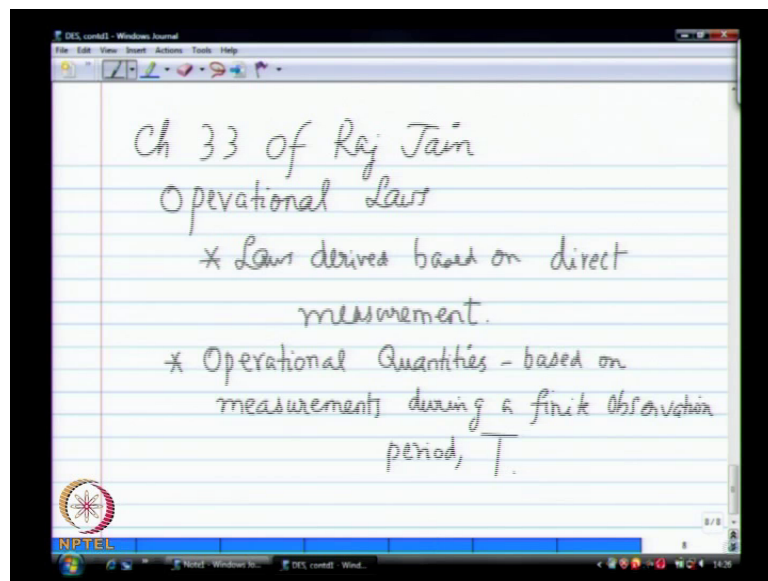


**Performance evaluation of computer system**  
**Prof. Krishna Moorthy Sivalingam**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture No. # 24**  
**Operational Law-I**

(Refer Slide Time: 00:10)



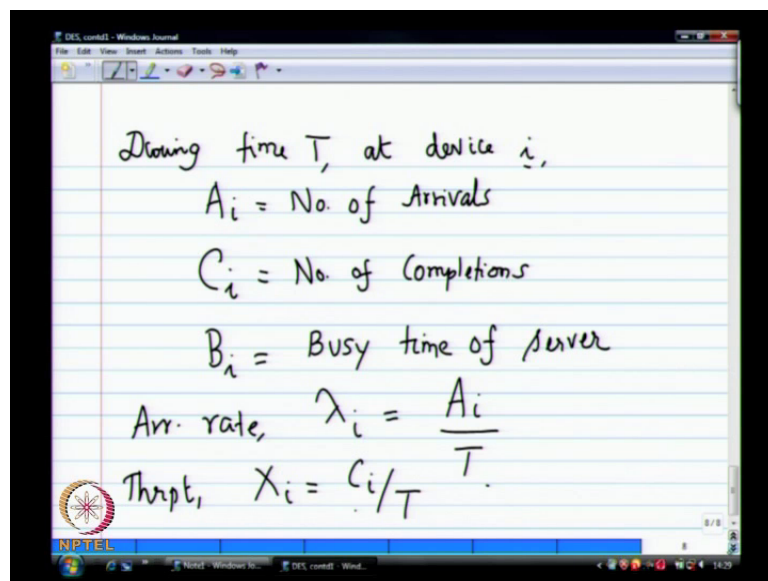
So, now we come back to ... So, these are so called operational laws. So, some of one of these laws we have already seen at Little's law. But this chapter provides the foundation for the close queuing network analysis, we will do in subsequent classes.

So, what is this word operational mean? What is an operational law mean? Something that is based on the system understanding the operational of the system; so these are laws that we can actually measure right, we can verify empirically. If I tell you that memory less properties, for example, that is a little bit hard to verify right. Some, some of we will see one of the laws which is actually memory less, which is based on the memory less properties. So, some laws we can observe in a system and actually experimentally measure and empirically verify that this law holds true. Whereas some other laws we are based on based on some sort of mathematical principles, and just use that as a law right,

even though it is kind of technically hard, might be physically impossible or infeasible to actually verify those.

So, operational law simply right, means that the laws derived; so based on direct measurement; the actual rules are fairly straight forwarded. And likewise, we will also use, we will measure; we will make some measurements, right. What are these measurements? These are called operational quantities. So, these are based on measurements during a finite observation period finite observation period, which will flung, now on use this T. So, something that I can measure, number of customers arriving per unit time, right; I can measure sit there for two hours with some bank or whatever and then measure the number of customer arrive in to our period. Then I can say average arrival time equals right. So, that is the measurable quantity.

(Refer Slide Time: 03:13)



So, based on these measurable quantities, we will try to pick up some laws so right. So, during time  $T$ , so  $A_i$  is the number of arrivals. Then we measure the number of completions right. So, how many customers complete; so this is measurement arrivals at whichever single queue or entry to the system or a queue within the system. Likewise, completions at queue inside the system; I can also measure the busy time out of the server right.

So, now I think you just know when the server is busy, when the server is not busy. Previously, we had this  $P_{\text{naught}} = 1 - \rho$ ; that is derivation. But here you

can simply monitor a system and then find out how half or what fraction of time this particular device CPU or disc drive has been busy. So, this is the busy time of the server right.

Once over, I am looking at how many customers arrive, and how many customers leave and fraction of time; that the time, the actual time and this is, right. So,  $T$  is the actual time and  $B_i$  is also in to the units of time. So, we know some standard rules right. So, the arrival rate for this queue, right, we will say at some right, this is device  $i$ , so that is where all the  $i$  subscript come. So, there are several such devices are used. And at one particular device  $i$ , making this measurements arrival rate to this particular device is  $A_i$  by  $T$ . Then the throughputs, which you will call  $X_i$  simply. Nothing fancy so far just basic derivations.

(Refer Slide Time: 05:42)

The image shows a handwritten slide with the following content:

- Utilization,  $U_i = \frac{B_i}{T}$
- Mean Service Time,  $S_i = \frac{B_i}{C_i}$
- ① Utilization law
- $U_i = X_i S_i$  [  $\rho = \lambda \frac{1}{\mu}$  ]
- $U_i = \frac{B_i}{T} = \frac{C_i}{T} \cdot \frac{B_i}{C_i} = X_i S_i$

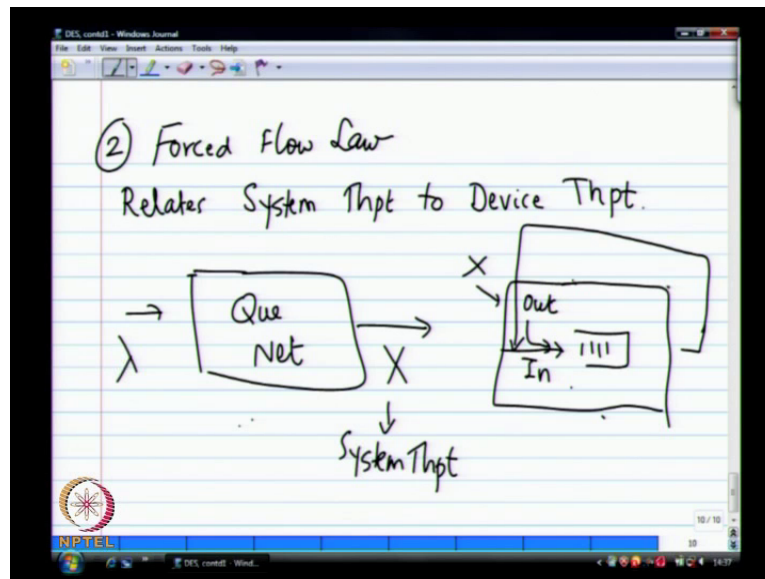
Then utilization; so this is the fraction of time that is utilized which is simply  $B_i$  by  $T$ . So, these are all now service time; right, mean service time. So, when we say service time is exponential, that is not... That can be difficult to measure, but here this is the mean service time right. Mean service time of a customer in that; that is when time in this queue is, what it is going to be... Basic time by completion right. So, therefore server was busy for  $B_i$  time units and then there were  $C_i$  completions. Therefore, this is the average service time. This is the service time not the queuing time, only the service time. So, we will have some four or five laws. So, these are so-called the utilization law.

As I said, by themselves the laws are trivial, but they help us in arriving at it. Properties are performance matrix for the queuing system.

So, the utilization of the server equals the throughput times, the average service time. So, the number of customers completed into the...; number of customers per unit time into time. So, this is the  $\rho$ . And this by the way is nothing but if you look at our old M/M/1 right. What is this?  $\rho$  equals  $\lambda$  into one by  $\mu$  right, was definition. Now, your  $\lambda$  is nothing but your throughput right. In any system, in M/M/1 system, like this, your  $\rho$ . In a given queue, number of customers should equal the number of **number of** arriving should equal the number of customers departing right. We saw this earlier on; that the departure is also a Poisson process with  $\lambda$  right.

So, in this case, your throughput equals your arrival rate. So, that is what this is in. If you really have to derive this; can you derive that. So, what is  $U_i$ ? This is  $B_i$  by  $T$ . So, what a... how do I compute this...? Just multiply, divide by  $C_i$ . Right. So, this is  $C_i$  by  $T$  into  $B_i$  by  $C_i$ ;  $C_i$  by  $T$  is  $X_i S_i$ .  $C_i$  by  $T$ ; that is the average number of customers finishing per unit time right. And average service time really means nothing else, right. It is not a proof. You can verify this right. We have only one system there. Just go there and measure for some unit of time, right, measure the time that the server is using; number of customers arriving, number of customers finishing and they have actual utilization time. You can start with this time when the server is empty. When it is partially busy, it is kind of difficult, because there is some part of the service time right from the previous.

(Refer Slide Time: 10:14)



So, if you really want a... We have done these examples before. So,  $U$  equals excess. Then there is a system called forced flow. So, this law will try to relate the overall system throughput; here, device throughput. So, have a system of queues; packets entering, packets leaving. So, I can measure the overall system throughput at the departure point. I have several queues inside the system. So, I want to measure the throughput in terms of the number of packets leaving every device, which I am trying to connect device throughput and the system throughput. So, that is what this is right.

So, throughput for an open system is clear; simply the number of customers finishing completion. Throughput for a closed queuing network; we have this notion of where there is one line that is connecting in to out, right. We measure the number of customers entering that link, the connection of the link out in link. There is one link; there from the out in link, you measure the number of, average number of customers travels in that link. That link will be called as throughput.

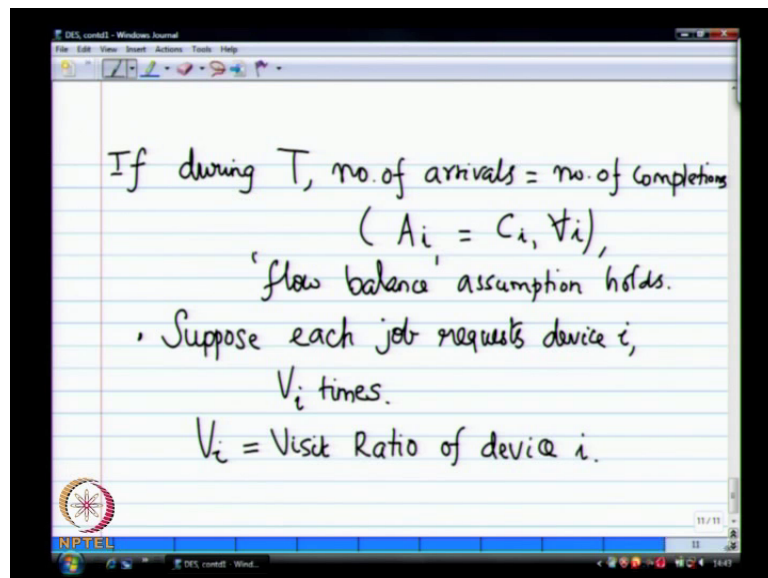
So, if I have an open queue system; this is a big black box right. There customers enter customers the, right. This is a queue network, so here this is where you measure here. we will call this  $X$  as a system throughput. This is your arrival right. And this is your system throughput. Whereas if it is a closed queuing network, like we seen before. So, there is one special entry point right and then lots of things happen. But then somewhere from

this system, from the system, right, I will have something. So, we have to specially designate as which is the out and which is the in.

And then that the customers that are traversing this link; from the out link to the in link is technically... when we saw this in, for example, there is a multiprogramming system right. So, when a job goes through several queues, CPU device queues device queues, when finally it finishes, what happens to the multiprogramming? It gets replaced by some other job in the ready queue right. So, that is why there is always a continuous flow of jobs in the system.

So, the number of packets traversing this out-in link is technically called the throughput of the system. So, this, we will use like that. So, this is this is your throughput from out to in. Look at the average number of customers traverse.

(Refer Slide Time: 13:38)



Then, we have so-called flow assumption. If during time T, so the number of arrivals right, at this... Other Words, this job does not go into the system and get extract. So, whatever enters, that ultimately has to leave the system. See, that can happen, anyhow a finite queue right, when you actually accept the packet, once the packet comes in, drop the packet, then so what? So their completion is not equal to arrivals right. In that case, this assumption does not hold true. Flow balance assumption, whatever accepted for service actually completes and finishes the service could be in partial service; it is some

other interim queue, but it is not dropped by the system as such right. So, if this condition is true, then we can have some results right.

So, this is your flow balance example of a system where flow balance is not maintained. have u seen IWC, one of those episodes of it anybody if you heard of Lucy. Lucy's... My god. So, definitely there was a big huge generation gap. If you look at Lucy's... so she was back in fifties. But anyway by the time she was not watching T V, she has already almost forty years old. But still, so there she is operate, she is employed as an operator; that is, working in a chocolate factory. So, her job is to, this pipeline chocolate. You know, pieces go by, simply take it, wrap it, right. And then this keep it, put it themselves. So, the chocolates come this way, you have to take it, wrap it and keep it, put it back in the pipe line. So, this is your, jobs coming in and jobs going out.

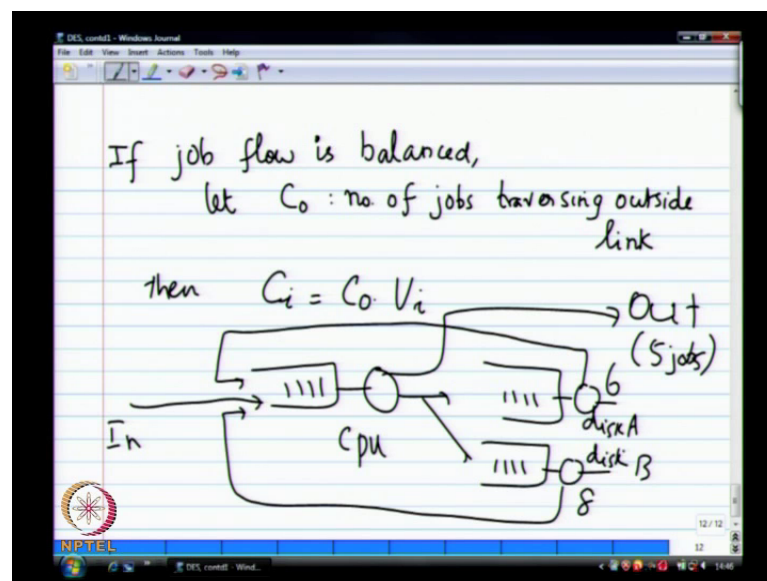
So, initially she is the pipeline the... Conveyer is operated very slowly. So, she is able to, right, she used to casually take on and then and then start doing this. Then of course start increasing the speed and that is when she eats. It is more of drastic comedy than anything else. So, if you, if you can download this one in you tube, Lucy's... So, then they start setting this belt, it is so fast that she has received, she cannot take everything and then at the same time, put it in, wrap it and send it out. She simply eats; she keeps on eating half way through. And that is all, that is, there is a job where system, where flow balance does not hold; because the number of chocolates coming in is not equal to the number of wrapped chocolates going out. And again only if you see that, then probably; it is more hilarious, when you see than when it is described.

So, this is one assumption that we make, right; jobs leaving equal to the jobs arriving for some observation period. Now, we will have to define something called a visit time. We did visit, earlier on when contexts same Markov, similar definition here, right. So, suppose each job requests device  $i$ ,  $V_i$  times. So, where does this happen? So, you have system regarding on the CPU, right. Even if it is simply running on the C P U, you run on the CPU once, then you get you get kicked out, because your quantum is up. So, you come back to ready queue; then again, it is again executed at some other point at time. So, simply might request  $V_i$  times CPU will be, you will ask for the CPU,  $V_i$  times right. So, you will run that many times on the CPU itself or you can done over the CPU once, then go to the disc, get processed and come back to the CPU and then run again and go to another disc and so on.

So, you have this repetition of asking for the CPU; asking for disc a, disc b, and so on right. To our some period of time, you make a measurement that this again is an average term right. Because it is possible that one C P U process; one process will request disc a twice, others, may discuss or require the disc, sorry, ten times and so on.

So, we are sort of, you know hiding away all that; assuming that, all jobs will request each device for the same number of times right, that is one simplification that we are looking at; which is null. Whereas real system that is not really the case right. You will have multiple... So, this is a, so  $V_i$  is so-called visit ratio; visit ratio of the device  $i$ ; this is for a single class system; there are multiple; you can derive this queuing networks for also systems in multiple classes right. Where each class of traffic as different requirements; each class of job that is running in the system has different requirements. For now, this is a single class system.

(Refer Slide Time: 19:34)



So then if the job flow is balanced and let me write  $C$  naught; this is  $C_0$ ... Sometimes can be also called as  $C_0$ , because all the devices are numbered one through  $n$ . So, this is the number of jobs traversing, this is basically your throughput right. So, this is simply the number of jobs that is traversing on your outside link. Then your  $C_i$ ; this is defined to be  $C_0$  into  $V_i$  right. Every job request device  $i$ ,  $V_i$  times, there are total of  $C_0$  jobs that have left to the system is observation period. Therefore, the number of times, the number of completions of device  $i$  will be simply the total number of



completions into the number of times at each that particular devices requested by a given job right.

So, if number of jobs finished is five and each job requested a disc ten times, then the number of completions of that particular device is fifty completions. You are treating them all as independent right. So, even though the same job is come to the this ten times, it is technically speaking they are all independent completions. So, that then that get, it is kind of struck. That is where the we are assuming that in this measurement period, jobs came and jobs left and that there is nothing left behind in the system, because if you have queues, definitely you have at least buffers.

So, let us say that, this is at device queue that we are talking about. right. So, there is a job that comes here, that goes to these two queues right. So, systems, right, this is your arrival, whenever job is finished always comes to the C P U disc right. This is your C P U, this is disc, this also disc right. Let a, this is disc a and disc b, so we are now assuming that process come, then request this C P U whatever, some number of times. It runs some times on the C P U, then it goes to the one of the two discs, then it comes back. There is no self-loop here right.

So, this is requested here. Then I go to this disc A, I come back in process in the CPU, then I go to disc B and so on right. So, if I am telling you that, now I have not defined mean this is my inline right. Then from here I also have, so these are the two inline options. I will also have an outline. This is the outline. So, I know that there are five jobs in this observation period that develop the system and if I know that every job requested will request disc A six times and disc B eight times right.

So, when I start the system, there is nothing in the system. When I am finishing the system, observation period that is, right, there is nothing in the system. So, if five jobs in the system; so means that this same each job must have come to this disc six times. Therefore, the number of completions, the number of completion of this is simply thirty and disc B is forty right. That is what that is I am trying to say you. So, the number of completions that is, device is simply proportional to the total number of completions and the visit ratio for that particular device.

So, even if there are small numbers, I mean because we are mostly looking for the performance strengths right. They can safely ignore that; means, say that this is one sort

of regeneration period, where it started with some nothing, and ended up with nothing again, right.

(Refer Slide Time: 25:57)

The image shows a whiteboard with the following handwritten text:

$$V_i = \frac{C_i}{C_0}$$

$$\text{System Thpt, } X \triangleq \frac{C_0}{T}$$

$$\therefore \text{Device Thpt, } X_i \triangleq \frac{C_i}{T}$$

$$= \frac{C_i}{C_0} \cdot \frac{C_0}{T} = V_i X$$

A box is drawn around the final equation:  $X_i = V_i X$

So, therefore visit ratio is  $C_i$  by  $C_0$ . So, where this law is useful is, I can sometimes simply measure the number of completions, right, at each queue that you can simply sit and watch. I have networks of queues; I can measure the completion at device  $i$ , I can measure the overall number of completions leaving the system. Therefore, I know that the number of visits to this particular device has to be this much; again as a mean, right.

Therefore, the mean number of visits to these devices is simply, this by that. That is, it is not that every job. You are just assuming that every job makes at  $V_i$  visits, it is possible that let us say five of you have to go to the  $D_r$ 's office the  $A_r$ 's,  $A_r D_r$ , and then register for some signature. And then they keep you in a loop ready to go from this, go to the  $d_r$ , then come back to the  $d_r$ , again some several loops you do. Then you just sit and measure the number of students leaving this  $d_r$ 's office. That is the total number of request made in that device. As you know, only ten of you are there that finally leaves the administration building, that is your  $C$  naught.

So, you know that you each of you have to spend. Even though, possible that once made twice, other, some other should have gone eight times and so on. We are simply counting the total number of requests made to that particular system. And using that as a mean approx.; mean only we are looking at... So, that is your so-called force flow law.

So, the  $C$  naught  $V_i$  need not be the value that comes. Yes, it could be also from the past jobs that are there in this system, but then assumption is that some of these guys also get struck in the same queues right **right**. So, it is a... What it is, so what, so for example, let say there are two customers already in this system. right. And then you had eight customers coming in and then in that time period that you are observing, these original two guys also left and six of these new arrivals also have left. If we take that as a very, so these are all they are very gross assumptions, Right to give us just some sort of indication of the performance.

So, with this, now we can derive couple of other laws. This is the system throughput  $X$ ; It is defined as  $C_i$  by  $T$ . right. This is the system throughput, so therefore the device throughput that as  $C_i$ , **sorry**  $C$  naught. This is the overall system throughput, which is  $C_o$  or  $C$  naught, whichever it is. And this is simply, this is again by definition right  $C_i$  by  $T$ . This is what we are defined earlier on which is nothing but... So,  $X_i$  equals, so again where this is useful is, once I figure out the system throughput, I can actually figure out the device throughput because I know the corresponding visit ratios.

So, the... We will see examples of where this will be used, right. Now, it is just you know, you just coring out these derivations. You will be, usually what will happen is, you can measure some parts of the system. On from that, you can make some derivations in terms of overall waiting time.

(Refer Slide Time: 27:41)

$$u_i = X_i S_i$$

$$= X V_i S_i$$

$$= X D_i, \text{ where } D_i = V_i S_i \text{ (Demand time)}$$

Device with highest demand is  
the "Bottleneck" Device.

And then next we have our  $u$  equals,  $u$  equals  $X_i S_i$  as  $u$   $X$ 's true. This I can rewrite now; because of this, new derivation is, so  $X_i$  is nothing but  $X$  into  $V_i$  or  $V_i$  into  $X$ . Right.

Now, I can combine this  $V_i$  into  $S_i$ . What is  $V_i$  into  $S_i$  actually represent? It is total time, actually number of visits and means service time for device. Ok. There was, it will give us total. So, ok, I will answer, then will come back and answer this again. So, totals that going to be changed.

It will give total time at that device **yes**, total demand of the device so-called demand; the amount of time that the total number of visits to the device,  $V_i$  into the total number of mean service time, total time in the **...** So,  $V_i S_i$ ; come back to  $V_i S_i$ , come back to  $n_s$  two. So,  $V_i S_i$  is simply the demand made by a given job on a device  $i$ . So, this is represented as  $X$  into  $D_i$ , where  $D_i$  is the number of visits made by given job to device  $i$  and the service time for each visit. So, this is so-called demand, right. So, this is the the unit of this is in this time, right, this is the demand time.

So, this is another rule that, again, when necessarily, sometimes you know  $X$ , sometimes you know  $D_i$  depending on what you measure. right. Then you can the compute  $u_i$  based on that. And this  $D_i$  is actually important in systems because ultimately what you want, when you are trying to conduct performance study of a system, then you look at the  $D_i$  across all the devices, you will find that there is always one bottle neck device... device with the highest  $D_i$  is so-called bottle neck device.

So, when you want to redesign a system for a throughput performance, you will try to go and attack this device with highest  $D_i$ . And then somehow reduce its delay time; reduce the demand on the particular device. So, there is a demand to make the device faster. Right. You can cut down  $S_i$ , for example,  $S_i$  can be reduced for making the device faster, either a faster disc or a faster link or something like that. So, that you will bring down your, the bottle neck in the system. This we will... later on we will see how this is **...** right.  $D_i$  will essentially impact overall performance of the system. in  $D_i$  will be something  $n$  by  $n_d$  plus  $z$  and so on. We will see. So, the device with the highest demand is the so-called bottleneck device. So, back your question  $n_s$  two.  $n_s$  two, you do not get, see the internals as clearly as this very high level  $t_c$  l's scripting what did you doing so. Yeah. This is more generic. And everybody here will not work on networks

tomorrow. You will work in anything else. So, you should be able to build discriminant stimulation system for any other system, not just networks. We are reducing amount of time, but not the number of visits. So, it is a kind of low balancing system. So, where is  $q$  c.

Yeah, it is, it is not the because the number of visits depends upon the jobs, right. What it is going to do, if it is going to have several discrete and discrete and so on, there is something, which you cannot control from the system perspective, right. You have a job with the particular behavior. And the only thing you can do the system to improve performance is to give faster devices that will essentially bring down your  $S_i$ , right.

So, that is what you want. If you want to say which of these devices are the bottleneck, right, then you should try to make that faster. So, if you go to the bank, for example, this printer is always a bottle neck. Then you know that printer is slow. Everything else you get processed, processed. Then you go to the bank and then used to wait for your D.D and that things take for an hour to print. And that is where your bottle neck is. So, you would say okay I know that this device has to be might faster, if you want to increase the overall throughput of the system. Otherwise, people get stuck at the bottle neck. So, same thing here, bottle neck device for the place where lot of jobs queue up, therefore end up in having longer queue because in relationship with that now for some numbers.

(Refer Slide Time: 33:13)

Eg Time sharing System, Accounting Log  
→ CPU, Disk A, Disk B

- \* Each program req. 5 seconds of CPU time;  
Makes 80 I/O req. to disk A  
100 " " " " disk B;
- \* Disk A takes 50 ms/request;
- \* Disk B " " 30 ms/request.

17 terminals ;  $X_A = 15.70$  jobs/second

So, we have, there is a time sharing system. And like any system, you always have one... right on logging. So, you look at the accounting log and you will get some accounting information from this log. So, this is the system with the C P U, this is a disc a and b. This is a standard example that we have seen so far. So, these are all the components of the system. Every job gets submitted to the C P U, but it goes several times to the discs before it. Finally, completes and leaves the system.

So, there are some things that we have measured. So, each program requires 5 seconds of C P U time. Then some other measurements we made. It makes 80 input, output requests, disc A hundred, disc B eighty input output requests to disc A hundred to disc B, by measurement we have found that disc A takes fifty milliseconds per request.

So, just have time, the disc, the disc B requires only thirty milliseconds per request. Now, in this, system is like a old fashion time sharing system. There are terminals, which are connecting the C P U and so on. And so there are n, seventeen terminals, normally there are n terminals. Right now there are seventeen terminals, which are recycled in this job. So, the job terminal; the basic operation of this time sharing system is each terminal generates a job and then it gets processed to the C P U, it comes back to the result. And then the terminal goes in to the thing state for some period, then it again regenerated. So, right now let us not look thing state. There are seventeen terminals. We will see how that is useful right. There are seventeen terminals in this, connected to this computer right. So, there are seventeen, essentially active, either they generate jobs to the system and throughput of disc A, right.

So, we have, from the large, again calculate the throughput of disc A is 15.7 jobs per second. These are all the measured parameters, measured values from your system. So, ultimately I want to find out the utilization of all the devices in the system first. Delay comes later. Just say, I want to find out the utilization. I need, want to see what is the total delay, total throughput and per delay and per device throughput per device. So, let us just try to find out the utilization of each of these three; C P U, disc A and disc B.

(Refer Slide Time: 37:28)

$$V_A = 80; V_B = 100; V_{cpu} = V_A + V_B + 1$$

$$= 181$$

$$D_{cpu} = 5 \text{ seconds}$$

$$D_A = S_A V_A = 0.05 \times 80 = 4 \text{ sec.}$$

$$D_B = S_B V_B = 0.03 \times 100 = 3 \text{ sec.}$$

$$X_A = 15.7 \Rightarrow X = \frac{X_A}{V_A} = 0.1963 \text{ jobs/sec}$$

So, how do we proceed, what do we know what is that you need to find out. We know  $V_A$  right. So, let us, so we know  $X_A$ , which is the throughput of this particular one device right alright. So, we know the visit count. What is the visit count of A? Eighty; visit count of B is hundred, visit count of C P U is how many times? So, eighteen hundred is given to us. How many times, job? 180 plus 1 plus because you have one final execution were we do not go to the device in finally, right. So, job comes, runs on the C P U, goes to disc A, comes back, runs on the C P U, goes to disc A or B and so on, right. So,  $V_{cpu}$  equals  $V_A$  plus  $V_B$  and finally, there is one execution where you simply say exit. You do not go to either of device.

So, there is always last quantum of execution. So that,  $V_B$  plus one, so every time you went to  $V_A$  or  $V_B$ , you have to, you go to a  $V_{CPU}$ . Therefore,  $V_{cpu}$ ; therefore,  $V_{cpu}$  is 181, then we do, we know the demand. So, what is the five seconds? That is your  $D_{cpu}$ . right. We have measured and found out the  $D_{CPU}$  is right each job required five seconds of C P U time. That is what something evolution. So,  $D_{cpu}$  we know and  $S_A$  and  $S_B$  also we know.

So, we have  $D$ . So, from  $D$ , let us try to find out the other things. So, what will be the utilization of the disc? This is simply  $S_A$  into  $V_A$ . right. We know both parameters. So, I know that is the  $D_A$ . As for see as, we saw they can this is just these are just routine computations, right. So, say, so 0.05 seconds into 80.

So, this is four seconds. So, we know that  $X_A$  equals 15.7 right. Now, which I need the final system throughput. So, let me say I find out  $X$ .  $X$  simply  $X_A$  by  $V_A$ , right. So, actual throughput of a system is just 0.1963 jobs per second.

(Refer Slide Time: 40:56)

$$S_{cpu} = \frac{D_{cpu}}{V_{cpu}} = \frac{5}{181}$$

$$X_{cpu} = X \cdot V_{cpu} = 0.1963 \times 181 = 35.48 \text{ jobs/sec.}$$

$$X_B = X \cdot V_B = 19.63 \text{ jobs/sec.}$$

Now, I want to get the  $u$  value, right, the utilization. So,  $u$  is different on  $u$  equals  $X \cdot S$ .  $S$  is, you know, right. So, we know  $S_{cpu}$ , we know  $S_A$  and  $S_B$ , what is  $S_{cpu}$ ? How do we get the  $S_{cpu}$  service time? You know  $D_{cpu}$ , you know  $V_{cpu}$ . right.

So,  $S_{cpu}$ ; simply,  $D_{cpu}$  by  $V_{cpu}$ ; which happens to be 5 by 181, what is that? so we will figure it out. So, now that I know the overall system throughput right. This is  $X$  into  $V_{cpu}$  right. That is nothing but 0.1963 into 181. So, that is , so throughput of the CPU is basically thirty five jobs per second right. This is what the throughput of the CPU can handle when we can compute the throughput of this disc B, which is again  $X$  into  $V_B$ . So, that will be 19.63 jobs per second. So, we can write a very small program. Do this; only thing is you have to know this will be sometime missing right.

So, in each of this like a symbolic program that says like in mat lab, solve this, this, comma right give you something is  $X$  something is not given. Then we will simply split out the values of  $X$  or right, some known or some unknown, some magically figure. Right. So, the better way is to start with; basic utilization start with  $u$  equals  $X \cdot S$ . so we need to find out  $X$ .



So, actually I am working this from bottom up the other way. should go you know the yes in this case and some use even know the yes little bit some other value. So, you have to find out X. So, to get X, I need to know the throughput of the system. Therefore, I know throughput for one device. I can calculate the throughput for the entire system in the cane of back word.

(Refer Slide Time: 43:18)

$$\begin{aligned}
 U_{cpu} &= X \cdot D_{cpu} \\
 &= 0.1963 \times 5 \\
 &= 98\% \\
 U_A &= 0.1963 \times 4 = 78.4\% \\
 U_B &= 0.1963 \times 3 = 58.8\%
 \end{aligned}$$

So, now my X is and then out of the utilization values. So, this is X c p u into D c p u and this we computed as 35.48. oh! sorry, X, no **sorry**, it just X. u c p u is actually X into D c p u, it is X. n x as, so if i, this is simply X into D c p u. Your u c p u, so that means your c p u is utilized to the fullest. Now, you look at your u A, which is 0.194. So, we have computed the earlier D A to be four, right. So, this is 78.4 and u B equals... So, this is the status of your system. So, what is a big deal? All of this right, So, at the end of this, so we have taken some measurement from the log, we are now post factor computing what were the device utilizations and so on, right. So, as a system designer you are supposed to look at this and say, you know what my disc A and B or A is actually reasonably utilized; eighty percent utilized. Fine, but the C P U s is heavily utilized; ninety eight percent.

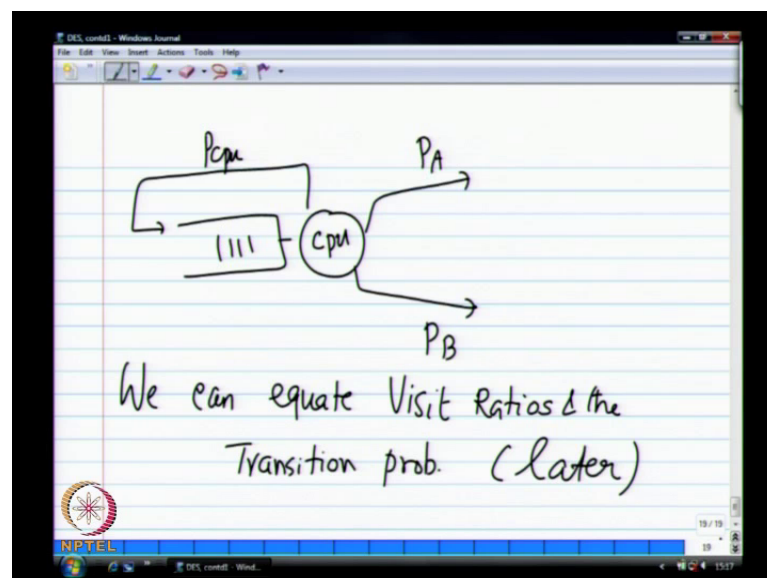
So, which means that, the queuing time at the time C P U is going to be very large. So, it is operating very close to its capacity, C p u is always busy. And therefore, jobs queued up will end up requiring, right, when having higher delays, because of the fact that the c

p u is heavily utilized, right. And disc B is actually underutilized; sixty percent utilization is probably not worth that you have paid some million dollars for the disc; in those days, not now. Then your not, you know, the disk is not getting utilized as that, so we can do much of course. Because the disc utilization depends upon the where the jobs are running and how often they read and write from this. right.

So, you know, go back and say well I have this. So, which is highest D here, highest D is for the c p u. Right. Of these three, C P U is for the bottleneck. So, we will have to somehow speed up your c p u, bring in a faster c p u to cut down the service time of this c p u. And that, there may be bring down in your demand and then bring down your overall delay in the system throughput because number of job solution per second is very small. Have related to whatever it is. ok.

So, you never really use that seventeen here. We will use that in later... Come later on, but whether the seventeen terminals right. We are not really first in term this. So, only two rules that we really needed here, one is three two three right. One is at first flow, which tells you given, we have C naught. How do you get the corresponding visit and vice versa. And then V equals X S, and then the fact the D equals S into V into S. So, these are all the three things, questions.

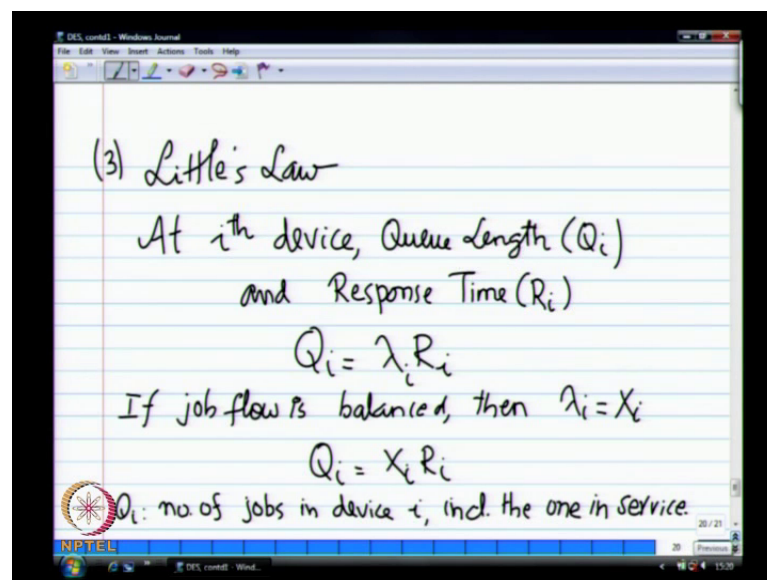
(Refer Slide Time: 47:27)



So, another way of representing this visit ratio is also as probabilities, right, which we will that is what we have been doing so far. When we did the first Markov chain into that

is a CPU and there is a probability right up going to this disc. So, probability of going to this disc and then there is also probability of right, coming back. You have seen all these things where the three probabilities should sum up to one right. There we looked up in terms of states. Now, we are looking in terms of queues with some packets and move between queues from one part of the system to another part of the system, right. So, we can specify this way to and essentially your, you can equate the visit ratios and this probabilities. So, if one of the other is known you can compute the other one. So, just put this down, we can come back and visit this here. We can equate the visit ratio and the transition probability.

(Refer Slide Time: 49:22)



And, the next law is Little's law. So, then at  $i^{\text{th}}$  device, queue length, so now we are getting into delay. So, far we did not look at delay. So, queue length and the response time; the queue length also includes the one that is in service, right. Response time is like we saw before the total delay that each customer is at the particular device. These are  $e f t$ . That is your response time that we saw before, right, this is separately. Yeah.

So,  $Q_i$  equals  $\lambda_i R_i$ , right,  $\lambda_i R_i$ . So,  $\lambda_i$  is number of arrivals in that device. So, we know this one. Before  $e f n$  equals  $\lambda_i$  into  $e f t s$ . Then if the job flow is balanced, then  $\lambda_i$  equals  $X_i$ . Remember, this was the assumption we made, right.  $A_i$  equals  $C_i$ . so then in terms of, right, it is simply  $\lambda_i$  equals  $X_i$ . So, in  $Q_i$

equals  $X_i$  into  $R_i$ . So, again to remind you, this is the number of jobs in device  $i$  including the one in service with the...

(Refer Slide Time: 52:26)

(4) Response Time Law

Q: Total No. of Jobs in System

X: System Thpt

R: System Response Time

$$Q = XR$$
$$Q = Q_1 + Q_2 + \dots + Q_M$$
$$XR = X_1 R_1 + X_2 R_2 + \dots + X_M R_M$$

So, that is you know; next is our so-called response time law. So, fourth law in the... So, if  $Q$  is the total number of jobs in the system,  $X$  is the system throughput, we know that from before and  $R$  is the system response time. So, this is the actual time taken for a given job as goes through the entire system. So, what is the relation that twice these three that is Little's law right. And  $Q$  is also, so the number of jobs in the system is simply the number of jobs that is queued up at  $M$  devices, right. So, devices are numbered to one to  $m$ . So, we have  $M$  devices, simply the number of jobs queued up. It is not possible for the job to have one like here and one like some other queue. So, the one queue are there ok.

So, this I can now expand with Little's law, right. So, if I divide everything by... Now, basically I want to find out the overall system response time. I can find out the individual queue response time, and then I want to compute the  $R$  s what I am looking to find out.

(Refer Slide Time: 55:05)

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

$$R = \frac{X_1 \cdot R_1 + \dots + X_M \cdot R_M}{X}$$
$$= V_1 R_1 + \dots + V_M R_M$$
$$R = \sum_{i=1}^M V_i R_i \Rightarrow \text{General Resp. Time Law}$$

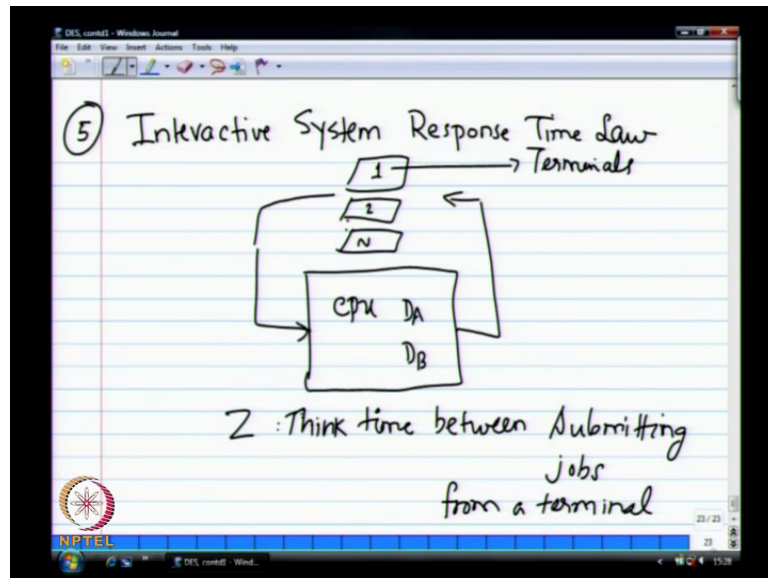
The whiteboard also features an NPTEL logo in the bottom left corner and a taskbar at the bottom with the time 15:27.

So, to get R, if I simply divide by X on both sides, then we will end up with... right. This is again very straight forward. So, what is X 1 by X? X 1 is the number of visits; total number of visits to make this particular queue. This is your V 1 right.

So, again in simply deriving this dot, it simply says that there if I visit each device X numbers of times, sorry, V number of times; that into the response time plus right. And this of course, this is where those independents, the our first form. These were the closed form comes in where I am independently analyzing this, when I am simply adding out we get right. This is this let be the closed form solution. Let me use this. Even though, it is intuitive, we actually show that this will work. We need the close... So, this is the overall system response time.

So, now we can proceed further right. So, we last time saw utilization, so utilization for utilization and then we can throughput, we can slowly start building up the... So you know finally, can get the response time of job that is a system and how we can... So, this is your general response time. So, if I give you the R s, the V s, you can simply compute.

(Refer Slide Time: 57:46)



So, now there is a special case called the interactive system response. So, this interactive system has this. You know C P U plus disc and all that in queues. So, it is like a circulating system, so you have a set of terminals. So, these are terminals. We will just assume that the number one, two, up to N. So, basically terminals interact the system will, right, you submit a job to the system, let say L S, this is what you submit in olden days and then goes to ... C P U get executed, you get back the response. And then you have to, then there is a think time before issue the next comment. So, there is this notion of think time between submitting jobs from a terminal time **time**.

(Refer Slide Time: 59:53)

Handwritten mathematical derivations for the Interactive System Response Time Law:

$$\text{Cycle time per user} = (R + z)$$
$$\text{No. of jobs}_{\text{submitted}} \text{ per user in time } T = \frac{T}{(R + z)}$$
$$\text{Total jobs submitted} = \frac{N \cdot T}{(R + z)}$$
$$\text{System Thpt, } X = \frac{NT}{(R + z)}$$
$$X = \frac{N}{R + z} \text{ i.e.}$$

So, each user cycles, right, each user generates a job which takes R times for service, then Z times for thinking. So, in a period T, right, so the cycle time per user is R plus Z. right. This is to handle the request from the system. Then the next is to, think before you submit next request, so the number of jobs per user, this is submitted right, in time T. simply T by R plus Z. rights. Z is the total jobs submitted because there are N terminals. So, this is a closed queuing system.

So, now the system throughput; so we are trying to find out R. right. That is where we are now heading to ... So, what is the system throughput? This is simply the number of jobs submitted. Again, it is the flow balance system. So, this is the total number of submissions divided by the total observation time.

(Refer Slide Time: 1:01:31)

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

In earlier eg,  $N = 17$ ,  $Z = 18$  seconds  
 We computed  $X = 0.1963$  jobs/sec  
 $\therefore R = \frac{17 - 18}{0.1963} = 68.6$  seconds.

Therefore, X equals N by R plus Z, where R equals N by X minus Z. This is your interactive system, response time. So, in the previous example that we saw; right, in the earlier example I said N was 17 right. Now, that is going to be used. N was 17, let us say Z equals 18 seconds and we computed X to be 0.1963 jobs per second. Therefore, R in this case, simply ...

(Refer Slide Time: 1:03:21)

Cont. W example,

$$X_{cpu} = 35.48 \quad X_A = 15.70 \quad X_B = 19.6$$

Let measured avg. Queue lengths

$$Q_{cpu} = 8.88, \quad Q_A = 3.19, \quad Q_B = 1.40$$

$$Q_i = X_i R_i$$

$$\therefore R_{cpu} = \frac{8.88}{35.48} = 0.255; \quad R_A = 0.203; \quad R_B = 0.071$$

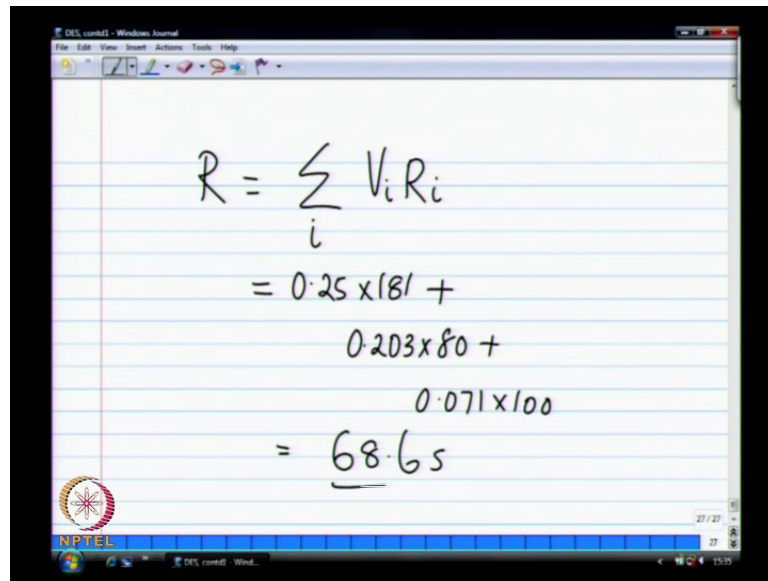
So, we will do one more and then only bottle neck also. So, we will not go to bottle neck analysis before the quest. So, we just finish off with the same. So, we continue with the example. So, we had computed  $X_{cpu}$  to be 35.48 and all that right.

So, this was the throughput of these various devices in previous, example right. So, let measured queue lengths, average queue lengths, the  $Q_{cpu}$ ; so the CPU had 8.88 jobs. So, these are the numbers, which we can just take times snaps. Look at the each other queues and then compute average queue, right. So, this is average queue length that is given. So, we have  $X$ . So, we have  $Q$ , therefore we can find out  $R$ . right. So, given that  $Q$  equals  $X R$ , like this is rows appear to four, right. So,  $Q_i$  equals  $X_i R_i$ .

So therefore,  $R$ , so the response time that is in CPU  $Q_{cpu}$  8.88 divided by 35.48. So, that is 0.255 seconds. Likewise  $R_A$ , you can compute this, this giving the values, but you can compute the others, right  $R_A$  is simply 3.19 divided by 15.701.



(Refer Slide Time: 1:05:50)



The image shows a digital notepad window titled "DEE control - Windows Journal". The notepad contains the following handwritten mathematical derivation:

$$R = \sum_i V_i R_i$$
$$= 0.25 \times 181 +$$
$$0.203 \times 80 +$$
$$0.071 \times 100$$
$$= \underline{68.65}$$

The notepad also features an NPTEL logo in the bottom left corner and a system tray at the bottom with the time 15:27.

So, once I get the R s, now I can actually compute the overall response time also different period in one way. I can do it other way right. So, this is equal to  $V_i R_i$ . So that is nothing but 0.25 into 181 plus 0.203 plus only one thing in this book, now and then they have examples that have in correct calculations. In older version you might have all those changes. So, you have to go back in check of individualistic 0.203 and so on always check. In this particular case it is correct, but there are many other cases were they have, so this is 68.65.