Digital Switching

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

Lecture - 19

Student: We are calculating the waiting time of the two components, w 1, w 2. So, what is the queue disciplinary output?

Output is a queue disciplinary output will be first in first out.

Student: Its true now we are w m I tower delay after v is eliminating w 2 w 2 at with this w 2 is having passage I here consider out of passage we.

Within batch, you can be placed anywhere.

Student: Anywhere, but we are selecting randomly?

Yes, it is with equal probability. Your tag packet will be equal probability can be anywhere, in the breath size of i, but anything with the earlier batch is s 2 always, go first. This is the first in first out among batches; within a batch, it is a randomly; random placement.

Student: And when we are doing the timing.

Taking as just a concept, actually, because actual timing is not happening in that (( )).

Student: Because we are calculating to probability tag given passage of time we are assuming that size I one already we selected n minus one we have selecting c minus I i minus i.

It is not clear.

Student: this calculating probability tag batch w 1 back side of i.

At the output, whether the batch size is coming or not; what is the probability of that?

Student: Out of n d s a we are taking.

Maximum batch size is n, minimum is 1. What is the chance that if a given slot get an output port at a batch of I will come.

Student: I will come, but one packet is already tagged with this particular packet.

Out of this i, one is my packet.

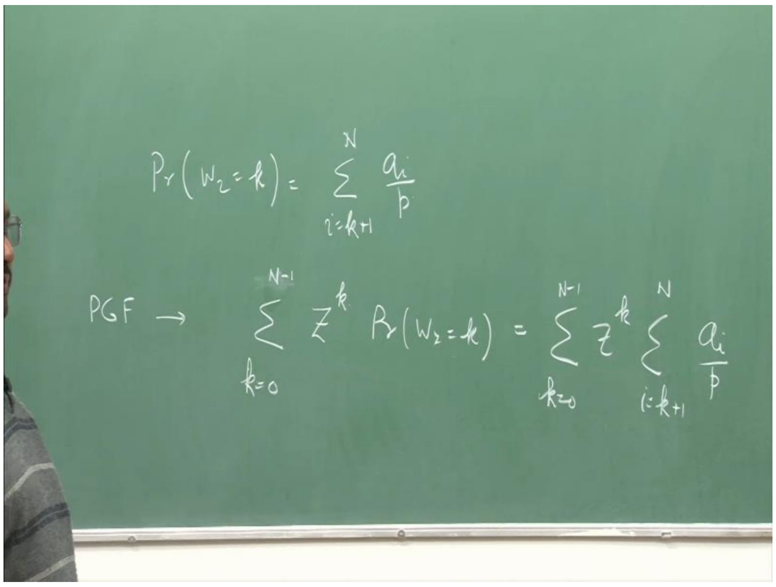
Student: My packet.

And it has the best size, has to be actually, moved greater than or equal to k plus 1 for w 2 is equal to k.

Student: Is to, but why it begin n minus one c I minus y.

One is my already tied packet, which is coming from one of the ports. Now, from remaining n minus 1, i minus 1 should also come. Then, only batch of 5 will be made. That is, anyways, part of it. That is the reason why it was done.

(Refer Slide Time: 02:32)



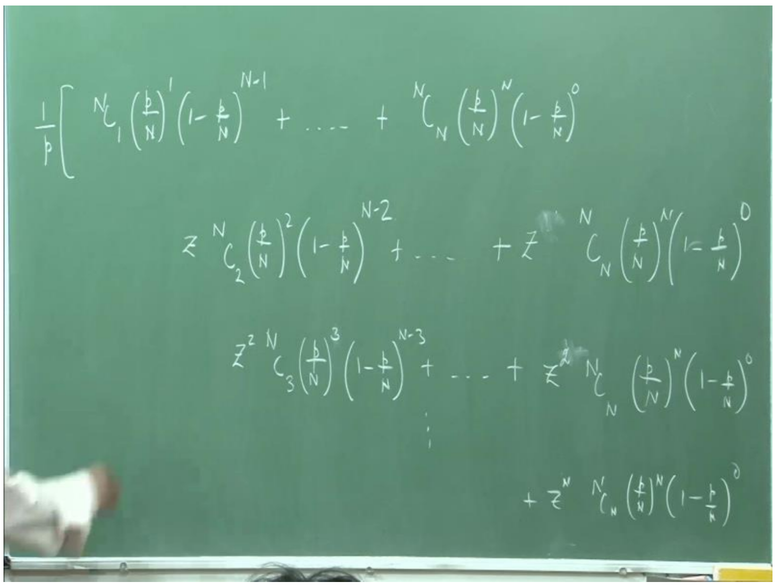
$$P_r(W_2 = k) = \sum_{i=k+1}^{N} \frac{a_i}{p}$$

$$PGF \rightarrow \sum_{k=0}^{N-1} z^k P_r(W_2 = k) = \sum_{k=0}^{N-1} z^k \sum_{i=k+1}^{N} \frac{a_i}{p}$$

Ultimately, yesterday, what we called was probability that your tied packet will be there, and it looks here, suffered a delay of k; this is what we had estimated. This was nothing but summation of a i by p; i goes on k plus 1 to n; that is what we actually, had got and ultimately, from here, I can find out what is going to be a g transform of this, which is a probability generating function, p g f. So, I have give a summation for e going from 0 to n, because that it the maximum, which is possible; the probability. Your batch that w 2 delay cannot be more than n, because your batch size cannot be more than n. It will maximize of what you get is actually, n minus one delay. It is not n, but n minus 1.

So, you have do z raise power k and then of course, whatever is this probability; then w 2 is equal to k, has to be put here, which in turn, will give you . So, this actually, can be solved to find out what is going to be our probability generating function, w 2 z. So, this is w 2 z; that is what we are interested in. So, you can actually, now solved for it. So, this one is pretty straight forward once, I give the technique.

(Refer Slide Time: 04:39)



$$\frac{1}{p}\Big[\ {}^{N}C_{1}\left(\frac{p}{N}\right)^{1}\left(1-\frac{p}{N}\right)^{N-1}+\dots+{}^{N}C_{N}\left(\frac{p}{N}\right)^{N}\left(1-\frac{p}{N}\right)^{0}$$

$$z\ {}^{N}C_{2}\left(\frac{p}{N}\right)^{2}\left(1-\frac{p}{N}\right)^{N-2}+\dots+z\ {}^{N}C_{N}\left(\frac{p}{N}\right)^{N}\left(1-\frac{p}{N}\right)^{0}$$

$$z^{2}\ {}^{N}C_{3}\left(\frac{p}{N}\right)^{3}\left(1-\frac{p}{N}\right)^{N-3}+\dots+z^{2}\ {}^{N}C_{N}\left(\frac{p}{N}\right)^{N}\left(1-\frac{p}{N}\right)^{0}$$

$$\vdots$$

$$+z^{N}\ {}^{N}C_{N}\left(\frac{p}{N}\right)^{N}\left(1-\frac{p}{N}\right)^{0}$$

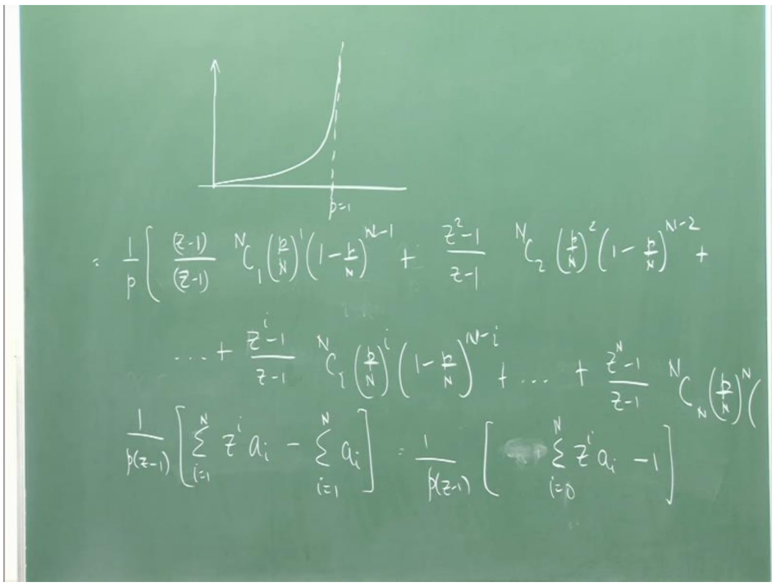
So, 1 over p; I can always take out, and then of course, a i, we already know, and I can solve. When k is equal to 0; for that, there will be a series, multiplied by z raise power 0. For k is equal to 1; there will be a series, number of times, we start reducing as my k increases. So, I have to actually, (( )) write all these terms, and I will just reorganize then. It has many ways of doing it; I will just do it this way, because that is the way I have learned it. So, for k is equal to 0, what is the series I am going to write in the first row? This will be nothing but n c 1.

This is for the k; i is equal to 1 and then of course, you will have the series. I am not going all the terms, but that corresponds to when k is equal to 0. When k will be equal to 1, in that case, all terms will be multiplied by z raise power 1. I will actually, have; because k is 1, i has to be 2 minimum, and so on. You will have z raise power n minus 1, n c n. I can write similarly, for z square. I think by this time, you must be understanding. You can write any time by this close observation. I think, it has to be z square.

Student: z.

It has to be z because, here I think multiplied. I am sorry for that; and so on. Then, the last term, which is going to be there, because z power n; last row below is not going to be out there; only one term. So, you can now, combine them. You take only this term. You take the term corresponding to n c 2, corresponding to n c n, and you will end up in getting nothing but the first one will have only one; there is no term. So, there is only one term, which is 1. Summation of that, because in every series, first time series is only with one term; second time series is 1 and z; third time series 1, z, z square; 1, z, z square, z 4. So, last series will contain 1, z, z square to z raise power n. This series (( )) put g p and then put the things. This will give you the complete thing. So, I will just write the final thing.

(Refer Slide Time: 09:09)



$$= \frac{1}{p}\left(\frac{(z-1)}{(z-1)}\, {}^{N}C_{1}\left(\frac{p}{N}\right)^{1}\left(1-\frac{p}{N}\right)^{N-1} + \frac{z^{2}-1}{z-1}\, {}^{N}C_{2}\left(\frac{p}{N}\right)^{2}\left(1-\frac{p}{N}\right)^{N-2} + \right.$$

$$\dots + \frac{z^{i}-1}{z-1}\, {}^{N}C_{i}\left(\frac{p}{N}\right)^{i}\left(1-\frac{p}{N}\right)^{N-i} + \dots + \frac{z^{N}-1}{z-1}\, {}^{N}C_{N}\left(\frac{p}{N}\right)^{N}($$

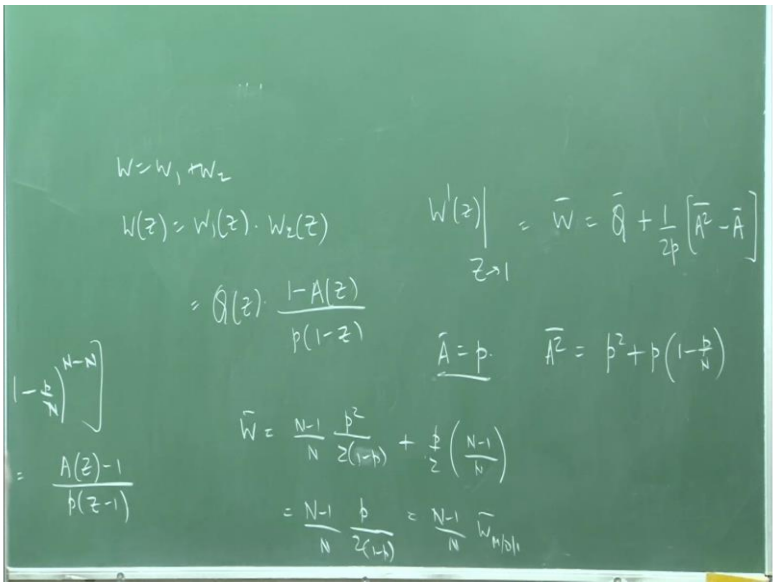
$$\frac{1}{p(z-1)}\left[\sum_{i=1}^{N} z^{i} a_{i} - \sum_{i=1}^{N} a_{i}\right] = \frac{1}{p(z-1)}\left[\sum_{i=0}^{N} z^{i} a_{i} - 1\right]$$

So, you will have 1 over p will be out; z minus 1 by z minus 1.

Student: Sir, the last term will be z to the power n minus 1 (( )).

Of course, you can write down the last term. The last term will be now, n c n p by n, which is 0. That is the way the term actually, prove this. I can actually, take 1 over p z minus 1 out. This now, can be written as summation of from i is equal to 1 to n z over i, a i minus; this minus 1 is there; this, very well can be now, written as 1 over p. Now, this term can be written as 1 minus a 0. So, I can write and this. I can combine this whole term; a 0 can be taken inside, and I can make this. This is nothing but a probability generating function of a i a z. So, expression will be a z minus 1 divided by p z minus 1; that is the root to z. So, I can now combine these two things together. In fact, there would have been another simple way of doing it. You multiply by z and then construct the two terms, whenever you want to find out that kind of series; that is also another simpler way, that z minus 1 would have come out; upper 1 would have been simply, a z minus 1. So, either way, it actually finds.

(Refer Slide Time: 12:56)



$W = W_1 + W_2$

$W(z) = W_1(z) \cdot W_2(z)$

$= Q(z) \cdot \dfrac{1 - A(z)}{p(1-z)}$

$W'(z)\Big|_{z \to 1} = \bar{W} = \bar{Q} + \dfrac{1}{2p}\left[\overline{A^2} - \bar{A}\right]$

$\underline{\bar{A} = p.}$ $\quad \overline{A^2} = p^2 + p\left(1 - \dfrac{p}{N}\right)$

$\left(1 - \dfrac{p}{N}\right)^{N-N}\Big]$

$= \dfrac{A(z) - 1}{p(z-1)}$

$\bar{W} = \dfrac{N-1}{N} \dfrac{p^2}{2(1-p)} + \dfrac{p}{2}\left(\dfrac{N-1}{N}\right)$

$= \dfrac{N-1}{N} \dfrac{p}{2(1-p)} = \dfrac{N-1}{N} \bar{W}_{M/D/1}$

Now, w is equal to w 1 plus w 2, which actually, implied w of z is w 1 of z; this is a standard result again, from (( )) for probability generating functions. So, whatever is the addition of the variables, will become nothing but multiplication in p g f (( )). So, this actually, means we will have Q of z, 1 minus a of z, p 1 minus; I had actually, taken 1 minus z; so, upper 1 is also 1 minus a z. So, that will be the final p g f. You take the derivative of this, and make z is equal to 1; that will give the mean value of w, w bar. So, you take that thing. This, value which you get is nothing but Q bar plus second movement around 0, and first movement around 0. First movement around 0 is mu, and a bar is p, because n into p by n will be p, actually. So, every value will be a bar, and of course, a square bar also, you can estimate.

This should be p plus; these are again, all these can be the steps that p maps. From here, you will get w bar equal to whatever was w 1; w 1 was n minus 1 by n. I think p square 2 p minus 1, 1 minus p i; that is what w; this q bar. There is a second term now, you can put from here. This will turn up to p by 2 n minus 1 by n. You solved for this particular; a square, you have taken from here; this a bar, you take from here; put it here; solve; you should get this. This in turn, will give you this, which is nothing but this is every delay for m d 1 queue. Only factor, which is multiplying factor, is n minus 1 by n. So, technically, all output queued system will actually, work like m d 1 queues, and the beauty of it is, if you want to actually, plot with p with maximum throughputs. So, when n goes to 1, this term goes way. There are p by 2, 1 minus p. So, if you solve for it, this is nothing like saturation. The delay also goes to infinity.

In earlier case, the queue length; they all will go asyntactically, when p is equal to 1. So, same is with the delay also. So, that you can actually, operate with any value of p; does not matter. Only if p is equal to 1, you go to infinity, which anyway, is unstable for any kind of markovian arrival process, queue p is equal to 1; you will always be in unstable. We have seen that for having one infinity queue at some point of time earlier. So, output queue is actually, the best option, if you can achieve that, but you need not be able to implement that all the time. So, what we can do is now, move on to the input queuing system for comparison purpose. Input queuing system; I will never be able to achieve very close to quotient; p actually, increases; throughput is increasing; it will try to reach to basically, as far as possible. Only the queue will build up, but my throughput performance will always, become very close to 1. So, 1 packet per slot, I will able to push at the output port. Only my buffer line requirement will be high, because of the randomness, because of the contentions.

If we somehow can ensure that there is no contention, throughput will be one, but input queue, because there is ahead of the line blocking, we can actually again, do that modeling and find out what is the approximate value. So, that is about 53 something percent or 0.53 something; that is the maximum throughput, which can be achieved, if I am going to have ahead of the line blocking in the input queue.

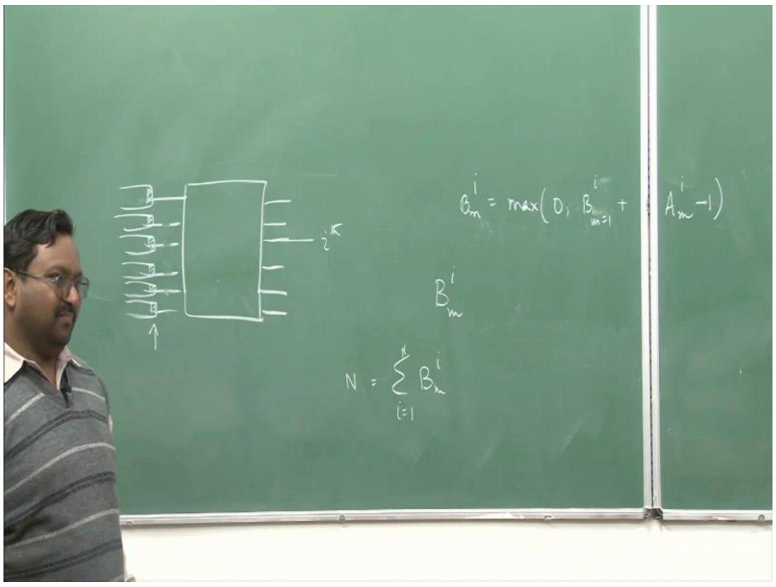
Student: In the traditional n p we are able only conclude technique queue will l e.

M d 1 queue.

Student: With some multiplied

Yes, it is n minus 1 by n; that is the only difference, which is coming because of the batch arrivals. Batch arrivals, technically, like a Markovian process. So, let us come to the input queuing system.

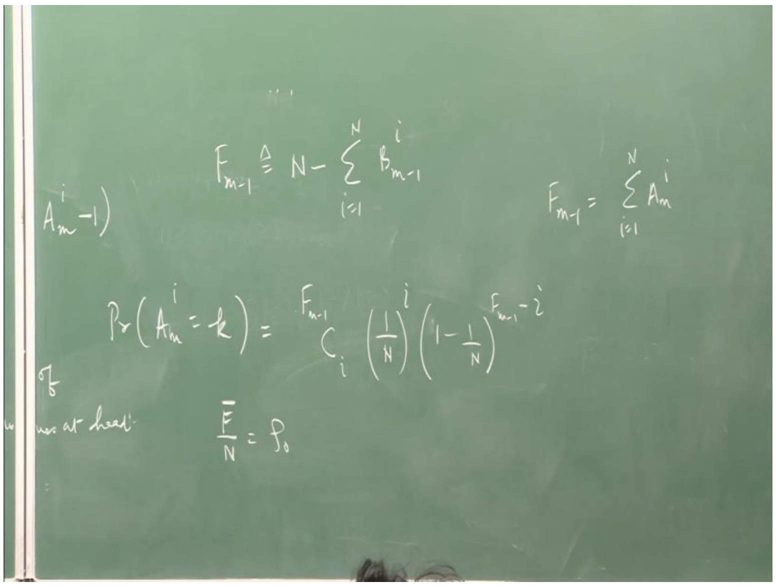
(Refer Slide Time: 19:00)


$B_m^i = \max(0, B_{m-1}^i + A_m^i - 1)$
B_m^i
$N = \sum_{i=1}^{n} B_m^i$

Now, this is going to have many outputs, and there are n inputs. Again, I am not worried about that n minus 1 only will be packets (( )); I am not bothered about that. Now, we have the buffers at the input. At any point of time, I am not looking at for one particular output i. So, whatever is proved for this; I had output is also proved for any other output. Let us say equivalent system; there is no bias. So, I think just take one output, and analyze for that. Now, what will happen is the packets will keep on coming at the head. Once the packet will come, for this, I have to think there can be only look at the head; how many packets are there? So, at the head, just for the packets, which are trying reach to destination i, let those packets be numbered as b m i after n th time slot.

One thing, you should be sure that summation of b m i over all i's, all possible output combinations from 1 to n; this should be equal to how much? This only says that for i, how many packets were queued up in the front of the queue. For i is equal to 2, how many are queued up? Total number of queued up packet, which is their in the head of the queue, has to be equal to n only. This cannot be less or more; in worst case, it is n. If in the every slots, the packets are coming; fully loaded condition. So, this will be equal to n for saturated condition, when there is always, a packet available at the input every time slot. In fact, similar kind of equation, which we have done earlier is also (( )) through. Whatever was there after the earlier time slot, whatever was the number, which has queued up; one of them will be picked up and will be sent to the outgoing port; remaining will remain there at the head. Only thing, now the balancing condition will be defined. There will be new packets, which can arrive in the m th time slot, which I will again, write as a m I, for the i th output port. Now, remember, these will be not out of n. Earlier case, it was out of n, actually; out of n, i packets are coming in the batches now. Whatever are the ports for which, the packet has been pushed out to the outgoing port in the m minus first slot; only on those lines, new packets will be coming in. From those only, I will get some packets, which will be distinct for i. So, I will not use n here; I will use something else, when I will make estimation, and one packet which will be going out.

(Refer Slide Time: 22:21)



$A_m^i - 1)$

$$F_{m-1} \triangleq N - \sum_{i=1}^{N} B_{m-1}^{i}$$

$$F_{m-1} = \sum_{i=1}^{N} A_m^{i}$$

$$Pr(A_m^i = k) = {}^{F_{m-1}}C_i \left(\frac{1}{N}\right)^{i} \left(1 - \frac{1}{N}\right)^{F_{m-1} - i}$$

of

at head

$$\frac{\bar{F}}{N} = p_0$$

So, this summation is very similar. In fact, a very similar result can be used here also, except now, that I will not be using n; I will be using something else here. The probability factor, which was there actually, will get changed; that is the only thing. The probability that your a m, i will be k; I will not use the n; earlier, I were using n hash. These are three ports on which, the packets will arrive, in that few, free head of the line; queues. Front of the queue, which actually, has become free. The next slot of packet only will come and those. This hash m minus 1 is that number; c i with equal probability; I am looking at a saturated condition. So, p is equal to 1. I am trying to find out what is the upper bound on the throughput performance. Under saturated condition, whatever you achieve; you cannot achieve better than that, unless we do some manipulation; we will do that, actually. So, it is 1 by this may s to be there, 1 minus 1 over n. Now, what is this F m minus 1? This is by definition, whatever are the block queues; I sum up all of them for each out going port. This may be, I have taken in worst case, will be equal to n, but ideally speaking, this should be always less than or equal to n. The difference between them is what is F m minus 1? So, n minus; the difference is this; that should be F m minus 1. Then, of course, F m minus 1 should be equal to; that is where, the new packets will come; should also, be equal to this. I have this; new arrivals. These two conditions should always, be satisfied, and I can actually, make an estimate of I can call something called F bar, which is many mean number of three input queues, after every time slot. This is statistical average of this, and rest is the state condition; that led to the F bar. So, you can actually, observe that F bar by n; that which fraction of packets will actually, be going as a throughput; that is the maximum, which you can achieve.

What is happening in that fist switch, in this case, some packets remain; some move; some packets remain; some moves in every time slot. On average, the fraction of packets, which are moving out divided by total lines; that should be the fractional utilization of each out going lane; that is the load; that is the number of packets going out while, no packet is being lost here, remember. It is an infinite buffer. So, this has to be equal to row naught. I am interested in finding out what is row naught, actually. Now, there is an approximation here. This is a very simple judgment; when n goes to infinity, this a i, under study state condition, a m i, will tell to become poison statistics. So, new arrivals, which are coming in, will be nothing but they will be poison with parameter row naught, because that is a utilization, every utilization coming. So, it should be poison statistics with row naught, actually. So, I should call it a steady number of packets for output, which is a i. I am not writing n, because this is a steady state number. This is poison with parameter row naught.

Student: Row naught is output to lay output line.

Yes.

No, I putting packet all the time, but you have to maintain a steady state. So, output utilization has to be equal to input utilization, because these are maximum, which you can achieve under saturated condition; you cannot have input utilization, have done this. If you are going to have it, you queue will blow up; it will not be stable queue. So, stability condition does not come at row naught is equal to 1, which was happening in the earlier case, in output queuing system, when p was utilization of every link. When it is going to 1, then only, you are going to a stability. Here, you will actually, achieve stability, even before that. That is the bad thing about input queues. So, you will blow up to infinity only, even before p actually, goes equal to 1. So, p has to be maintained at lesser than 1 for input queued switch to operate. So, these is a chance that you may actually, lead to instability, because p is equal to 1; most likely, you really, never operate with that p is equal to 1 fully saturated condition. So, what I am saying is under fully saturated condition, your queue is going to blow up, but there is a maximum utilization. So, you should keep input utilization, equal to output utilization; your queue will be almost, kind of stability; just, I could actually, blow up that point, but anything, which is going to blow row naught, is going to remain stable.

Student: That means output is better than…

This always, obviously, common sense says that this has to be actually, better.

Once this is row naught, you look at this expression and this expression. This is very similar to what we did in output plane. I had a very similar expression. This was the poison statistics for n is equal to infinity, and binomial for finite number of n. We could directly, estimate what was the statistics of the output size. It was m d 1 queue thing. So, I can use the same result here, but n is going to infinity. So, n minus 1 by n is 1, actually. So, what ultimately, you will get is the queue length; I do not have to put b i bar. This value will be how much? Now, it is row naught; I am not going to put p; it is row naught. That is actually, equal to p; p is also utilization. Time is row naught here. That is what it will be, and of course, now, I can use this expression. Under steady state condition, m actually, can be removed; this value will be nothing but n by n.

Student: one has to variance in this theorem.

n by n; n by n becomes 1.

So, b i, I know already from else, this 1 by n can be taken out. Now, solve for this upper one. This is nothing but row naught under steady state conditions; 1 minus and you will have 1 over n. So, for each of these component summation, I am taking bar, actually on top of it; both, I am taking expectation on both sides. So, this will be nothing but n number of times (()) b I bar, which is nothing but rho naught something. So, this should be, because I am actually, having n terms, all same values is multiplied by n; this can be cancelled with this; solve for this rho naught. So, you should get 1 minus rho naught is equal to rho naught square; 2 plus rho naught square minus 4 rho naught. There also has to be a 2, fine. So, this rho naught square, I can make equal to 0. This is the equation of what we will get. Solve for it.

Student: 2 minus root 2.

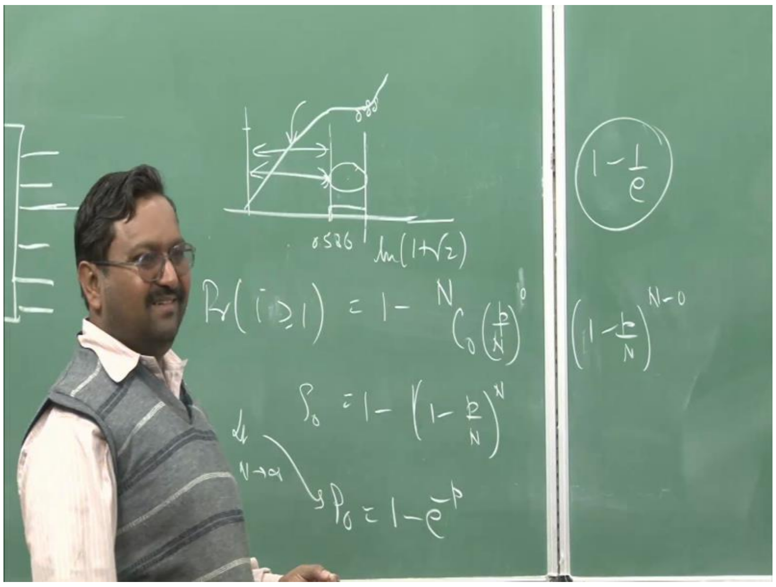
Yes.

So, you will get actually, these two solutions for rho naught. Rho naught can never be greater than 1, obviously. So, 2 plus root 2 cannot be the solution. Only possible solution is 2 minus root 2. So, that is the maximum throughput performance, which you can have out of a input queued switch. It cannot be better than this. This number, I think, turns out to be 0.586; 5836 percent. This actually, means at the input side, number of packets, which are turning on every incoming port on an average per unit slot, has to be less than 0.586 for a stability. Because what is happening is as p changes, the throughput performance rho naught actually, will change this n to saturate, roughly about 0.58. So, throughput is saturating, but my input is high to somewhere. If the input is higher and output is low, what will happen?

Something has to be stored; flow is not balanced. Where it will be stored? In the input queues; the queues will blow up, ultimately, which is not good, actually. We have to always, operate below 0.586. Now, question is can I do better than this; is it possible? I think it is possible to do better than this. So, what we do is; this actually, means they are going to operate, if there is an input queue switch in two possible ways, so that, I can improve my throughput to about 80 percent, actually. Instead of limiting myself to 0.586, 0.88 something, I think, we should be able to achieve. I cannot, if you go to 1. So, what we have to do is very simple root. So far, my p is less than 0.58 here, inside, keep on operating here. Suppose, what I do is instead of this, whatever packets I can transmit; I transmit; if I cannot transmit, I do not keep them. I do not keep head of the line blocking. I just simply, drop those packets. Let the new packets come in, and that will improve the throughput performance. Remember, there are now, two phenomenas, which are happening here. One is the packet, which are arriving at the input. When you are actually, working with lower p, lower arrival rate then less number of packets are coming. So, what happens is even, if you keep head of line blocking operational; does not matter, because packets most likely, would not come and you should be able to transmit. So, you achieve a higher throughput performance. After certain threshold, if you do not drop the head of the line packets, that blocking itself now, will cause you the loss, because sufficient packets are anywhere, arriving, so that, you can achieve higher throughput performance. So, from there, this actually, idea came. So, if you start dropping the packets, how will you estimate the throughput? Every time, there is a fresh slot coming in front of the queue. So, what is there, going to be performance? What is going to be the throughput performance on the i th output?

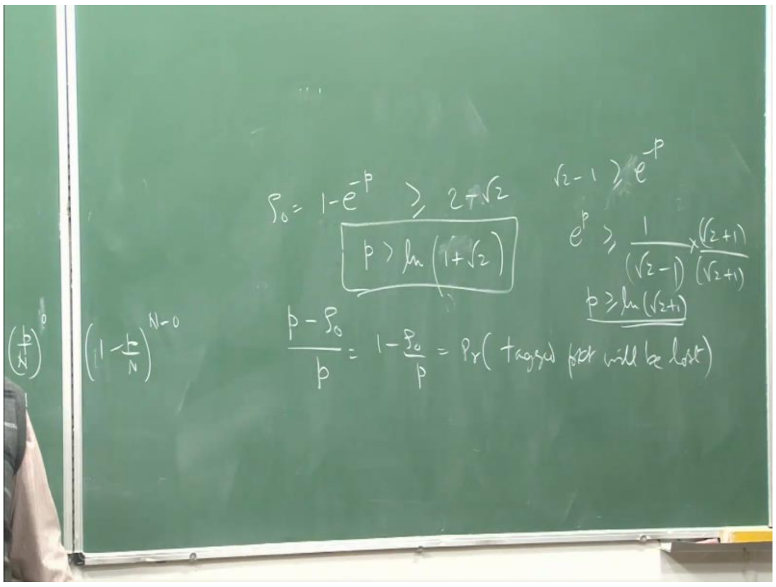
So far, even if I get what is the probability that I will get one or more than one packet, destined for i out of this m; that will be the throughput performance. Is it that is a probability that you will have a packet in the outgoing slot? If you one or more than one being directly destined for this, if it is more than one, then only one will be going; if it is less than one or it is zero, then nothing will be going. So, 1 and 2, 3, 4, 5 for all them; only one packet will be going.

(Refer Slide Time: 37:17)



So, you just estimate the probability that i will be greater than or equal to 1. That is nothing but your throughput performance rho naught, and this probability is 1 minus n c 0 p by n. This is nothing but 1 minus. So, that will be rho naught or the throughput performance. Then, when n goes to infinity, then what will happen? This rho naught will become 1 minus e raise power minus p.

(Refer Slide Time: 38:32)



$$\rho_0 = 1 - e^{-p} \geq 2 - \sqrt{2}$$

$$\boxed{p > \ln(1+\sqrt{2})}$$

$$\sqrt{2} - 1 \geq e^{-p}$$

$$e^{p} \geq \frac{1}{(\sqrt{2}-1)} \times \frac{(\sqrt{2}+1)}{(\sqrt{2}+1)}$$

$$\underline{p \geq \ln(\sqrt{2}+1)}$$

$$\left(\frac{p}{N}\right)^{0} \quad \left(1 - \frac{p}{N}\right)^{N-0}$$

$$\frac{p - \rho_0}{p} = 1 - \frac{\rho_0}{p} = \Pr(\text{tagged pkt will be lost})$$

Now, how many packets are being lost per line? Because incoming rate is p; p number of packets are coming per line per unit slot, while your outgoing is rho naught. So, how many packets are being lost? p minus rho naught packets are lost per line per packet, sorry, this is per line, there whatever is being lost.

Student: Per line long that.

Per line per slot; these many packets are lost. So, p are arriving; rho naught is going out; p minus rho naught are going to be lost. If you know how many packets are coming, you divided by that; this should your probability that a packet, a tagged packet is going to be lost. So, this will become that probability. So, this is the probability that a tagged packet will be lost, Then of course, now your rho naught is 1 minus e raise minus p. For a given p, this is what is the value in this case. When this is going to be greater than or equal to whatever, I have solved for fully saturated condition; 2 minus root 2; find out the values of p for which, this is going to happen. Till 0.586, if my p is till that point; it is fine. After that, I want to actually, have higher throughput. So, what is the value of p? From here, you should be able to get a value of p, which is equal to. So, this is the cross over point, after which, you should start dropping the packets.

Student: This is minus one, two.

You are correct; one. Which one?

Student: This (( )).

You solved for it. Actually, it will; let me solve for this (( )). I will write these two equations; this will turn out to be root 2 minus 1; root 2, you put on that side.

Student: Greater than equal to e power minus.

This 2 minus 1 is 1; 1 minus root 2 minus e, this power minus p. So, root 2 minus 1, I will change the sign. That is the way to come, and then of course, this is nothing but you multiply by root 2 plus 1; divide by root 2 plus 1. This will turn out to be 1, actually; 2 minus 1 is 1, and then you take the log. So, p has to be greater than equal to natural log of root 2 plus 1. So, the moment you cross over this particular value, you have to estimate what is this value. You will find that; this is remember, this throughput is dependent on p. At certain probability, this will be better than the saturation throughput. At that particular point of p, you should switch over, and then when you put p equal to 1 here, that is the maximum, which you can achieve; we cannot achieve anything, better than that. You put p is equal to 1. So, 1 minus 1 over e, which is roughly, I think 0.882.

Student: Recently, we are applying hybrid approach?

No, what is this value, it should come? I think this is 0.882, when you start dropping the packets, but you cannot actually, operate. You have start dropping the packets, when it crosses 0.586, and that is the maximum, you can achieve. Anyway, when you are dropping your packets, your buffer will have to be refilled up; everything, which is coming afresh all the time. So, you have to start dropping the packet on tail side of that, also. When you are dropping, there is only one buffer required for input. So, this actually, means you can actually, operate input queue switch, but you have an operating reason. So, whenever your input arrival rate is less than 0.586, it is fine. The moment create 0.586 is going to explode, it is better to switch over to dropping the packets, and it will keep on operating. In this zone, you will find out; you are dropping this thing actually, you will always, give a better performance for dropping the packet situation. So, this is actually, another; there are two thresholds because of this. One is because of stability reason, I have switch over. So, 0.586, whenever it is there; till this point, I will be operating with ordinary strategy; head of the line blocking will be happening. At this point actually, this stability happens. So, I have to switch over to dropping the situation, but I may not be still, getting or only getting 0.586 performances; it will not be better. It will only become better when I make cross over natural log of 1 plus root 2. Somebody, either, calculate to find out the value. I think it is 0.89; that is it should not be 0.8; it should be the same value.

Student: 0.88, sir.

0.88? Yes. So, this is 0.88 till this. After this point, you will start getting the (( )). Below this, you are operating below without saturation begins. We are operating in saturation within here where, you will only operate a throughput of 0.586, and after that, you will get, you start dropping. Even if you have dropped or does not drop; does not matter, because even, if you are dropping, your performance will be poorer than 0.586. So, queue will remain unstable; that is the only problem in this zone. So, what you do is at the input buffer, you limit them on length of the queue. If the queues get fully filled up, you start dropping the packet; that is only way, because you do not have infinite buffer. You require an infinite buffer for this operating (( )). Usually, buffers will always be finite. Once they fill up, you start dropping; you do not actually, store any incoming new packet; unless, head of the line gets cleared.

At this point, you start dropping from the head of the line, because then I have the throughput performance will be better than 0.586, because if you drop from head of the line here, you would not get better performance; your rho naught will be lower than 0.586. At least, you are operating at 0.586, which is the maximum possible. So, at this point, you start becoming better. So, I think it should be something like this, that of the throughput, and then here it will become better. Then, you reach to 1 over, because that is what you get the maximum; you cannot get anything better than this. What is this value 1 over 1 minus e?

Student: 0.38.

No, it should be higher.

Student: 0.63.

0.63. How come this will be this? This is rho naught; that is p equal to 1. So, it is 0.586, 0.63; whatever, we will be reaching here, and you will saturate with that, actually. So, 0.586 to point whatever, 0.63 into this value; that is where, you will saturate the time. You cannot get anything better than this; it is not possible, when p is equal to 1. So, that is the maximum in the input.

Student: Sir, utilization is arrival rate by departure rate or departure rate by arrival rate?

Utilization; no, it is not the ratio; it is you look at the output line; for how much fraction of the time, the packet is there. If it is a slotted system, in every slot what is the probability of packet is there. So, if you observe for 100 hours, whatever is your utilization, into 100 hours; those many time, you will see that packet; for remaining time, you would not see the packet. Utilization is how much you are utilizing the link.

Student: Generally, in queue system we will be taking lambda by mu lambda is the arrival rate and mu is the departure.

No, that is the load; that is not utilization; that is the loan. Lambda by mu is the load factor. That has nothing to do with the utilization. You can compute utilization in those terms; that is fine. Usually, that will be known as load. Load is also, technically, equivalent to utilization. Under steady state conditions, that also, gets you the fraction part of the time for which, a link will remain utilized; link will be having some time.

Student: Sir, here, we are dropping the packets factor a queue.

So, incoming utilization is higher than the outgoing utilization.

Student: Is there any relation between the next packet, which is coming in the same thing?

All are independent; there is no correlation. Marcovian arrival process actually, means every packet is coming, independently; I am only assuming parameter called probability p; in a slot, a packet is there or not there; next slot, if a packet is there or not there, is independent.

Student: And how these lost packets can recover?

We do not recover; they are gone; that is the responsibility of higher layer. This is what is technically, known as congestion. Your switch has the capacity, but there is a congestion building up, because too many people are trying to go through the same line, and you have to be dropped, because it cannot go through. Congestion cannot be cured in the network in any switching load, or any link. Only thing is that, you can start reducing your traffic, which is being pumped in. You go on to the road, for example, go to the city; try to drive. There is congestion; what you have to do? Do you get a solution? No. Best is, you can start improving utilization or some discipline; you can still improve the performance, but if it is still there is congestion, you will leave with that, or best is do not go. Have a radio broadcast, telling everybody, there is congestion on this road; please, do not drive with this. So, incoming rate itself, you are trying to reduce, and that will then keep the congestion within limit.

That also what we do in networks where, in TCP algorithm, for example, t c p (()) algorithm, which technically, does this. It guesses when the conjunction builds up; it will start reducing its transmit window. So, it will pump less number of packets per unit, but everybody should do it. It should not be that these four honest guys were doing this, and these guys say now, (()) let me push this. So, this kind of free riding can actually, happen. This happens between UDP and TCP, for example. UDP does not have congestion control. Last semester, I have taught this. So, if you are transmitting say, audio, video transmissions typically, will be on the UDP. When you are doing a file transfer, network conjunction happens; your file transfer will slow down, actually, because of congestion control algorithm. UDP; there is no congestion control; it will see a very good benefit to do actually, even pump more.

You can actually, conduct this experiment. Try downloading heavy file of few GB and also, try to play video. Make sure it is an UDP transport; this one is TCP, and let the link be congested. You will find your video actually, still perform better; not an issue. What will suffer is always, that TCP traffic. Well, (( )) observation here, we are still looking for better algorithm for doing UDP based congestion control.

Student: Sir, excuse me. In that figure, what is the axis are corresponding to 0.586?

This one; this one is p.

Student: On the y axis, what is the y axis?

Y axis is the utilization rho.

Student: No sir, the value corresponding to that 0.586 of y axis?

This one? That is the way you will be increasing, and there will be some formulation based on that here is that stuff.

Student: Sir, what is that rho naught corresponding?

Rho naught is output code utilization.

Student: Corresponding to 0.586, the value sir?

Next time, you can get is only 0.586 is this thing. Actually, if this will be lower, this is not 0.586, but you will saturate at 0.586; this much I know. This 586 might be happening, somewhere here. So, your saturation is actually, not happening before it. This is not the mean point; it is somewhere, here having point size. I have not done the exact calculation, but see by intuition. Important thing is by intuition, by understanding, I should able to predict; I should be able to generate a hypothesis. So, science is all about that; gaining insight and based on that, predicting something.