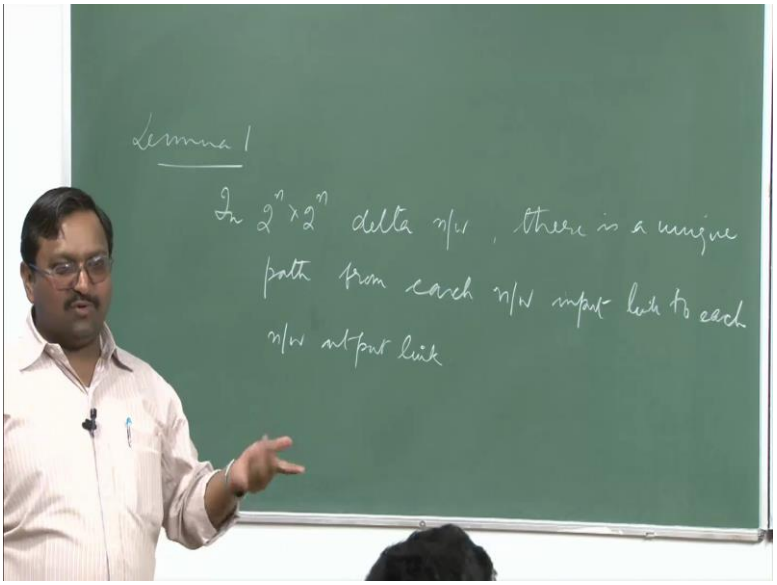
Digital Switching
Prof. Y. N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Lecture – 24

We will start with wherever, we left last time, and I think I planned to do buffer delta network analysis now. This is basically, buffer delta; how this whole thing will be analyzed by creating a stimulation model, as well as analytical thing, and then you will be comparing it with the delta network, which was there, which we have done earlier, and then of course the cross bar. Before we do this, there are certain properties, which were not evident, but now I think I am opening up one by one, all of them.

(Refer Slide Time: 00:53)



First one is a lemma actually, which comes from there. If you have carefully, observed the structure, you can immediately, know that this statement is correct and formally, there is no proof required on this. This is basically, 2 raise to the power n, by 2 raise to the power n delta network. There is a unique path from each network input link. When I am talking about a network input link, it is the whole delta network, the input to that. When I am talking about a switch input link, that is an input to switch in anyone of this stage. So, network input and switch input are two different things, actually. Similarly, it is true for the output; from each network input link to each network output link, which is obvious, actually now. This was the definition, which was given for, in general, for

banyan network. There has to be exactly, one path from each input to each output, and data also means that has to be unique; there are not multiple options.

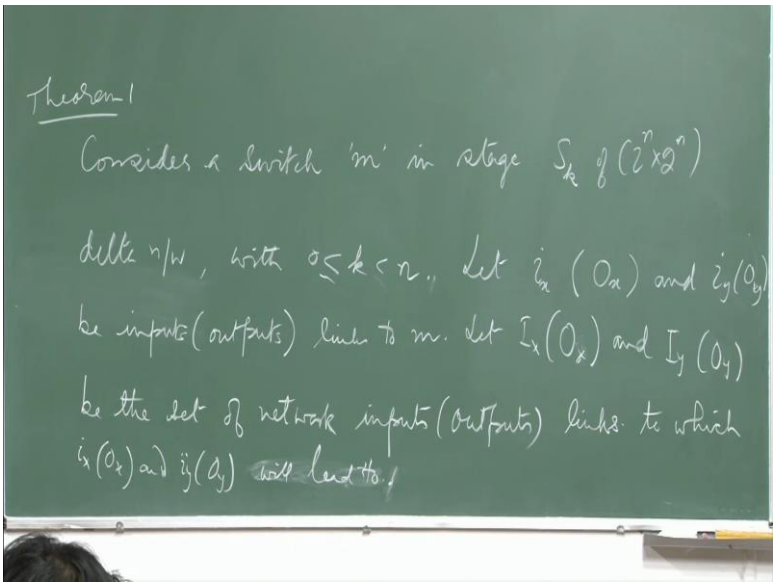
This is actually, going to be true. From each input to output, there will be 2 raise to the power n possible; each input to all the outputs, there will be exactly 2 raise to the power n paths, which are possible, because there are 2 raise to the power n outputs. This is obvious, because from the first one; this is for binary system; remember. I am not talking about a by b cross bars being used in between, but this is also true, even for that. So, this case, which we are discussing, is actually, buffer delta.

Student: Is path line a constant (( ))?

Yes, path line will be constant from each input to each output; that is true in terms of number of hops. So, it will be exactly, equal to number of stages minus 1; those many hops will be required for transmission from any input link to any output link, or whatever, n stages which are there.

So, if you keep on explaining, there are total n stages. Each time you are multiplying it by 2. From each input, you are going to generate exactly, 2 raised to the power n paths, and there are exactly, 2 raised to the power n outputs. There has to be exactly, one path available between each input and output, which has to be unique. This has initially proved, come from there; it is by definition. Now, second one is a more important observation. This is an observation, I have not actually told earlier.

(Refer Slide Time: 03:52)



Now, this one is a theorem as per the paper. I am just using the same notation. This says that you consider a switch m; m is a switch actually, in the intermediate stage. This can be true in general, for any kind of delta network; a raise to the power n by b raise to the power n delta network also, it will be true, but I am giving you a very general statement here in stage s k. So, stages are actually, going from s 0, s 1 to s n minus 1. So, that is the way, the stages are defined; total n stages. So, remember, stages are numbered from 0 to n minus 1. Yes, from source to destination. Since, paper is about 2 raise to the power n; it is a binary system. So, I am specifying it by 2 raise to the power n by 2 raise to the power n, but this is true in general, actually. This statement will be true in general. This actually, helps us in making the estimates; blocking probabilities and all that things, can be estimated from here itself. As I told, k k and t k value, but it will never be equal to n; it is always less than n. Now, what I am writing is an input to a switch; not input to the network.

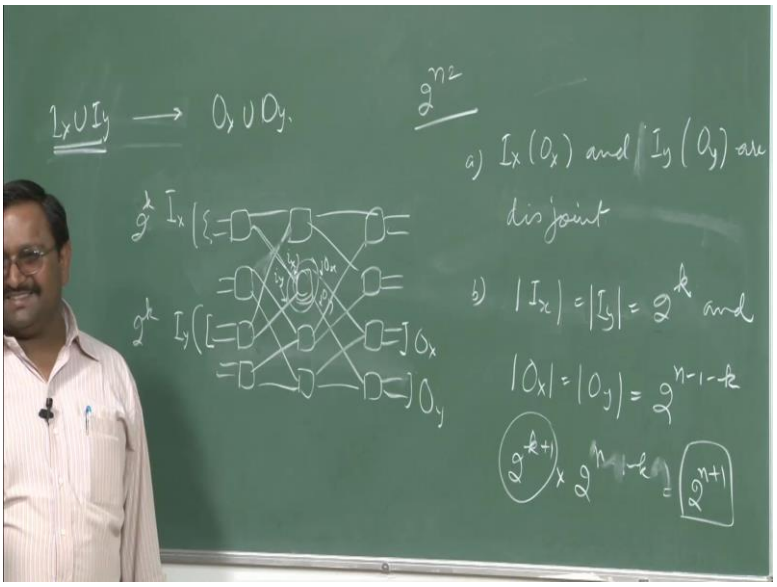
Student: Sir, k should be less than minus 1.

k is always less than minus 1; it is never equal to; there has to be exactly, n stages.

So, when I talk about input to the switch, equivalently, there is going to be also, an output. So, I am writing technically, two statements in one statement. So, you can replace whatever, is there in the bracket also. So, I can read; let i x and i y be two inputs, or I can say, let o x and o y be the outputs. This, I think, is a very standard thing; this is about the

switch. Remember, you take any switch in any stage; there are, if it is 2 raise to the power n by 2 raise to the power n; every switch is binary. So, there is one top and one bottom. There are two links; input links. Similarly, there are two output links. So, you can call them i x i y and o x o y, whatever it is. Now, I think I should draw this.

(Refer Slide Time: 07:25)



A flat statement is fine, it is clear, but usually, what will happen is there will be a switch. I am actually, taking an example, 8 by 8. So, let me make an 8 by 8, and I will give an example with that and I can take any arbitrary configuration of a delta network. So, I can take a shuffle or inverse Ben's; both are fine. Let me take a shuffle. You take any switch from anywhere. You take, for example, this switch. This switch can reach here. So, these two inputs can reach to this switch. So, I call this set in capital I of x. This is I of x; this will be I of y. This can reach here. So, these two; this set is I of y.

Number of inputs to the network, which can reach to this particular switch is this. On the output side, similarly, this is 1 and this is 1 pair; this you can call o of x, and this you can call o of y; this is a set. These are smaller o of x, o y. I am talking about these sets. Let I of x, o of x here, and I of y, o of y be the set; remember, be the set; it is not individual inputs, and it is not the input to the switch; its input to the network; that is what it means; to which, I x; I have to change the language; will lead, actually. In this case now, important observation; the three points, which are important observation. The first thing is your part a, which is the conclusion that your set I x or ox and I y or o y, are disjoint.

This set and this set will not have any overlap. This set and this set will also, not have any overlap.

Student: Overlap means?

There is nothing common between them; they are disjoint sets, which is obvious, because there is always a unique path from which, it reaches each output.

What if, this is the path this can reach here, and none of these inputs; I should be able to reach through this path, because there is uniqueness property; path has to be unique. So, I cannot reach to these inputs through any other path, except this. The first fundamental principal for banyan network itself ensures that this is true. So, sets will be disjoint. And of course, the cardinality of the sets will be how much? You count how many numbers of stages are there, before this. You are in stage s k, remember. So, how many stages are there before this? 0, 1, 2, 3, k minus1; k stages are there. So, number of elements, which will be there in this I x and y, both, will be 2 raise to the power k, obviously.

So, I x is equal to I y will be 2 raise to the power k. That is the first observation. Similarly, this should be after k, how many stages are there? You have till n minus 1. So, n minus 1 minus k; those many stages are there. You must be expanding through that. So, cardinality for that will be 2 raise to the power n minus 1 minus k. The most important thing is that if you take a set, I x union I y; any input, which is forming, which is from this input, from this particular set, is a union of two sets. Any inputs in this, if you want to connect it to o x union o y, this has to pass through this switch; there is no other option. This has to pass through this switch. All members, which are from this set, if any one of them have to be connected to any member of this set, the path goes through this particular switch. That is again, intuitive observation is going to be true.

Of course, the last one exactly, how many paths will be passing through the switch? This is going to be true here. You can actually, put; you have two actually, k will be; because this is 0, 1. So, 2 raise to the power 1 is 2. So, membership is 2; membership is 2. So, 0, 1, 2, which is n minus 1; 2 minus 1 is 1. So, again, output membership is 2, 2. So, total number of elements, which you can cover in I x is 2 raise to the power k; I y is 2 raise to the power k. So, total number of input network links, which you can reach is 2 raise to the power k plus 1, obviously. Total number of output links, which you can reach from this switch is 2 raise to the power; again, multiply this by 2; n minus 1 minus k. So, when

multiplied by 2, it plus 1. So, it is n minus k. Total number of possible paths; you multiply 2 raise to the power n plus 1.

So, every switch will have these many paths, passing through it; 2 raise to the power n plus 1 paths will be passing through every switch, and total number of paths is 2 raise to the power n square, because total number of input is 2 raise to the power n; output 2 raise to the power n; multiply, those are total number of possible paths, which can be there. But from each switch, this much will be there, and of course, you can actually, compute how many switches are there. Based on that you multiply by that; that has be equal to 2 raise to the power n square.

Student: Will you please repeat; this is not understood; the last one?

At least, you understand; these many paths, which passing through the switch?

Student: Yes, sir.

How many switches I can reach from here; input side? I can reach this, and I can reach this. This is 2 raise to the power k; this is 2 raise to the power k. So, 2 into 2 raise to the power k, actually, those many, I can reach, which is nothing, but 2 raise to the power k plus 1. On the output side, if you take one link, I will be able to reach 2 raise to the power n minus 1 k, if you look at the cardinality of the set. So, from the other link also, I can go to 2 raise to the power n minus 1 minus k. Total number of output links, which I can reach is, multiply this by 2, which is like adding 1 in the exponent. It is 2 raise to the power n minus k. I know these many outputs, I can reach through this switch; these many inputs, I can reach through this switch; multiply these two; those k will be the total paths passing through the switch. So, multiply these two. These 2 raise to the power n plus 1, and this is independent of the stage; remember, it is independent of the stage, and it is true; it is going to be same for all switches, actually. While, total number of possible paths, which are existing in a switch is 2 raise to the power n; on this side, multiplied by 2 raise to the power n; on this side. So, it is 2 raise to the power 2 n; total number of possible paths, which can exist. I think this is now, this is; there is a problem here.

This is not correct, actually. This is for all permutations; all permutations are not permitted.

Student: Banyan network?

Yes, it has to be a different one. It has to be, because this is 2; each possible thing. Now, you take from here; this is one path; second. So, there is 2 raise to the power n possible thing. So, total number of possible paths, which are there from this input is how many?

Student: 2 raise to the power n.

2 raise to the power n, and total number of inputs are 2 raise to the power n. So, it has to 2 raise to the power n into 2 raise to the power n.

Student: But all possible combination may not be.

Simultaneously, may not be permitted; that is a different question. See that is a simultaneous permutation thing, which you are looking at; not the possibility of paths, but technically, every input, can be connected to every output. But the moment, you set up a path, some other paths will not be possible. We are looking at that when you are using factorial method, we were actually, considering that particular fact. That is, suppose, there is no connection being set, when I set this path to here, there has to be at least, one path which, will get occupied. I can do it in n possible ways; I can set up the connection, but possible paths, I am talking about; possible paths, or the wires or possible ways in which, paths can exist. That is 2 raise to the power n square only; 2 raise to the power n 2; 2 raise to the power 2 n.

Each switch is actually, the number of paths, which can pass through. Then, all of them cannot; there can be exactly, two paths, which are permitted out of these. Possible path, which is passing though a switch is only, 2 raise to the power n plus 1, but only two paths will be permitted at any point of time, depending on, whether switch is in cross or bar state. If it is in cross state, then also, two will be permitted, and bar also, two will be permitted; remaining combination will not be allowed, but possibility of all paths is 2 raise to the power n plus 1. Total number of switches here is 2 raise to the power n, whole thing divided by 2; 2 raise to the power n minus 1. So, 2 raise to the power n plus 1, multiplied by 2 raise to the power n minus 1, will give you this only; 2 raise to the power n 2 n, actually. If you look at all things in the stage, all possible number of paths, passing through a stage, has to be equal to this, which has to be constant.

Important thing, the number of paths, which can pass through a switch, is independent of a stage. You take whichever switch in whichever stages; it is same. So, this fact actually, was not explicitly, told earlier. This comes from this. Now, before we go for this buffered system, we actually, have to make certain assumptions. So, without which, we cannot hold. So, let me come to assumptions, now.

Student: This will be true for banyan network also. (( ))?

Yes, it is true for banyan network. Only thing, banyan network; you cannot have that uniform tag for reaching to an output. For same output, depending on input, you might require a different tag. Delta networks; a common tag is good enough, which define the output port.

Student: What is the difference between exactly, banyan network and delta network, except tag?

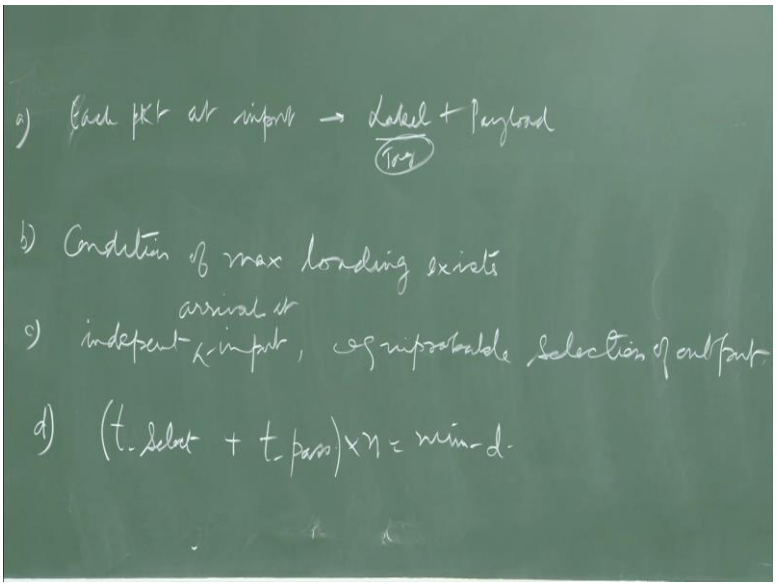
There is no other difference; it is only tag, which is the difference.

Student: Sir, tag is only for the (( )).

Yes, but then if you are using banyan; what I did in banyan network? I always said that in n stage, the packets switch are coming in. They have to come from the same output Id of the switches in the previous stage. So, in this switch, for example, if you look at this, I am getting 0. I am also, getting a 0.

This is a delta. This also, a sub class of banyan, if this condition; you do not even bother it is a banyan network. So, it is not clause or anything; there is exactly, one unique path, which is available. So, for 2 raise to the power n by 2 raise to the power n, built using 2 by 2, you require n stages, obviously, for banyan network. There will be exactly, one path by design. First one is a basic assumption, I think which is obvious; that each packet, which is going to arrive at the network input link at here.

(Refer Slide Time: 22:17)



We are not bothered about there is going to be an interface card or not. We are simply saying, there is going to be a packet, which will contain a label and a pay load. Label will be identified, to which output, the packet has to go. So, that is the first and foremost assumption, which is obvious; I am not writing it. So, I just simply say, each packet at input, consists of a label plus pay load. We call it also, a tag. Usually, interface processor will insert this tag. So, I am not bothered about the interface processor. Usually, you will look at IP packet header and then based on that decide what tag has to be inserted, after looking at routing table. So, this is being done; this is an assumption here. Of course, this is an important; condition of maximum loading exists. So, condition of maximum loading actually, means; at the input, there will be a buffer. Only one packet can be stored here. So, at the input, only at the input of the input links of the network; not of the switches, I am talking about.

Sometimes, it is possible that buffers will not be occupied here, but at the input, there is exactly, one buffer, one packet. The moment, this buffer becomes empty, immediately, a packet will be coming. That is the maximum loading condition. So, traffic is always available at the input to be pushed into the switching matrix.

Student: These switches or the whole network?

Whole network, I am talking about; not a switch.

Student: There will be only one buffer (( )).

One buffer for each input port.

Student: Each input port, sir?

There are buffers inside also, in buffer delta.

Student: So, we are saying 1 into 2 raise to the power n minus buffers are there. 2 raise to the power (( )).

2 raise to the power n buffers at this point; 2 raise to the power n buffers here; 2 raise to the power n buffers here; for a buffer delta situation. If it is a un buffer delta; the earlier case; then these buffers are not existing; buffers are only here. That is a maximum loading condition, when p will be equal to 1, technically. Of course, the output link, if there is a packet, it will be almost, instantaneously, taken out. Buffer will be emptied, immediately. Ultimately, packet is hopping. It is being read, one after another; it is not a cross bar. So, the moment it comes here, it is immediately, taken out. There is no re route time; re route time is 0. So, that is another assumption, which is taken.

In fact, interesting part is when the buffers are going to be here. So, you will find that sometimes, buffers are here, but you cannot feel this buffer, even if it is empty, because these two guys made a conflict, and they both wanted to come here. So, one of them was pushed here; other one was left in the buffer. So, one buffer emptied; packet came in, but no packet actually, moved here in this buffer. Here, the probability that buffer will be occupied, will be always, equal to 1. As you move across, the probability that buffer will be occupied; will start reducing from 1 to smaller value, as stage is increased. Ultimately, if I know what is the buffer occupancy probability here; that is the throughput performance, under maximum loading condition, and that is what we are going to do, ultimately.

Of course, third one is an important thing that to all input links are independent of each other, and then independently choose an output port; there is no correlation. Every packet coming in is independent; that is an important thing. Now, this is something new. At every switch, when two packets come in, you have to detect, whether to which outgoing port it has to go by looking at that tag; a digit in the tag has to be looked; and once you

have indentified, which outgoing port it has to be a red out. Usually, there will be a buffer here. It means there is going to be a 1 packet buffer, and it has to be red out. If they both want to go here, there is a selection time and then after that it will be passed; it will be read out, actually, to the outgoing port. It will go to the next buffer. Unbuffered delta technically, means if there is a conflict, one of them will be simply dropped. So, fresh packet is always allowed to come in. That is what it means.

A buffer delta; you will keep it there; you will not allow other packet to come in. So, what happens unbuffered delta technically, means if you decide this packet has to go, you simply transmit. Do not bother about whether, buffer ahead is empty or not empty, because if there is a conflict, one of them anyway, will be dropped. At every stage, this is what is going to happen. Buffer delta will not allow this. It will keep the packet in the buffer; it will not drop it. So, the packet from the previous stage cannot come.

Student: Sir, how to choose this, in case of conflict, which packet only (( ))?

Here, assumption is a random choice. Here, it is random choice, and remember, the arrival rates here, are also independent. Why they are independent, because the input set, I x and I y; they are disjoint. I have assumed that all inputs here, are independent of each other. So, take off the independent inputs; take one set; take another set; and if they are disjoint sets, these inputs also, have to be independent of each other. This independence is actually, holds through all the stages, because this independence of this I x and I y set, is true for also switches in all stages; it is a consequence of that actually.

Student: Sir, in buffered delta, we are not allowing the packet to be dropped in the intermediate stages.

Yes.

Student: That means, the buffer will keep on filling if we move towards left and that means, the packet will be dropped at input stage, then?

I am using maximum loading condition. There is some infinite buffer at the input; I am not bothered. So, you are not bothered about that. There is always, a packet available, when the buffer empties; that is the maximum loading condition. So, maximum loading

condition may happen at your utilization factor, which is when, it is less than 1; it is possible.

But inside, you will have unit buffers in that case. Delay now, consists of, this is basically, independent arrivals at the input and equi probable selection of output. The d part is, there are 2 parts; one is known as t select; there are 2 times. Other one is known as t pass. So, this is the time when, the selection is made, after reading the tag, and second time, it is t pass; that is one t slot, actually. When the minimum delay will happen is when, I am using unbuffer delta; if there are 2 packets, if there is a conflict, one of them is permitted; other one is dropped. So, how much time will take from input to output? So, this t select plus t pass, multiplied by number of stages; that is the minimum delay, which a packet can suffer. So, this multiplied by n is mindi, or the minimum delay in this case.

Student: Sir, what is t pass, actually?

t pass is, in a switch, when the packets are there, you have to first of all, select to which output, these have to go.

Student: That is a t select.

That is, that time taken is t select. After that the packets will be passed or dropped, actually; one of the two. So, that is a time in which, the packet will be transmitted out. It is like a store and forward; start a circuit switching; remember, you have to receive the complete packet. Then, you have to transmit. After reception, you do not start a transmission immediately; you have to make some decision. On time scale, actually, if you put, if I put in a time scale like this; you receive a packet; packet reception is completed here; you will take some time to decide; this is t select, and then you will start transmitting packet; packet is finished. So, this is your t pass. Usually, we will assume one of them to be 0, actually, ultimately, for doing the simulation thing, but real life, this is the way it is.

Most of this cannot be analyzed with close form solution. It has to be done through simulation, actually, but for certain cases, certain kind of bounds can at least, be estimated; that is possible. In case of when the network is without buffering, all packets will require, will be delivered in this case, or they will either, be dropped, actually; one

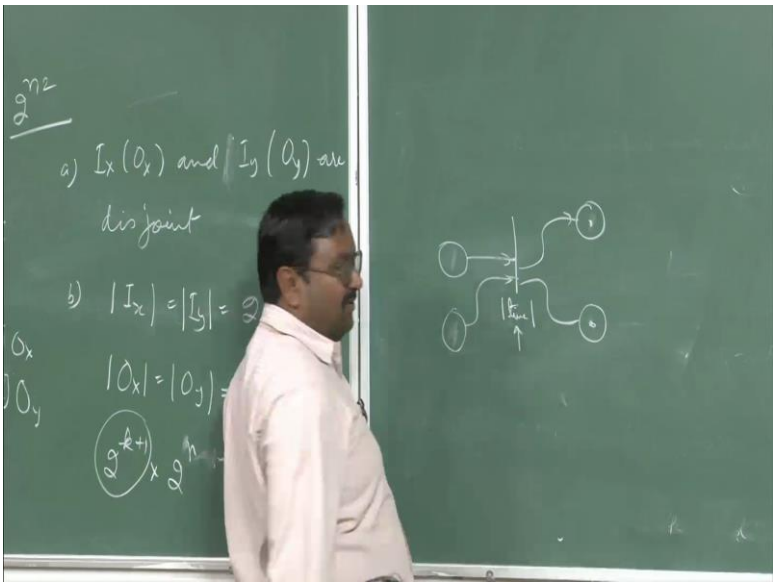
of the two things will happen. Either, they will pass through in this time, or they will be dropped; they will be lost.

Student: Sir, the passage term will be less than from one stage to the other, or the difference itself.

Which one? The difference; that is the propagation time, that is going to be small, actually. So, you can assume it to be horizontal line, almost; it is like a horizontal line. This one is first previous stage, and next stage.

Unbuffered, this is what is going to happen, but for buffering, this switch now, can be modeled as a patrinet. Patrinet model is, it will be of this kind. Again, I think some of you have not read patrinet. So, I will try to explain what is it, because each one of you have not done a course on communication networks, earlier. Patrinet is basically, we have some kind of token buffers; this is basically, used finite state machine depiction; it is for that thing.

(Refer Slide Time: 34:14)

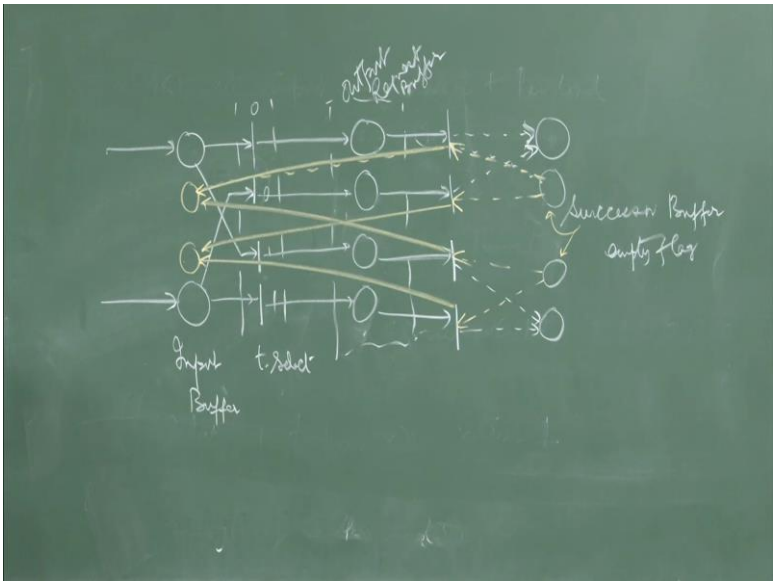


If there is a token here, and I connect it to something where, the decision firing will take place; for example, there are two tokens. This firing can only take place when, both tokens are available. If there is only one token, there is no firing possible, actually, and when firing happens, all the tokens on the input side will get consumed, and they will fill up the tokens on all outgoing sides. We then put certain restrictions; this is what I will

do. For example, I can put a token here. I can put a token here, for example, So, if there are two tokens here; there are no tokens; so condition has been met for the firing, and I will end up in getting two tokens here, which then can be further connected.

I can also, put a time here, is known as delayed firing, actually. I can put a time; time patrinet we call it. When the condition is satisfied for firing, after that there will be a delay, and after that delay only, the token will be generated on the outgoing token holders. So, I can build up, I think, lot of machines in this fashion. I had done earlier, I think, one of the data link control protocols was implemented this way, in the previous semester, that is basically, the fundamental, but now let us build up the switch, because I am now doing backward propagation of the buffer status. In a buffer delta, the buffer is busy. I have to tell the previous stage that buffer is busy; you cannot send any packet to me. Packet only can come, if the buffer is empty. So, I have to build up the patrinet model for understanding this. Actual implementation, it means, there is going to be backward signal path also, which will give the status back; if you do the actual implementation of buffer delta; otherwise you cannot do that actually.

(Refer Slide Time: 36:19)



I will use two colors here; one for the data, and one for the control path. Data will contain now, two paths, and this is known as input buffer. There is nothing like an output buffer, because output buffer for this switch is nothing, but the input buffer of the switch in the next stage. Input buffer in the next stage is nothing, but the output buffer for this.

Then, of course, if a packet comes in, there is some time, which it takes for making the selection; which outgoing port, it has to go. Once the packet is here in the buffer, this will cause what we call a selection process. So, there will be, we call it p select, will be the p rate for this. Now, this packet can go to up, as well as, can go to down; both ways.

There will be, I have to now, build up technically, four things. After this time t select, depending on; now, the firing will depend on outcome of, firing will depend on the bit, which is controlling the routing. If that bit says, it has to go to bottom, in both of them, one of them will not fire, because here, I am doing a selection process. So, whatever is outgoing buffer, which is connected, outgoing token holder, which is connected to this particular firing edge; will not get a token, because that will depend on the selection thing bit; it is a bit control. That is not explicitly mentioned, but that is how actually, it is. This in turn, will then; this is known as output request buffer. So, they are requesting the output. If this has to go to here, this will actually, be fired up; this will not be. That is decided by bit. If the bit control is 0, it will come here. If it is 1, it will come here. If it is 0, it will go here; for 1, it will come here. So, there is output request buffer; there will be four of them. One of these will always, be active; not both.

Similarly, one of these will be active; not both of them, or other way around. No, it is possible; both of them can be active; both of them can be on at any point of time. If there is only a conflict, then only, one of them will be. So, you will keep on holding and try next time. If there is a conflict basically, what happens? This is 0 and this is 0; both of them will come here. Both of them should fire up. I should actually, hold up output request buffer. I think both models are possible; I will explain this. This is one particular path. Now, this is the input buffer for the next switch in the next stage, and usually, I will have this status, which will be coming here. We call it a successor buffer empty flag.

If this buffer is empty, this will be set actually, immediately. So, they are linked together. There will be two such things. I will use this color for. Arrows are important here. Same thing, this will be for the previous stage. Whatever, you are seeing here is for the next say switch; similar thing also, has to happen here. There is a backward propagation of the control signal, because of that. For example, if this guy selected the upper one, it will pass the packet to the outgoing thing, and this buffer will become empty, actually, in that case. So, I have to get back this; will come here. In fact, this should be, highlighted with red. Of course, if this one passes through, instead of this one, this token will get

consumed, and I will set this particular flag in this case, and then they will talk back further. So, backward propagation is there.

So, this t select and only, one of them, should be there, actually; both of them cannot be there. That, I think is taken care of by this conflict resolution, t select. So, if both of them want to go to the top, both of them will come here. Only one of them will be fired up; other one cannot be. So, it will remain there as it is, in that state. After t select, this output request will be there, and this will pass through, when this condition is met, and as this packet is going out, this buffer status flag will be reset and this can then communicate back. So, new packet can come from again, this direction. Usually, t pass will be the time in which, the packet will be passed on to this buffer. This will also include usually, this signal time; backward propagation signal time can be included in that; it does not matter. Every time, when you are forwarding your packets, signal is coming back. So, include both of them; combining them does not make any impact.

That is how it will be actually, modeled as these are (( )) model for the buffer delta system, actually, for a 2 by 2. If it is a n by n, then it is, of course, going to be tricky. Now, what you are going to analyze, when you want to do a performance analysis. In any network, one of the most important things, which we need to do, is throughput performance. You have to analyze it under maximum loading condition. Ideally, if it is possible, you keep on changing your input arrival rate or input probability of getting a packet in the slot, and then do the analysis.

Second thing, which is important, is known as turnaround time. This time is nothing, but the time difference between two instances. The first instant is when, the packet is arriving at the input port, or input link of the network; not of the switch of the network, and when it reaches to the output link of the network; the gap between that is what is your turnaround time. So, minimum turnaround time will be mindi or minimum delay. Similarly, maximum throughput will be what you will get with the buffer delta; unbuffered delta, actually.

So, that is a bench mark, which can be taken. Based on that you can do create a normalization of whatever exactly, you will compute. So, I think I will close now. Then, we have to do now, basically, unbuffered delta analysis first. So, this actually, I could have done it earlier also, but I am doing it now, along with them; cross bar and then of

course, the analysis of this buffered delta system, and then we can basically, make the comparison of the three.