more jobs in that. See, you are initially; say let say you are running a batch computing center right, so there are hundred jobs waiting and you have each job requires, right, all these devices and CPU and so on. You can simply just put one job at a time. You are not going to get much through utilization of the devices. You will get very low delay for this particular job, but other jobs will wait for such a long time because they just sitting, waits, sitting outside the queues, the entire network waiting.

So, if you find out that, I can actually feed in five jobs at a time and I find that the response time is still very close to the D , then that is satisfactory. If I go to seven or eight, then the queuing delay starts increasing, response time starts increasing. So therefore, I that there is no benefit in keeping more than five jobs system, that is all.

How is this...

You both that example, this is a window size not the buffer size. You want to operate your system at the appropriate window size. The number of packets I can have outstanding; for example, right. That is my N , and that is going to... If I inject too many packets on the network, I am going to end up with queuing delays right. So, that is why, that is how I was trying to write that. In oral, length enter will also be the 100 percent of the bottle neck.

Yes. When you get N^* , you are getting one by D , yes. You are closer to the utilization of the bottleneck. So, N^* equals one over D max which means that that device is close to one.

So, in this example, what is N^* ? So, whatever five jobs, six jobs, whatever it is and take that and then that is your idea, but now there is lot of approximation here. You are assuming the service time, your exponential right, this is very, you know this is only a trend analysis. You might find that sometimes particular mix of jobs; they are very large, lot of jobs that require fewer visits to many of these devices. And therefore you may find the utilization that actually is lower.

So, you may have to go back. Well, I can feed two more jobs in system. Still, does not really change the performance as such, so use that some sort of palpate figure and then work around that say five or put 10. And some queuing, I am willing to live up with just because the distribution of times, like an average. I might not be seeing that worse of performance right, say twenty percent compared to the minimal; still, I can live with that. So, these are design level decision by gives you just a guideline as to what, how many number of jobs you should operate in the system at a given point and time... So, this is your chapter 33, this is your operational laws. So, next chapter deals with actually how you handle the specifics of trying to find out the various values. We will see what those.