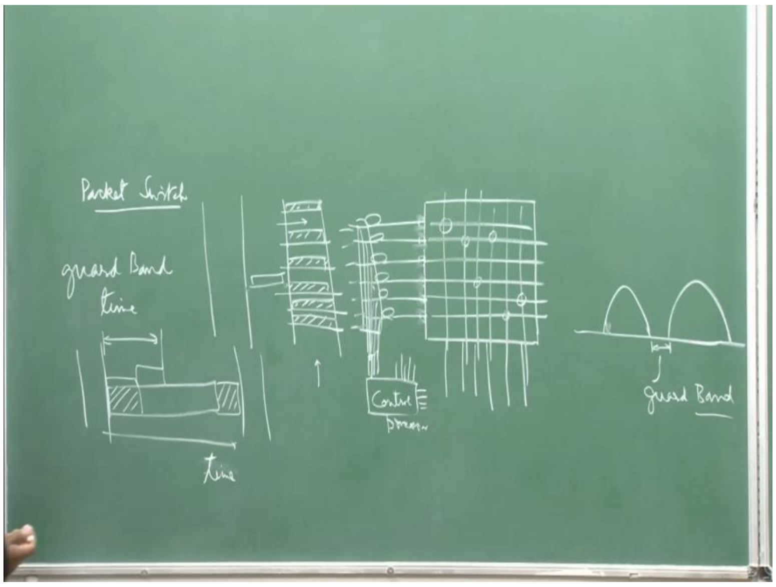
Digital Switching

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

Lecture - 17

(Refer Slide Time: 00:18)



So, the next topic is about the packet switching system. So, this I will continue from wherever I left last time. So, we want to implement a packet switch and one of the possible options which I taught of is going to have a, we will be going to have a cross bar. Cross bar will have certain number of inputs and plus certain number of outputs. Usually there will be same in number and we are going to assume that packets are coming on each one of these lines, and they are all synchronized in time. These are the packets which have arrived. On each one of this, there may be situations that there is no packet on certain index is also possible.

So, what happens in the first slot is it will come and then second slot, third slot that is the input which is coming inside the switch. We have an interface card setting on each of these ports. These interface card will read the packet, at least the packet header. Usually every packet will have a header as I have told earlier. So, this is when this excesses. This time slot will always be received first. There will be a possibility of tailor, but we are not sure about the tailor part. This is not a store and forward. So, we will not do c r c checking in this case. So, there are errors happens. This may be misrouted. There is a possibility of even doing a c r c checking, but then old packet has to be received.

First here c r c has to be verified and then only the routing will happen where is going to be one slot delay which will happen it interface port in that case. So, usually delay will be for reading of this, only header part, and then there is some processing time, worst case processing time. This much delay will be required. So, we have to introduce that delay here I am showing it by loop. This can be a kind of memory which you can put with the, you are going to have a still better time granularity or it can be delay line. It is going to be fiber optic system. It is going to usually a delay line.

Now, all these interface information, this will be always after the interface card once header has been analyzed and is been dead out. So, these all through interface thing delay part. That is why the packet will come. Now, this has to all come to the central control processor. This control processor will then decide after doing an appropriate computation that which cross points have to be activated. So, these are the cross points. Cross points will be chosen like this. So, this is going to now do a vertical row and column control thing for this. They will be activating just before the slot arrive. So, usually there will be guard band time between two packets.

So, packet actually is not going to fill up the complete slot, it is going to be only part of it because you repair a guard band time for the activation of cross point and then another time guard band time for again what we call revising of the cross point and setting of a new cross point guard band. We call it a g u because this will be temporarily space. If it is not in frequency space in f d m for example, if you have a spectrums of various, if you consider n multiplexed signals, you always keep some gap, so that there is no cross top. This is also known as guard band. In fact, this comes because this is frequency bents. So, guard band look like, but it in time there is nothing like a band. Actually that is a guard time, but you can also call it guard band time. Both are fine.

So, you require a gap between that has to be kept when you are designing the system, but usually whenever you do analysis, we assume all these operations can be done in almost infinite space. So, you need not to actually have a guard band on paper. It is fine critically, but in real life you need to have a technique that will depend on your circuitry time which is required here. What is going to be tolerance is in activating and deactivating the cross points, but you remember the activation also means you have to first of all link the row and then column and the row has to go down and column has to kept up top for one slot period. When the cross point is activating, column has to go down.

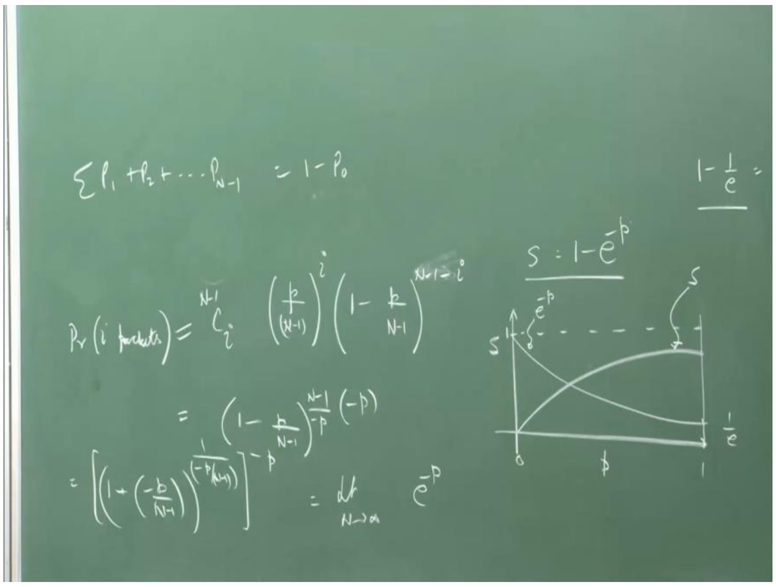
So, these thus take time. They cannot happen continuously. So, as for this, now this packet will be passed on to the output and this scheme is going to work fine. So far each packet is going to have a separate destination. No, two packets will have the same destination. If these conditions somehow can be satisfied, then this is going to work fine and this is what is known as basic simple switch actually, but in real life, the switch will not work this way. There is always the possibility because the packets which are arriving are coming independently at each input port. So, there is a possibility that you will have more than one packet coming in, and they would not go to the same destination in that situation.

What has to be done? So, one possibility is that in this kind of scenario basically what will happen is, if these two wants to go to for example, this for remaining of them unique combinations are actually ok for all of them. They will not have any anybody. They are going through only these 2 1 to go ahead, but I cannot pass. I cannot keep these 2 1. I can actually have multiple cross point active, but not in the column. So, when you have multiple cross point active, this way it is multi-costing. So, since this cannot be done, what you will do is only one packet will be allowed to go through. Second one will be actually in this case has to be sent to the blank one or you do not activate that cross point actually. So, second one will be read out, but it will not be going anywhere. Remaining one, this line it will die out actually. It cannot pass through the switch. It is lost. So, that could be one strategy. So, we will actually essentially make an estimate of the throw put maximum, throw put which can be achieved in such a scenario.

How do you choose the packet? You have to decide an algorithm. If you say I am going to give one higher packet into a lower packet, all you have to choose the one. First drop the two. If all ports are having equal priority, then choose any thing randomly. Let it go. If packets are lost in the higher layer which will take care of it, they have to keep trying. If the packet is lost, I have to do a retransmission. There will be some data link control or flow control mechanism will be setting in there. Window flow control will be there which will take care of the retransmission. So, that is one design via the packet thing at lost.

Now, there can be an alternative strategy we can say. Now, before actually going to the alternative strategy, I can just quickly make an estimate of what we will be a throw put of this particular thing. So, a packet is going to be there on a line is be probability p. So, that is the loading factor basically utilizing factor of every line, every input line, it will have a packet with probability p and then there are n lines.

(Refer Slide Time: 09:33)



$$\Sigma P_1 + P_2 + \dots P_{N-1} = 1 - P_0$$

$$1 - \frac{1}{e} =$$

$$Pr(i \text{ packets}) = {}^{N-1}C_i \left(\frac{p}{(N-1)}\right)^i \left(1 - \frac{p}{N-1}\right)^{N-1-i}$$

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

$$= \left[\left(1 + \left(\frac{-p}{N-1}\right)\right)^{\frac{1}{(-p/(N-1))}}\right]^{-p} = \lim_{N \to \infty} e^{-p}$$

$$S = 1 - e^{-p}$$

S
e^{-p}
1
S
0
p
1
$\frac{1}{e}$

So, n by n switch I am assuming. So, this can go to any one of these packets with any one of these lines with equal probability. Usually what will happen, again there is an input 1 and output 1 is connected to the same node. This is not actually line, but actual links are bi-directioning. So, it will not be choosing. At least output 1, it will always be choosing out of the remaining n minus 1. So, you will have with the rest of n minus 1 probability, the packet will be using choosing any one of these which actually implies that.

Now, if you look at n output if you give me assuming that the first element, how to put in the P is the probability. That packet is there and then with equal probability, it is going to be direct to have any of the outgoing ports except the outgoing port corresponding to itself. Each line, each input line is always paired with an output line because both pair will be connected to node or another switch always you do not send the packet to yourself actually in the loop. That is not possible that I am actually precluding from possibility. So, take an output. So, how many packets will be directed to probability?

That one packet will be directed. I have an estimate that from here you can get the simple commendatory. It is actually from there you can get it. So, probability that a packet from a line will come, this is given by p divide by n minus 1. It can receive only packets from all other lines except the line which input line which corresponds to it. This is the same situation discussions I gave that if one is talking to two and always talking to one is, yes we will never send a packet to yourself, but in the other stretches will discuss that cross bar when you discussing.

We will never do a looping. So, there I am sending a packet to you sending it to. That is the different situation. You are sending a packet that may through a different route altogether. That means, if packet is going independently, but when I send a packet to such, it should not be sent to an outgoing input which is again back connected to me. So, my packet itself will come back to me in that scenario your voice was not coming back to you because other persons voice which have coming to you. So, this actually means that packets will be coming to this particular node in one single slot which will be directed.

What is the probability of that actually? So, that probability will be p divided by n minus 1. So, from out of the n minus 1, possible input of them will be directed to me. This is with this probability that the remaining one of them will not be directed to me. This probability n minus 1 minus I and this one happened in combinations of n minus 1 c i. So, that will be this probability of receiving i packets. So, even if you receive anything which is when you will receive no packet in a slot when i is equal to 0, if you receive i is equal to 1 or I is equal to 2, I is equal to 3, you will always get only 1 packet. Remaining will be dropped.

So, if I sum up all the probabilities which go on p 1, one packet receive to till p n minus 1. I will always get one packet. This is the probability of achieving getting at least 1 packet at the outgoing port. So, outgoing line will have in a time slot with this probability 1 packet. There that will be the throw put actually and this is nothing but 1 minus p 0. So, what is p 0? From here we can estimate this will be n minus 1 c 0. This will be nothing but I can remove this. This is 0. So, this will be 1. So, this 1 minus p divided by n minus 1 n minus 1 under full load condition actually in every slot, there is a packet. The packets are coming all the time p is equal to 1 and then you will make an estimate from there.

So, that will be next input. So, I can divide this by p minus p n minus p and this will be nothing but in this when limit when n goes to 0, infinity, this will become e raise power minus p. So, I want n minus p 0 to be estimated. So, 1 e raise power minus p is what is going to be the throw put. Yes, per line throw put per line will be this much. So, this will get a maximum value. When p is maximum that is negative term actually remember. So, I have to minimize on that since the exponent is negative. So, I have to maximize over p. So, p gets a maximum value which is from here actually you can float how the throw put increases with p, and this will achieve, we can slot this s. This parallel throw put per line s p goes from 0 to 1. So, e raise power minus p is this. So, this value is 1 over e. So, this is what will be curve for us.

So, when p is 1, initially throw put will increase, probability increases, but then when p goes 1, it means full load condition at the input put that value, it is 1 over 1 by e which is going to be I think 0.532. I think value is 0.532 or something, but this is what we get as maximum. So far this policy is fine, but I need to improve because there is a maximum throw put limit, and I will also have a loss. Even if p equal to 1, roughly about more than 40 percent of packet will actually be lost. This is not good which inductance ideally if I actually have even under high load conditions, low load conditions, it is fine, but on to high load conditions, I ideally would like to have something which will achieve almost 95 percent or 96 percent probability and as less number of packets should be dropped as possible. Then, of course we will be interested in what will be delay in the packet being transmitted out.

So, now, let us look at this strategy for buffering. What we did is, we were trying to build up cross bar switch I have taken because that is what we know about in order to build up a packet switch with this, we have taken at the input and we assume fix slots, false slots at all the inputs as synchronous. I am actually neglecting all the delay everything and I assumed that hatred can be process control. Processor can process almost certain speed and can apply the control action in real life. There will be some amount of delay after this header has been detected. There will be some guard band which has to be provided between two packets, two consecutive packets in two consecutive slots. So, I write this action of setting of everything can actually take place. Packet will only be transmitted when the cross point is connected, and not when it is under the process of connection.

So, based on this various cross points are said and you can get the output. That was the first thing which we did to kill. I thought to kill let me do a very simple analysis. There is no buffering here. I could just keep on coming and whatever comes in the slot, we just transmit. If there is more than one packet trying to go to one single output, we have no option, but to allow only one and rest of everything will be dropped. I am not worried about. Actually randomly any packet will be selected and going out, but if there can be priority disciplines where you can say 1 and 2 are going to content packet. From 1 will be priority and it will go out. Packet from 2 will not go out because remember I have not still estimated the probability that your packet will be lost. I am only worried about the throw put part. Throw put is how many packets per unit slots are going out from an outgoing port. So, whatever it is true for one port has to be true for any other port under that assumption. Then, we estimated what is going to the probability of having a packet here from an incoming line.

So, the packet probability p, the packets will be there in this slot. This will be directed to remaining n minus 1 with probability 1 over n minus 1 equally likely chance uniform routing. There is one doubt sir. The probability of p should be 1 by n probability because equal probability n lines are there. So, it can be 1 by n or p 1 p by n. These p's are arrival probability having a packet on one single line. Each line is independent. If you want all the line is to be in packet, that probability will be p raise power n. Only one packet one line as the packet that probability is p. All other lines do not have 1 minus p raise power n minus 1. So, in general, packets will be present here that probability will be given by p raise power n because they can be in any order, any combinatorial.

So, it is n c i and 1 minus p n minus i. That is the probability that i packets will be there in a slot at all the inputs of a switch. So, p is the probability of having a packet on one line. The same probability exists in second line also and third line also. The case which you are talking, it will be 1 by n. There is only one packet which is going to come. It has gone to any one of these lines. Then, it would become 1 by n, but we are not taking all inputs are independent, they are not dependent on each other. So, once we get a probability that this guy will get a packet from 1 line, say this one. So, that probability p divided by n minus 1, the input which is corresponding to this output port, I am not bothered about that anyway. When I take limit, n goes to infinity that n minus 1 does not matter. You can actually observe that I have used that thing.

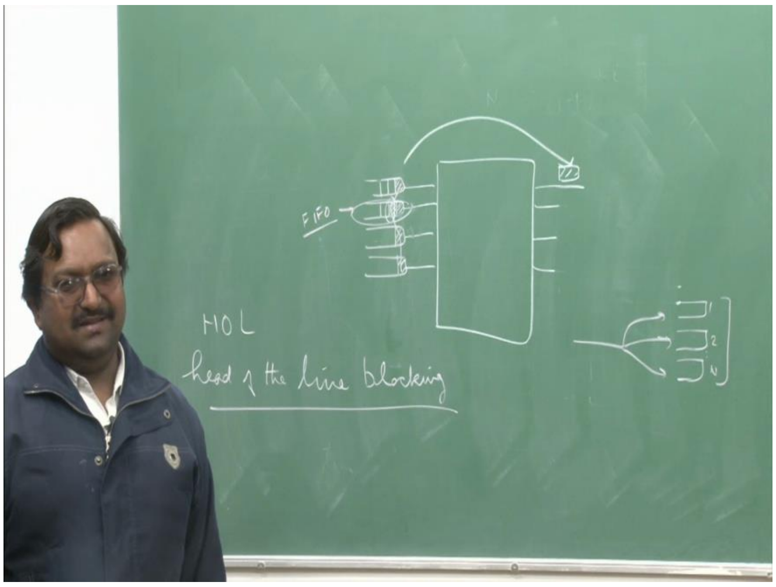
So, getting i packets from various combinations directed to only this port, this probability as to be that i packets have been directed out of n minus 1 which could have come to this port on which is coming at the corresponding input port corresponding to this outgoing port. It will never come to this. So, I have excluded. So, n minus 1 c i. This is the probability and it is being directed here i of them remaining. What will happen, they have not come to it. So, it has to be 1 minus p by n minus 1. The remaining is that simple binomial thing and then of course we say let us find out what is the probability that no packet will come to move because either you find out each of them is sum of all these, or you do 1 minus of p 0. Both are actually same.

So, we have found out p 0. So, when you put i equal to 0, this will become 1. This term goes away. 1 minus p by n minus 1 will become this. I multiplied and divided by in the exponent y minus p. When I take a limit, n goes to infinity or n minus 1 goes to infinity. This will become nothing but exponential that becomes exponential; sorry this limit actually should be put here. You will get e raise power minus p. So, what is the probability you will get? At least one packet remaining will be dropped. I am not bothered about, but I will have one packet in that slot. So, if I have 1000 packets, how many packets will come? So, this will be 1 minus p 0 multiplied by 1000. Those many packets per unit slot will be going out.

So, I have put that we will leave it becomes 1 minus e raise power minus p. Of course, e raise power minus p will be shown by line which is decaying and they are growing line is what will be our throw put 1 minus of that. So, as p increases, your throw put will increase, but it will saturate. It can never be higher than this. When p is 1 which is 1 over 1 minus e 1 minus 1 over e, I think at earlier 0.58 or something, I must be on that value I think. So, you can compute that and then verify. So, there is quite large number of packets will not pass through, even p is 1. P is 1 is saturation case when every input is going to have a new incoming packet in every time slot, there is what p is equal to 1 packet is under full load, and the complete saturation. You will saturate to this 40 percent of leave packets will be lost. So, you will be happy, but we are going to have a very high loss rate. I have to minimize on this loss rate. That is the question now how to do it.

So, we have to solve this particular problem. So, I have now moved from simple cross bar which was a circuit switch element. That is way we have learnt it, and we have converted into a packet switch, but my throw put performance is limit. I have to now move one step ahead. What are they improved the performance? Now, how we will improve the performance? So, what we can do? We can do buffering. Some kind of buffering can be done. So, let us see if we can do the buffering, but we have to do the buffering.

(Refer Slide Time: 25:26)



My question is this. Some people can say what I can do is I have this cross bar, non-cross bar outputs. I am framing this side and this side, not in the bottom. So, where I should put the buffer? So, there are four lines. For example, this 4 by 4 is where I should put the buffer. I cannot insert a buffer inside a cross bar is. Of course cross bar is a cross point in every slot. I just snap certain cross points and this snap cross point sets will keep on changing in every time slot. So, where to put the buffer? Can I put at the output? No, we cannot do it. Why? Which leading more than one forget come out and the input there will be lost because there will be only one packet coming at output, and that anyway we will have a chance to transmit.

So, this buffer will always remain empty. Packet comes and goes out. Packets have been lost here because the contention is happening here. So, common sense will tell I think I can have the buffering here. I need to have the buffer at the output, but once you do this, now if those packets are coming and there have been read out synchronously from the buffer. Any packet is of fixed length. Again that is our assumption here. Now, there are two packets at the head of the line. This only is the line. This is head of the line is the first position in the queue, and they both want to go to same outgoing port. So, what I will do is, I will just read one of the packets and this packet will go out. This one packet will not read. This will remain in the buffer. These packets will be read because they are not contended. So, they will be cleared, but in the next time slot, this is empty.

So, new packet will come. The new packet will be coming here next time slot. Again if there is a contention between this packet and this, there can be many over things which can be happening. So, contention will always be looked into only in these front packets. The back packets we are not bothered as long. So, again if there is a contention, I will choose the maximum number of packets which can be transmitted and then of course once you decide on that maximum number of packets which need to be transmitted, some packets which cannot go through because there is a contention, they will again de-buffered. So, this buffer can keep on filling; keep on going out actually depending on this.

Now, there is a problem. Actually in this case, there can be scenario. If there is a variation on this, now seems I can have a case like this. If I look at the front of the packets, this is the current situation. For example, this is the current situation. There are two packet buffered here, one here, one here, one here and if I allow only the packets which are hidden, there might be contentions between these two. If there is a contention, I may allow this one. So, there are two packets in the buffer, but if I take this combination, this one, there is no contention. All four of them can be transmitted simultaneously because there is no contention among that pair, but since I am using first in first out display here, all the queues I will not be able to transmit this packet, even when this could have been adjusted. Actually this would have improved the throw put.

Now, this kind of blocking, this is blocking not because of switch. This block is happening because of the queue. Somebody is there ahead of you sitting and he does not want to go through in this particular slot. You could have gone, but you were not actually allowed to go. This is known as head of the line blocking HOL. So, this is the one variant of buffering. This is certainly improved, but this is head of the line blocking problem actually. So, next variation of this is you are going to have a situation where this buffers are not managed a single queue. They can always be managed as a hybrid multi-queue. Actually for each input port I can have 1, 2, 3, depending on because of their inputs. This is known as a kind of different structures. So, packets which are coming in are buffered into multiple buffers corresponding to each output.

So, this one at each input port, you will have this particular thing scenario. So, there is no head of the line, there is no first in first out. For each output, you maintain a separate queue. So, earlier you were having n queue and now, each queue consists of n queue. So, you require n square queue. That is only thing n into n minus 1 queue technically will be required. If this guy will be maintaining a queue for all outgoing ports except the output port which corresponds to itself, so n minus 1 queue is there, n minus 1 here n minus 1 here, n minus 1 here. So, n into n minus 1 queue will be maintained. Advantage is that when you look at the slot, you will now choose the combinations in such a way.

So, because for example, there was a contention here, but I can then look at the queue number 2 and there is no contention. I can always find out a combination, such that the maximum number of packets which can go to the outgoing port. That is the maximum match problem. Actually if you do not do maximum match, you can actually you might have less number of packets possible actually by just choosing appropriate combinations. You choose any one of these flow. One of these n minus 1 has to be chosen in each input port. We choose such that the maximum number of mapping can happen to the outgoing ports.

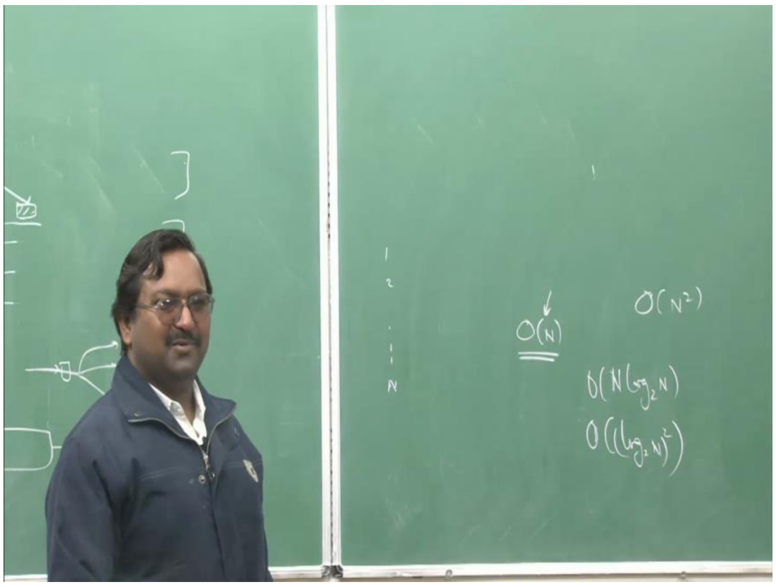
So, every slot you are maximizing one by throw put. So, once you do that, then you will achieve the maximum throw put. Actually on head of the line blocking can be taken care of, but we have to (( )) on implement multiple queues and a separate algorithm as to be implemented inside switch see here whenever at one port. So, it is directly to one of the outgoing ports if you go in that particular. Sir, then who decides in which queue will be a priority is the problem here. See question is you do not know that when there is no complex. For example, which one is of the natural thing? Technically we should always go for first in first out. So, this kind of queue actually is not required. Your better strategy will be only thing is trying to complemented to manage, but this way if you manage, it is simpler, but how many interface ports will have to this switch. It is only one interfacing. One packet will come and it will be arranged to one of the ports. So, single buffer I am now stating into n buffers. So, one queue I am stating into multiple queues.

In this multiple buffers are able to use round, then actually it is. There is a complication. I should actually always, the delay is the issue. One of the important thing is not only I should maximize output, I should also minimize on the delay of the transfer. The problem is you get a packet for this particular thing. It went to packet number 2. Then, the next one of the packet number 1 has come. Now, only for packet on the, only for output 1, the packets kept on coming in and if by our strategy where always choosing, one of this guy will keep on waiting indefinitely. First in first out, this will not be maintained if this same problem can remain in first in first out.

First in first out, you are going to have queue 2, this line at this slot of the line blocking. The second first stated is first in first line. How did it come? Then, second line there will be continued again. The first line now we give the priority to first line, not second line. No, you choose randomly between them. You can randomly or first line has been given. The second time, you can start from the second one and third time, you can start or you can do around them, otherwise there will be because they are also maintained the queue. If the first and second was contending, first one was given last time. Last was the priority. This slot if there was another contention, I will start from 2 for getting the priority.

So, next time I will start from 3. So, I will keep and do in round problem. That is also possible. There are various possibilities in the switch is not that very simple. There is only one possible solution. How is splitting? The multiple routed single routed is you calculating. No I am saying the (( )) input. They are at each input port. There are n buffers when the packet arrives, there is one packet arriving in a slot depending on the outgoing port. I will put it in one of these things for output port.

(Refer Slide Time: 36:29)



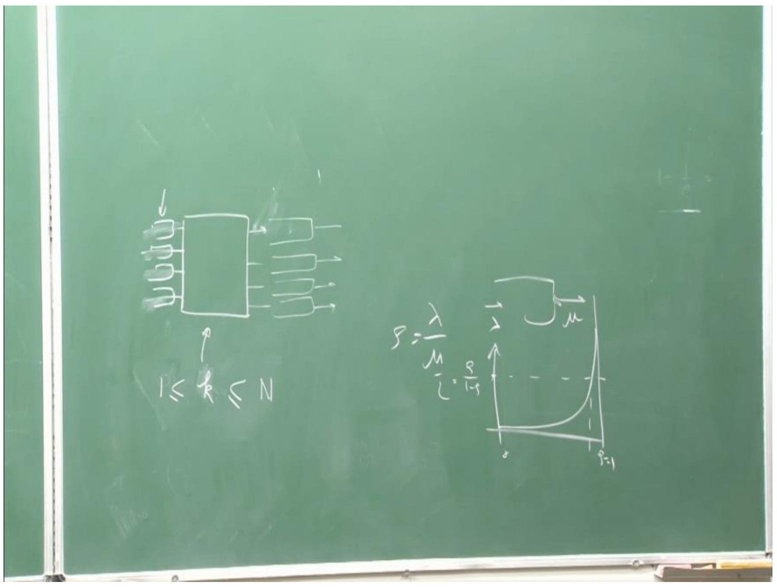
So, square complexity. Therefore, number of queues and then it is like I have to now map on to 1 to n possible outgoing thing. I find out one can be mapped here and then queue can be mapped here. So, I have to find out the maximum number of packets can be transferred in this slot. That is the maximum match problem. The complexity of the switch as i whole will increase in the switching case. Yes, compulsory the processing is increasing because we need an average. There is something for buffer complexity which also comes into picture for the same throw put performance, say this is not same as circuit switching. Because of the circuit switching, you are only worried about computation complexity is one thing if you have to do rearrangements or try to find out a path, and second thing is number of cross points. Number of cross points complexity is not the issue if mostly it is the processing which is the problem. Size rows processing complexity will increase.

In this case, again I think it will increase quite fast. Actually it is not desirable. Ideally I want everything to be or even lower than that. Every kind of complexity should be this or lower than this. The tendency of the switch is always going to have something on a square. You usually will not find anything greater than o n square complexity. That will be rarer phenomenon. If that is real, I think that is really were designed. So, you usually are going to have something in between this. Something in between this is o n log 2 n. You have seen all these items already. Remember you have got this o n log 2 n square kind of thing, many combinations. This exists in between. These are if you can get something better than this, if you can get over load log to n, that will be better because that is better than this o n.

So, all kind of computational complex the time required to make the decision, the delay part, the throw put performance, and the switch cross point. Switches are required as n grows. What happens to that and that is what complexity is. So, this has to be somewhere in between this and closer to this. I think it is acceptable lift can be even lower than this. It is still better. It tends to be 1 to n minus 1 n minus 1.

Yeah, it tends to be 1 to n minus 1. Actually one of them will not be used. One will be defined if it is the ith input port and then the ith buffer is not required. You will not be buffering for yourself. You will never have any factor directed to yourself. Now, I can further improve. You know this was one possibility to remove the head of the line blocking. So, usually and of course, there is a maximum match issue. That is another complication. So, people thought of an output queue switch, but we know that output queue switch is going to be difficult to implement. So, how that can be done?

(Refer Slide Time: 39:52)



What you do is, you have four ports. One packet is coming in every slot per line. There are four packets coming in one fine slot somehow because technically it is like a memory which is there. While reading out the packet if I read it four times faster speed, so switch internal is working and not at the line rate, but four times the line rate, I am taking an extreme case. So, line rate internally or internal speed is four times in this case. Then, all four packets I can read, I can analyze their header and all four of them can be pushed out to an outgoing port in one time slot. We will get all four packets, but in one time slot. The line rates are assumed to be same on both sides. Only one packet can be out to on the line.

What you will do with remaining three? You can put a buffer here. So, all four packets will be queued up here. The remaining buffers will remain empty. Actually in this, now this is speed up of four times allow this thing to happen, but who said is speed up always has to be of 1 or 4. I can have a speed up of also who stops. So, if you have a speed up factor of 2, what will happen? So far 2 packets contending I can push both of them to an outgoing port. The movement has 3 which are contending. I cannot do that. One of them has to be drawn. So, two of them will be pushed out and one of them will be dropped. Again some policy decision has to be there regarding this mechanism.

Now, this I am moving to a hybrid approach. If I want to even forget this particular loss, I can also have a input buffering. I can even have buffers here. So, when the speed up factor k, so maximum value of k is, k is always greater than or is equal to 1 or less than or equal to n, where n is number of ports. Actually n minus 1 is good enough. Again I have written n, but n minus 1 is good enough and if I am using somewhere k which is not n and which is greater than 1, I would also require input buffers. If I want to lose less, there is no loss which is happening inside the switch loss, less switch. So, this is the hybrid configuration. Here the input side is problem.

The input side can be a normal p4 or can this kind of system. So, there were summations known which come because of this hybrid thing. In hybrid combinations, there will be no loss. There is loss behind all buffers are full. Then, what you will do like means have to be loss in the case. See if you are having infinite buffers, there need not be any loss, but there is fundamental principle that will be number over view which is rho for a queue. This is mu is the rate lambda is arrival rate what is called rho 1 minus rho. So, what is the average to length n bar? This actually up load when goes to infinity. When rho goes to 1, you require an infinite buffer. So, if your buffer is only limited to some value, this is the maximum throw put which can be maximum load which can be taken up. That is the average value, remember.

So, statistically because this is an average for this rho, this is the average value. So, statistically there can be due which can exceed that average value and if you are having a buffer only till this, average value packets will be dropped in that case. So, actual average buffer capacity which is going to, they are going to be lower. So, there is a probability distribution which happens for the queue occupancy. So, the next step I think after the examination, we will look into the performance of the input queue system when head of the line blocking actually happens with that, and if I remove the head of the line blocking, then what happens is that we have already done 1 over 1 minus e. So, I will compare the situation for head of the line blocking and without head I am actually dropping the packet if there is a contention. These two scenarios and we will also analyze the output queued system and delay is also for that.

K is the speed up factor. K is equal to 1 goes taken here when do treated that full proof of analysis, where we go valid for both input as well as output. K is the internal speed up factor. K actually means in one slot, I can read two packets and we should prove it outgoing buffer. If k is n, I can read all k n packets. All k n will be read out to a single outgoing port which means k time buffer, we are reading and k time buffer we are writing also, but this is not both kinds. Switch is internally working k times at higher speed and then the line rates, see line rates are same or some r r packets per slot is the one packet per slot. My switch internal is k packets per slot. We cannot break it if the reading is something different. They are different, and then both are same. Clear?

How come this and this, the line rates are going to be same always? If you want to do that, you can do that and then you certainly require a buffer. You cannot work without buffers. You are reading a packet. A packet is been pushed inside in once to time slot, where you might be reading out a packet in two time slots. That is the case which we are looking. That is possible, but that is usually never used. The input and output pair, 4 pair always make it to common and get you on the other end within two ways two examples switching nodes. This output is connected to this input and this output is connected to this input. There is always a pair. There is always a bidirectional line. I am showing unidirectional and coming unidirectional outgoing port on the left and right side, but that is fine. Actual systems are concerned. There is always a pair. We cannot be unidirectional. So, with that topic, I close today and we move ahead from here for the analysis part.