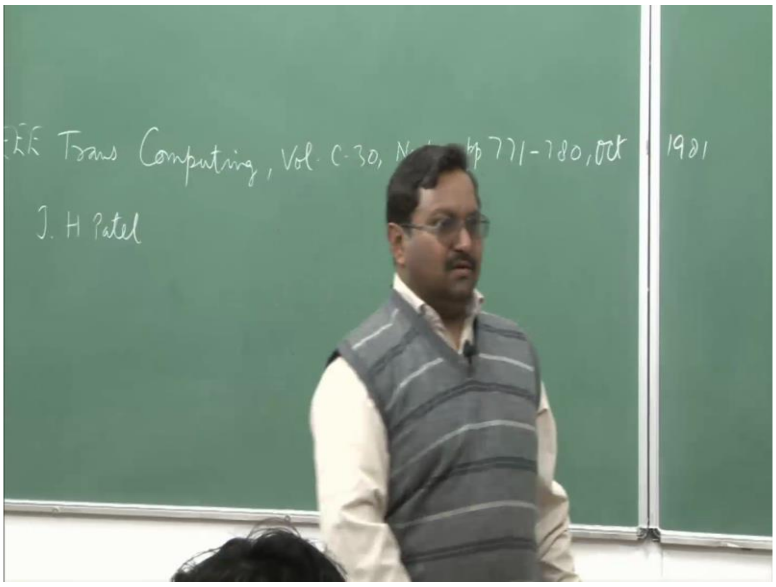
Digital Switching

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

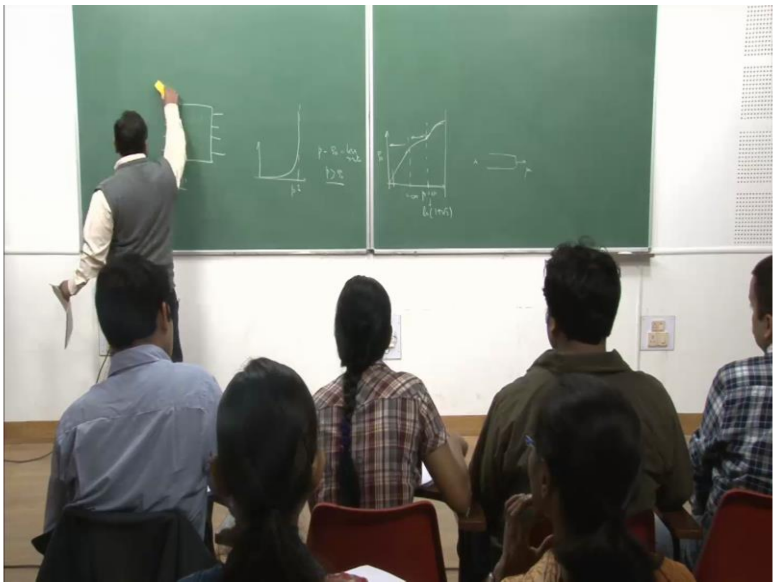
Lecture – 20

(Refer Slide Time: 00:14)



This performance of processor memory interconnection from multiprocessors; here, it is not written; October 1981 is a triple transactions on computing, because remember, switching problem has been common in; this basically we are not trying to look into packet switches, which has to do with nothing, but interconnection systems. That is how the paper was actually here; performance of processor memory interconnection from multiprocessors. I will upload the paper, actually. I will download from explode site and upload. Those two papers, I have to upload; earlier one and this one, also… Now, I think, where we left last time; we actually, had discussed, I am just making a quick recap and then will move ahead.

(Refer Slide Time: 01:39)



What we had done was we have taken a simple cross bar. We put the input, and we took two, actually, possibilities where, once we will use the input queues; there is no speedup factor, and we analyze the performance for that. We had done before that an output queued system. Output queued system can operate, till your load becomes equal to 1. So, when load becomes equal to 1, then only, the queue length explodes to infinity, as well as the delay also, goes to infinity. So, that is Marcovian arrival, deterministic departure process, single queue infinite buffer. It actually mimics that system, essentially, that was the thing. In the input queued system, what we figure out was whenever, your p is greater than that 0.881, at that point of time, your throughput will be higher than 0.58; whatever, was the saturation limit, under fully saturated condition, when full load is there.

Unless, this probability higher than this; your throughput will not be higher than 0.581. So, what usually, will happen is when the switch will start operating, you increase your p your throughput will also start increasing. At some point of time, when p is greater than 0.881, at this point, I can achieve more, the higher than 0.58. After this point, I should switch over to dropping the packets, actually, from the head of the queue, because then only, I will be able to achieve a higher value, and then from 58, I should be able to go to 0.631 or something, whatever was there for infinite n. So, I will be able to push it more. Before this, it will be going something, like this way and then I can further; I will do a switch over at this point. Your packets will start losing as you grow at some 0.581, roughly around that time. Before that whatever you are pumping is lower than what is outgoing rate. So, if you have actually, infinite buffers, you will not be losing anything, but the movement, you cross 0.581, your queue will actually, become, will start exploding, because outgoing rate is smaller and incoming rate is higher, and then of course, the throughput will be small here. So, there will be some cross over point. So, below this, the queue is stable; this point, the queue is unstable. So, you will actually, ultimately, have to start dropping the packets after this point.

At this point, anyway, you are dropping, but from the head of the queue; not when the buffer overflow happens. Here, you will be doing, when the buffer overflow actually, happens, but following this study does not make sense, because your throughput will be less than 0.581, because that was the condition by which, this was derived; 0.881 is makes you log of 1 plus root 2. So, that is, I think, what we did. So, you move at from here, and now, you go to another domain.

Student: Sir, did the queue is over after this 0.881 find you can never also.

Yes, queue, now you are dropping. If you do not drop, then queue is unstable. What is the stability of the queue is that as time grows, because outgoing rate is mu and lambda; will lambda is going to be higher than mu; outgoing service state, what is maximum possible. If you are putting more, queue will start increasing, and it will increase until, it becomes infinite; that is the instability. It will remain statistically, constant; the mean value remains constant; you can estimate a mean. For unstable system, you cannot estimate a mean, actually. There is no steady state condition; it is transient now. Because 0.581 is the maximum throughput, you are pumping packets at higher than 0.581; what is happening to the remaining? You are not also, dropping here. So, queue must be exploding, and queue will then build up to infinity. So, this is an unstable condition. So, usually, you will be actually taking, for example; I will have buffer size of 100; over 100, I will start dropping till somebody, these guys are pushed up further.

Student: But this 0.581 be analyze for infinite both this they through put we have.

0.581, under what you call fully loaded condition; there is always, a packet available. When p is equal to 1, what is the maximum throughput which, you can achieve; that is a saturation throughput; that is 0.581, and I know if my input rate is higher than 0.581; it is 1, for example, that is what we have done. We have looked into unstable situation, but you are not looking to buffer depth. You are only looking at what is the maximum saturation throughput which, you can achieve; just 0.58.

Student: What will you have limited after that.

You have to drop ultimately; cannot help it.

Student: But graph will remain the same?

No, graph will not.

Student: What is the impact of the buffer size on the throughput?

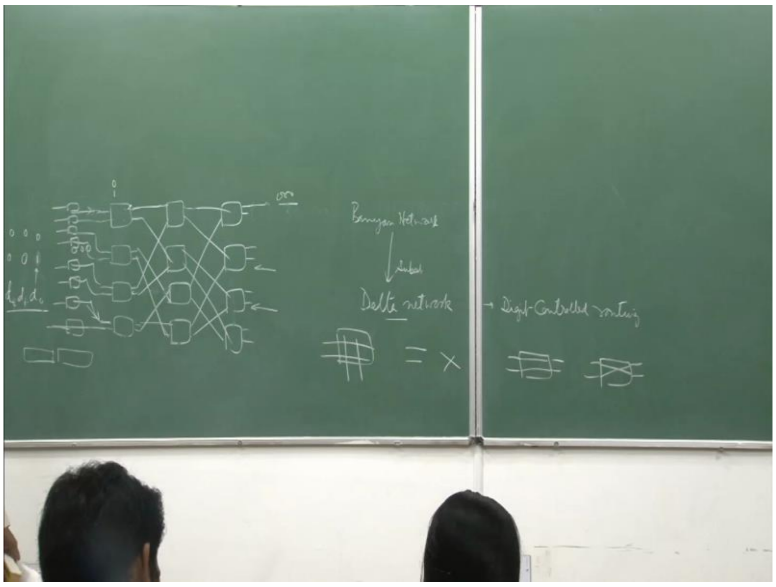
Throughput actually is differ. This will not; will remain same; it does not matter for the curve as far as concerned. Curve is not showing where, the packets are being dropped, or where, the packets are being always retained in the system. So, throughput will be this. Even, if you do not drop it, only thing, you will have instability, or you will have infinite queues, but throughput will remain same. So, it is symmetry; it is not shown here, but I think you should be able to reduce this. You should not assume that there is no packet being lost in this system. There are different things happening in different portions. Even, technically speaking, because if you say 100, there is a finite probability even, with very low load condition; your buffer depth might become 100; very small probability. If that happens, packet will be even, dropped there, but the probability of packet lost will actually, start; will be very less here, until we start increasing; it will very high at this point, once it crosses; 0.581 will be there. So, you can actually, estimate even the probability of loss, but that will be the function of buffer depth, but I know, on an average, how many will be lost.

You can actually, find out. This is the maximum throughput, which is there; row naught and whatever is p. I told you that p minus rho naught packets, are going to be lost, per unit line, per slot. Those will always be lost; you cannot help it, and in this scenario, p is always greater than rho naught. You have to always understand, p is always going to be greater than or equal to rho naught, and this is loss rate. So, loss will always, be there in the beginning, because root p will all be almost, equal to rho naught; it will be 0. The moment you cross 0.581, is the maximum; when p is more than 0.581, you know there is going to be loss. If you have infinite buffer, till p is equal to 0.581; there will not be any loss, because after that only then (( )) stability will come. You will have stable queues, but may be, you might require a very large buffer depth; that is only thing.

Now, coming to; this was fine; we were actually, using cross bar, but handling the things with cross bar is complicated. Can we do something better, actually? Secondly, there is a processor, which has to analyze all the packets, and based on that it has to keep on setting all the cross bars. As you increase size n, n by n cross bar; this becomes a complicated thing. Can I make a simpler system; that is the idea, now? So, people thought about it. Then, this was thought kicked; why I cannot build up a switching network itself to act as a switch.

Remember what we did earlier in class network, because I could not have maintained larger n by n; I thought of multistage interconnection network. We came up with class, and all kind of variants, which we have studied. Then, we say it, the properties of those interconnection networks. Same thing, we can do with the packet switching system also, and I want my switch control is still, in that. For example, class network; when I have to setup the path, all inputs that I has to still come to a single processor, but we were trying to reduce the cross point complexity; not the computational complexity. Here, I now, want to reduce the computational complexity, actually.

(Refer Slide Time: 11:07)



One of the things which, we can do is we can always, create an interconnection. I am showing you a very simple one. So, 8 by 8; I want to build. There is something called banyan network. Remember, in circuit switching, what we had done was we have always, studied the switches where, there are multiple paths. So, you can always, setup from a free input to free output a path; we have actually, built those kind of scenarios also. Sometimes, it is not possible. Banyan network is a special category of multistage interconnection network, whereby, from every input in the middle stage; for example, these switches are like small cross bars; small packet switch cross bars; the one which, we had looked, but it is without buffers. It has a speed; we assume that it has a speedup factor of 2, if it is the 2 by 2 cross bar. If it is a 3 by 3 basic element, speed up of 3 is being used. So, it is like a output queued system, but there is no output queue. If there is a contention happens, (( )) packets are actually, dropped in this case.

So, I am looking in to 1 buffered banyan network as of now. There is a category of buffered banyan networks, also. So, if time permits, we will try to look into that analysis; a buffered banyan delta networks, basically. What we have to do is from each input to each output, I should be able to have exactly, one path. If that is possible, there should not be 2, and there should not be 0. There has to be exactly, one path; it is a banyan network. Routing is going to be complex phenomena in this case.

Student: (( )) in the curve.

Why?

Student: Because if you have a once more napping then only packet not.

No, from this input, I want to go to this output. There is another output, I want to go. A path exists from here to here; a path also, exists from here to here. So, I can make a same packet; I can take and I can make two copies, and then push one here, and one here; how does it matter. So, wherever there is a common path, only one packet goes; wherever, the split happens, I make duplicate copies. Multicasting also can be implemented in banyan networks, but this has to be done by the intermediate switch. So, only problem is who will control? In a class network, there was centralized commands, centralized control, which was controlling everything by analyzing the inputs. Here, I want to do away with that ultimately is the limitation, because remember; in circuit switching, when you are setting up a circuit, you have sufficient time when a request will come. You set up one circuit at a time. You basically again release every circuit one at a time. May be, call is there for a 5 minutes.

So, you have computing time. Here, computation has to be done in every slot, every packet slot. All addresses have to be analyzed, and a new configuration has to be computed and put. So, it has to be that time, less than 1 percent of the slot duration, if roughly, 1 to 2 percent is a guard bin period between two slots, which is a very tight requirement, actually. While, in case of multistage interconnection for circuit switches, you setup a call for, say half an hour; how does that matter? You are going to do a calculation only for, say 30 seconds; does not matter, actually. It is acceptable. Only problem is; and also, you are not computing everything; current state, you maintain; we just reorganize some of the states, some of the paths; setup a new path. The computation is not too heavy. Here, it has to be computed for the complete input to output map in every time slot. It is not differential computation. It is an absolute complete computation, happening in every time slot; that is the issue. For example, let me draw a simple one, and of course, common sense tells; since it is 8 by 8, every input has to fan out to 8. So, I can connect in any arbitrary way.

Now, can I connect this thing to here? Is it a legal connection? No, it is not, because then if I setup this line; there will be some output to it which, I cannot make the connection, from here. I have just drawn; I have not followed any structure, and I think, you have exactly, one path from every input to every output. Other one also, would have been a multistage interconnection network, but would have been two networks, which are having some coupling between them.

Student: (( )).

Yes, there has to be exactly, one path and there, interesting things, which will come. We have something called a permutation network, and we also have we can look into how many possible maps, which can exist. I will show an example. Banyan network is fine, but now, how to handle the control part of it; that is the issue. Within banyan network, then there is another classification. There is a subset. These are superset. Subset of the banyan network is something called delta network. Another thing, which is termed, which is used is known as digit control routing. Here also, I can implement the same thing, but I have to understand the logic. Usually, lot of our routers are actually built using this method; this kind of self routing. This is known as self routing metrics. On the input side, there is a board, which will look into every packet, will analyze what is the destination. There is going to be exactly, one unique outgoing path, which will map, corresponding to destination. It will then take that whole packet. It knows that from here, I should be able to reach to any path by putting up a tag; can be appended, actually. Tag will have certain number of bits or digits, and each digit of that particular tag, will be each different particular digit. For example, I can put, in this case, digit d 3, d 2 and sorry, d 2, d 1 and d 0, and then I can build up the matrix such that d 0 will used to switch here; d 1 will be here, and d 2 will be there.

So, I want, for example, to root this input to this output, and if my switches are such that; the control digit for a switching element, if it is 0, I will be routing at to the upper side; if it is 1, on the bottom side. There will be a contention, actually. Remember, there might be a possibility that two packets come, and they both want to go to up, then there will be a contention; I have to drop one, and I can (( )); that loss can always be there. That is the reason why, in router, sometimes, packets actually, do get dropped, because there is a buffer limitation, and they also will have contentions of the link; this kind of thing. So, from here, if I want to go, I will have a d 0 as 0, so that I can reach upward, but then I cannot reach to this particular point. So, this has to be 1. So, it goes to bottom, comes to top; it has to go to, again, these things; there will be 0, again 0. So, 0, 0, 1 is being used here; it will be always going here. You take some other 1, which also want to go to same outgoing port, the top one, for example; it will also require 0, 0, 1.

Look at some other switch; this one. This one will require a 0, 0 and 0. Look at this one; this one will require, corresponding to this; it will be 0, you have to go to top; again, you have to go to top; again, you have to go to top. Now, my problem is my output port is still the same. So, I can put this Id as 0, 1, 2, 3. For same output port, different inputs have to put, different tags; that is one issue; that is one possibility. I want to identify the property of a structure, such that if I want to go to this outgoing port, I just put simply that tag value is 0, 0, 0 everywhere, irrespective of which input is taken by the packet; it will always, reach that output, same output.

Student: Sir, here, rearrangement in as is a mapping is unique here. So, rearrangement is not possible.

The arrangement of?

Student: Suppose I want to connect one input to one output. Next time, I want to change the configuration. So, that is not.

Only one packet can go; that is it. If there is a contention, packet will be dropped; there is no alternative path.

Student: Suppose, I have established my network. Can I change the?

No, you establish every time slot

Now, you do not even, worry about time slots. The packets come, just do add a cross bar for every packet. Every slot, you do a cross bar for different switches. It is now, reaching here, separately. Then, they may not be synchronized. There will be a delay in going from here to here. So, it will be working based on whatever packets, which are coming here. So, every packet is working, independently. Only thing what I am doing is instead of analyzing all addresses in central command, central and then setting up the things; I am analyzing all the headers at the interface board itself, and based on some local routing table, which is given into these boards, I am now estimating what routing tag, which has to inserted; it has to go in front of the packet, and this packet, this tag is going to be analyzed. A particular digit in this switch; it will be switched out. Once it reaches, the next tag will be done, whatever, is the tag, certain another digit will be used. So, in this case, for example, d 0 was used by this guy. This guy will be using d 1; third stage will be d 2; these are add ported configuration.

Student: This stage we are removing a single digit and?

We are not removing there, but these have been programmed to you use only certain specific digit in the tag for routing.

Student: Sir, 0, 0, 1 will go to which port sir, in that 0, 0, 1.

From here? 1 is being used, that comes to down; goes up, again 0 will be there; second digit, d 1; again, 0; it goes to top port. See, this is the problem, which I am now trying to come up. Banyan network is a general system, but it may happen that from this port, you require a different tag; from this port, you require a different tag to reach to the same outgoing port. This, I want to remove. If you can remove this, and ensure for reaching an outgoing port, you require same tag at all input ports. What you get is the delta network. So, the delta network; remember, will be a subclass or a sub network; it is a subset of the general banyan network. Banyan network; this condition does not hold.

Student: Delta networks in is more than self routing matrix plus this use interface board which is going to analyze a odd coding d 2, d 1?

It is not a user small interface.

Student: Sir, but that additional facility?

Is required always. Now, it is a distributed control. So, you can actually, build up very large size switches.

Student: Then, what this interface port set complexity will not shift to the?

You just simply put the routing; see how you will do the higher amount of computation? Have multiple processors.

Student: So, you are distributed computation stages

Instead of having one single controller, which is going to do all computations, I am using now, eight things. So, you can use lower speed ones, if you wish.

Student: This will not require any processor?

Processor will be required in interface book. Technically, every switch board will have some micro controller or processor, because it is to understand what is the digit coming in that tag analyze, and then do the cross on bar. So, there are smaller processors in every switching element, and the board will also have a processor.

Student: This competition will be much lower than what we have in interface broad where, they going to analyze and hard code the d 2, d 1, d 0, sir?

But how many packets you are analyzing per unit slot.

Student: 8, I assume.

One; there are eight interface boards. This also has its own; this also has it is own; this also has it is own.

Student: One port will be able to be, sir.

Line interface unit is this.

Student: Then, we decrease the complexity.

Suppose, if you have a high end processor in interface board, it can process for eight ports. For eight ports, you can make one common interface board where, eight lines come in and eight lines go out, but now, you have a choice. You can make board for each line, separately. Low end processor; you can combine two, and use one single board; combined four, and use one single board. So, flexibility has come in, now; otherwise, large dimensional switches cannot be built. So, this is nothing, but an engineering solution, honestly speaking.

Student: These switches are identically, three stages.

These ones?

Student: Yes.

These are basically, now 2 by 2 cross n bar; this is a cross bar switch, and I always represent either, this is a cross or bar stat. Cross state means you take a 2 by 2 switching element. This is a connection configuration. Bar state is; only thing is that if you can ensure that whatever tag value I am using, if these tag values are used by any of the incoming port, it will always reach to the same outgoing port. You can still do something better than this. If you ensure that you can give the address 0, 0, 0, 0, 0, 1, 0, 0, 0, 1 and 0 and so on, for all outgoing ports, if that address itself is used as a tag; that is still better for you. You do not have to remember tag corresponding to one outgoing port, but that problem is very trivial; you can even, fix it through a remapping of these outgoing ports.

Student: Here, it can lead to packet loss, but if you do delta network, it would not lead to packet loss?

No, it can always, leads to packet loss. Why it will not lead to packet loss? Why that assertion is there?

Student: Because in this case, let us say two packets going to the same output port, that one each is to be dropped, but if buffering is allowed in the input port itself.

But how do you know that there is a contention?

See, I am assuming separate interface board for every line; this guy only knows about this line; it is not aware of these; buffering is not known. You have to do something only, at the buffers; in these small, what we call switching elements. Contention can only be known here, because it is going to analyze both the tags, the digits in the tags. If they are same, then there is a contention. So, only analysis can be done by this controller of this particular switching element. That thing either, actually, that is a buffered configuration; I am not doing that thing as of now, because that analysis or simulation is much more complicated. So, you can either, put buffer here; input buffering. You can put an output buffering, and then let it go. If these buffers get full, you have to drop them. So, these buffers are only used by this particular switching element; they are not shared buffers, or you do not use the buffers, actually, as simple as that. So, important thing is that you cannot use input buffering or output buffering in this configuration. If the input and output buffering can only be used within these switching elements; that is a difference between cross bar.

Student: That is an external buffering, this an internal buffering?

Yes, you can call it, but see; remember what I have done that in that cross bar, that same cross bar, I have just pushed it here. Think, what we discussed in the previous lecture was this single unit.

Student: Specific to a single?

Of course, I have shown larger dimension; that is a different issue. I can even, build up a larger, using larger dimension switching element, whatever is technically, feasible and join them, together to build up a larger size banyan network, but you have to have a well defined structure, so that self routing can happen. Self routing can happen, but I want now, even the addressing; if I use the same tag at any input, it will always go to the same output; that is a delta network configuration.

Student: This may not be specific to the particular time, because if I put 0, 0, 0 on the first line, and if I put 0, 0, 0 on the last line.

They should, somewhere, they will collide.

Student: Their destination is totally different.

No.

Student: Though, the architecture is (( )).

In this case, if it is delta network, then they will always, be reaching the same destination; same outgoing port; not destination. See, what happens is this board is now, analyzing the; for example, if it is an IP packet; it is analyzing what is the destination IP address. That destination IP address may not be attached to this port, but this is the port, which is connected to a link, going to the next router, which is in the path to the destination. So, only thing is that you build up what we call forwarding table, based on the routing table. Routing table; what it contains is who, is the next hub to reach to certain IP address. So, given a destination IP address, you know what, is the next hub. Based on that you have a forwarding table. For this particular thing, I have to go to this outgoing port; not the next hub.

Student: That delta network will be having standardized output ports, output addresses?

This is also a standardized output; this also got 0, 1, something, everything, but the text, which are going to be used, will not be uniform for the same outgoing port.

Student: And that will be used as digital control routing?

We call it digital control routing, because of that; because, I am using digits. Even, this is also digit control routing, but this cannot be used, because you are using different tags; obviously, it is not that I cannot use it. I can use it, but it is not preferable. I will better, actually, build up a delta network. So, let us see how delta network actually, is going to be built. I will take same 8 by 8.

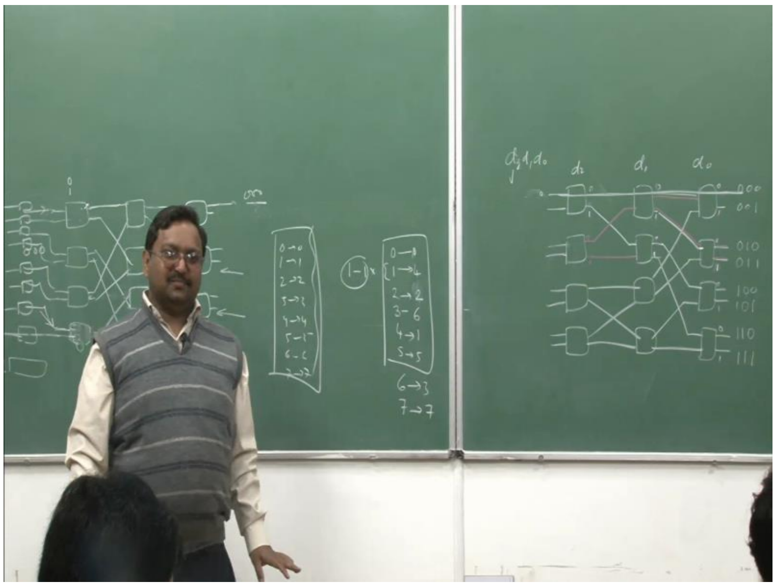
Student: Sir, why we call it delta?

Do not know; I think it is also, not a short form of digital control routing. I have not come across any reason; just people, just call it, and why it is called banyan network; I do not know. Same question can be asked there, but answer is not there. Sometimes, I think, some of the inventors who invented; based on their names actually, the things will come.

Student: Is banyan not same as the banyan tree?

No, banyan tree of switches or banyan tree, which actually, grows; agricultural thing. No, it is nothing to do with that. It certainly, does not look like that actually.

(Refer Slide Time: 33:44)



What I will do is I think, let our object to be; all outgoing ports will be addressed from 0, 1, 2, 3, 4, 5, and so on, and I will write them in binary form and ultimately, I will give you general a statement for a delta network connection. So, I am making a 3 stage system, and I hope by the end of it, you should be able to figure out what is the magic, which goes behind the delta thing. So, I want to actually, root, and I have numbered the output, say 0, 0, 0, 0, 0, 1. Now, from this input, if I want to; what I will do is I am actually, writing all my tags as d 2, d 1 and d 0. So, I will be using d 2 here, d 1 here, and d 0 here; for controlling the switching elements. So, pushing in a 0, 0, 0 from here, will move the packet to upward side, upward side, upward side; I go to 0, 0, 0. If I put in 0, 0, 1, it has to go to the bottom side, sorry, 0, 0, 1; it will still go to the upward side. It will go to upward side, and 1; it will come here. If I put 0, 1, 0, it will go to the upward side, 1; it has to come here. So, I should connect it to this. If it is again, 0, 0, 1; 0, it will come here; 0, 1, 1; it will come here.

Student: (( ))

When I am putting a tag of 0, 0, 0, what happens to the packet? So, 0 means upward; 1 is downward; in all switching elements. So, 0, 0, 0; I come here. 0 1 0; I come here. 0 1 0; I get 0, 1, 1. So, remember these; whatever, I have written in this; just read in that sequence as you are moving. It is 1. Remember, the outgoing which it matters; 1, 0, 0; I will come here. 1, 0, 1; I will come here, 1,1.

Student: Sir, it will be 1 in the second row.

This one?

Student: No, next; third stage, third column. (( )).

Second row; this is fine, and this is what? This is nothing, but at demultiplexer tree. I am demultiplexing. Based on first digit, I am splitting either, it is going to the 0 portion or 1 portion. Again, further I am demultiplexing, based on the second digit, and last using the third digit; 1 is to 8. I can now use a second one, and create a demultiplexer tree. So, this is for one input demultiplexer; this is also valid for second one. Again, combine these two things here. I can create a demultiplexer tree for the other one, and remember; this is the path. If a packet goes, it will take these paths.

So, if I create a demultiplexer tree for this, I forget this particular tree; there will be overlap, whichever overlap is the common part. That is where, the contentions will actually happen. So, for 0 actually, 0, I created this whole tree; for 1 also, this is whole tree. So, 0 and 1; they will actually, contend on anyone of these lines. So, contention between these two will always be happening in this switch itself. That is what actually, it shows. So, from here, if I put a 0, it will be further split. So, contentions within these will be handled here, because they will share the same tree, but with one input here and one input here; it will be apt thing in the second stage. This will be nothing, but inverse Benz network, actually.

Student: Sir, how to reach 0, 0, 0, from?

From which one?

Student: First one; pink colored one.

This one.

Student: Third, yes sir, I said.

If you have a 0, 0, 0, what will happen? You will take the upper path, 0.

Student: Yes, sir.

You will take upper path; upper path; go to 0, 0, 0; take the bottom one; 1, 0, 0; you will come here; 1, 1, 0, you will come here; you take any path. The bits or digits, which are written at the output port, use them in a sequence. You will always get the output port address. Same thing, I can do it here. I need not even, remember everything. It is an inverse Ben's network. Ben's network is also, a self routing network, sorry, inverse forward Ben's what you call; Ben's network and also, inverse Ben's; both are self routing matrixes. I will make some changes and I think, from there, you should be able to observe the property, which has to be followed by a delta network, because that property is very crucial for the property of self routing; I will make some change here.

Student: Sir, this is delta network, stroke inverse Ben's (( ))?

It is always, a banyan network, because there was exactly, one path from each input to each output.

Student: Yes.

But it is also, a delta network, which is, because the routing is happening; self routing.

Student: And also known as inverse Ben's.

Inverse Ben's, because of our topology. So, every delta network is always, a banyan network, and every inverse Ben's network is always, a delta network, and every delta network is always, a banyan network, but reverse is not true. Every banyan network need not be a delta network, and every delta network need not be inverse Ben's.

Student: The reason for last statement?

Every delta network need not be inverse Ben's network

Student: But some delta networks can be based at (( ))?

Every Ben's network is a delta network, but the reverse is not true, all the time.

Student: A Ben's is superset?

Delta is superset of Ben’s network. There, Ben’s is only one configuration; there is only one topology. In this case, I can build many. In fact, there are already two topologies, which exist; I am only showing one of them, which is given in this paper. There is another one, which is given by Piece and Lorry, we call them; there were two separate networks, but they were proven to be same equivalent networks. In fact, honestly speaking, I can just change these maps. Based on that I can create what we call; all permutations, which can be configured. Permutation is what; permutation is, I am taking these eight inputs, and these eight outputs. I can create permutation or connections between them. So, one particular permutation, for example, is I can connect 0, 0, 0 to 0, 0, 0. So, this path is occupied. How many simultaneous connection maps can be done?

Now, it is very important; I cannot connect from 1 to 1; that is not possible in the permutation, because 1 to 1; there is no other path, which exist. If the one permutation is 0 to 1 and say, 1 to, say, you will be going to put a 4. Remember, in this case 0 to 0, 1 to 1 is not permitted. There, all possible combinations which can happen. So, this is one permutation, specific permutation of the switch, which I am writing now, sorry. One of the most important aspects, observation has showed; I cannot create a map from 0 to 0, 1 to 1, 2 to 2, 3 to 3, 4 to 4, 5 to 5; these possible combinations cannot be created.

Student: It is clear to you (( )).

Can you create 1 to 1 at the same time? This whole is one single set of permutation; this permutation actually, is not at all possible in this configuration.

Student: Sir, 0 to 0, (( )).

That is only a connection. Now, can I create this whole set of interconnections? Is it possible? This particular permutation is not possible. Total, how many permutations which can exist in the switch; can anybody guess?

Student: Eight square.

No, it cannot be we eight square. How you come to eight square? Tell me.

Student: The errors are eight into four. So, there are eight output ports.

Yes.

Student: With every input port, I can associate with any one of the output ports.

I associate each one of them with 1. 8 factorial is the correct answer, actually, because this guy can be connected from anyone of them. But, once the connection is made, next, I can only be connected from the remaining ones, actually, and so on. So, it has to be 8 factorial. How many permutations? This is a permutation between input and output ports. How many permutations are supported by this switch? Let us see if you can guess that. Every elementary 2 by 2 switch has only, how many permutations?

Student: Two.

Two. How many 2 by 2 elements are there?

Student: Four.

So, how many total permutations are possible?

Student: More than 24.

No, it cannot be 24; you should not add them. It is 2 raise power 12. See, each one of them can be independently, put in cross and bar state. So, this has two possible options; cross of bar. This has cross and bar; there is exactly unique path. So, 2 into 2 into 4 into 2; something, you put 2 raise power 12. So, exactly 2 raise power 12 unique combinations or maps, which can be generated from input to output.

Student: Sir, minus 1, there has to be for this?

For?

Student: For executing that.

Which set?

Student: This mapping; 0 to 0, 1 to 1.

That can be created; I am not saying that cannot be done. In this particular structure, it cannot be done; inverse Ben's, but I can even do a jugglery there, and do it actually. I am showing only three maps here. I can create another map here, without a switch, and then I can even create that. Now, which one of the number is larger; intuition, at least?

Student: 2 raise power 12

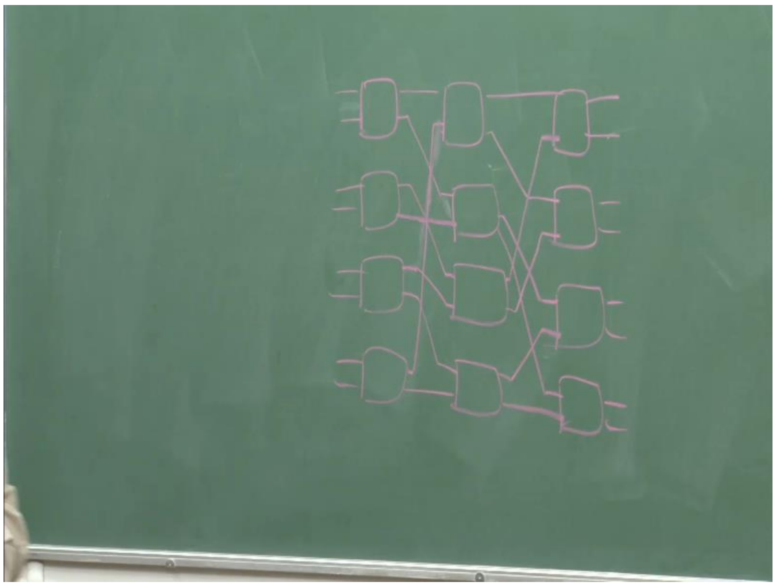
2 raise power 12 is larger. Is it possible?

But this is, I think immediately, why intuition you know, because all possibilities have been taken care of here. So, this can be either, smaller or can be only be equal, but this certainly, is not equal; it has to be always smaller, and is smaller by a very large number; not very small. You can verify this actually, if you wish. You explain this; find out how many 2s come in these factors and then compute how many, whether 2 raise power 12 is higher or smaller. In fact, you use a larger number of permutations, are not permitted in this case. In fact, the switch configurations are pretty tricky, in the sense that one particular switch, one which is given here; this covers up certain set of permutations. So, what is the difference between these two; those many possible delta networks configuration does exist.

Student: Difference?

Not difference; it is actually, banyan network switch exist, because I have not still looked into the; no, even in the delta network, those many possible delta networks exist, which are different from each other, which are non equivalent of each other, actually, whatever is the gap. But we do not know all of them; that is the problem. People using their own mathematical properties; they build up; this is one example of them. I am asking a question; how many possible such networks exist? My question is this. So, this has to be, because I am not covering all permutations. A switch can only cover these many permutations. So, whatever is the difference, those many necessarily delta network configuration must be existing, and many of them are unknown. So, you can do your PHDs for finding out the ones, which are unknown, which are not known, theoretically. So, this is one of them, actually. There is another one, which is given here in this. See everything is very simple mathematics; all are 11, 12 class kind of stuffs.

(Refer Slide Time: 49:43)



This is another one, which is drawn here. The problem is that for a 2 by 2, 8 by 8; you have done all the research. Who will do for 16 by 16? Then, who will do for; I can use the 3 by 3 basic element. So, the problems are actually, very pretty large. So, people try to just, build up only, what is called formalisms here; not the specific problem. So, this one, I am actually, drawing from here. Yes, this will be a good exercise for you people. I am just drawing it straightly, uniquely. I think you must be able to appreciate this. Now, are these two equivalent? The second interconnection is same; look at that between first and second stage; whether, you can prove it is same or not; you have to tell me.

Student: How we are checking the equivalence?

You tell me.

See, these are all graphic. Only just by visualization, you should take it up; otherwise, you cannot build up the mathematics of equivalence. Look at the map. Anybody, so far, is able to figure out what is the property, which a delta network should follow? Really, it is a whole matter of observation.

Student: Every inputs should be asked to go all the outputs.

That is true; that is the banyan network. So, this is actually, the precondition that none of these input ports have to be left open. None of the output ports of the middle stage have to be left open. All output ports of the last stage have to be left open; none of the inputs.

For the first stage, all inputs have to be left open; not the outputs. There has to be exactly, one path from every input to every output. It should be possible to connect every input to every output by exactly, one path; that is the banyan network. But for building up a delta network, if you carefully observe; the key factor, I can do; for example, a trick here. I just change this. Now, is it a delta network? I have broken the rule.

Student: Output (( ))

No, output status is still standard, but the tag value is not going to be same for the same output. It will be different at different inputs.

Student: Yes.

Now, why it happened?

Student: Mapping has changed (( )).

There is a rule which, I am following; I have broken that rule. The rule is all inputs for a switch in a stage, should come from the same output level number of the previous stage. Earlier, I was getting both the inputs from 1, 1. Now, I am getting 1 from 1. and another 1 from 0. I have broken this rule. See, what is happening is earlier, when I was connecting this to this, this was 1 and this was 1. If you again think, I think, you should be able to appreciate why this rule is important, because uniqueness cannot be maintained, if this rule is not there. Because for coming to particular switching element here, I require 0 in the d 3, and here, I require 1, which actually itself will make the output tags different for the same outgoing port, because this is connected to the same outgoing ports. They have to be same, and that is a fundamental property of the delta network. Create any dimensions; this rule has to be followed. Banyan network; this is now, is a banyan network; it is not a delta network anymore; it is a banyan network.