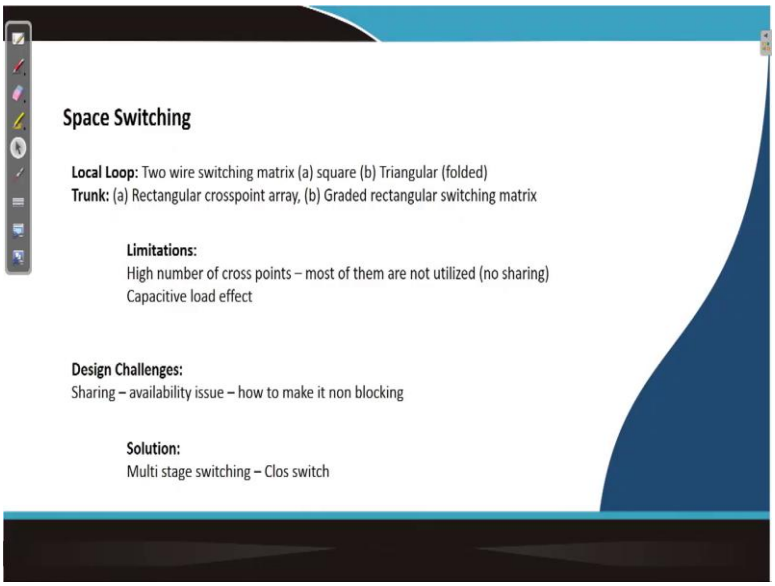
Communication Networks
Prof. Goutam Das
G. S. Sanyal School of Telecommunication
Indian Institute of Technology, Kharagpur

Module - 02
Circuit Switched Networks
Lecture - 06
Space Switching Architecture cont'd

Ok. So, in the previous class, we talked about Space Switching. And in space switching, we have tried to see that there are two kinds of space switching that has to be designed for telephony network; one is the local loop, and one is the trunk.

(Refer to Slide Time: 00:43)



And for the local loop and trunk, we have seen that there is this two-way switching matrix that we have designed; one is square, and one is triangular and folded one, ok. So, this is something we have seen for the local loop; for the trunk also, we have seen this rectangular cross point, rectangular cross point array, and we have seen this graded rectangular switching matrix, these two things we have seen.

The target was to reduce the number of switching elements. Why did we want to reduce? We have already discussed two points; one is the high number of cross points coming in those cases, so most of them are not utilized, which means no sharing is being done over there. So, for every one port to another port, there is a switching element, and it is not getting shared by anybody else.

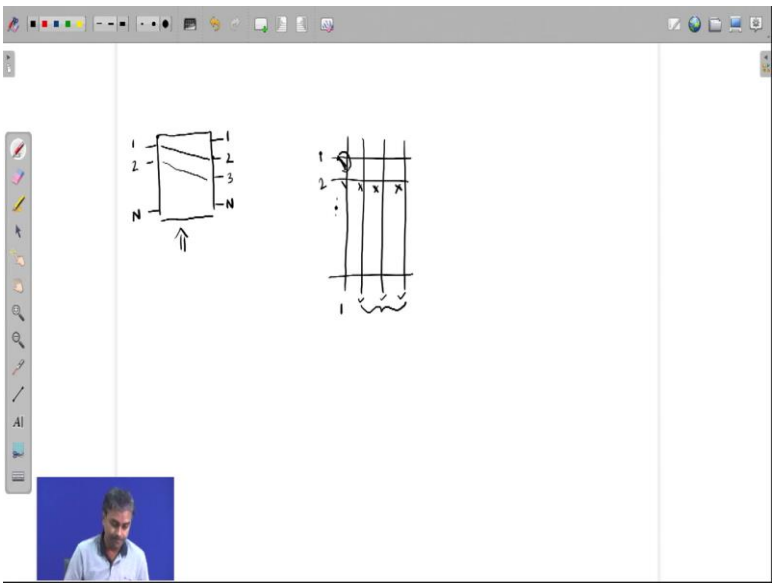
So, with this sharing concept, we will see how to bring that into our switching design, ok. And, of course, there is a capacitive load effect that you want to reduce. So, for that, we wanted to actually reduce the number of switching ports. So, what is the process of doing that? So, that is something we will see, which is actually our multistage switching, which is called clos switch; we will talk about that today in this class. So, over here, what we will be trying to do is, we will be trying to do this sharing.

So, this design challenge, if you see, we will be trying to do sharing of switching elements, which was not happening earlier in the cross-point switching. So, for every input to the output port, there is a dedicated switching, which means this cross point that was not shared by anybody else, ok.

We will see how this sharing is being done in this multistage switching, something the clos switch; actually, Clos was the person who invented this or who has proposed this, and we will see. But remember, we should not compromise on the switching performance; that means the term I am talking about in this design challenge how to make it non-blocking.

So, whenever we start sharing, there is a possibility that it is being shared by somebody, and I want to switch; I am blocked; this should not happen. See, I am not talking about it; let us go and discuss this a little bit. So, I am not talking about this input-to-output blocking.

(Refer to Slide Time: 03:05)



That means suppose I have a switch over here. So, and 1 to let us say N, they are trying to communicate to 1 to N. Now, let us say 1 is talking to 2 already. Now, if anybody else wishes to communicate to 2, he will be blocked, ok; that is alright; that is the output blockage. Output blockage is ok; if somebody is not available right now and he is already talking to somebody else, then I cannot communicate with him. This blockage is alright; this is not technology's fault; it is actually the guy whom I wish to talk to; he is already in conversation with somebody else.

So, this blockage, I have nothing to do; but if it happens like this; 1 is talking to 2, and because of the sharing I introduce over here among the switching elements; now, if somebody else, let us say, 2 wants to talk to 3, now here 2 is also free because he is not in conversation, 3 is also free.

So, from the perspective of the user, they are not blocked. Now, if the switch somehow blocks this connectivity, then that is blocking, and that is something I cannot allow; all our cross-point switches were not creating this thing.

So, we have to, whenever we start sharing, that is the danger; because some components or switching elements might be shared among multiple users. And if somebody is already talking to another guy through that shared port, now some other connectivity, where the input outputs are free; still the switch might introduce blocking.

Let me give you one example, one very simple example; that graded matrix we have seen, right? So, let us talk about, you know, of course, I am talking about a reduced matrix of course. So, let us say that is why I am taking that graded one because the graded one I am reducing.

So, suppose I give connectivity 1 and 2 to this trunk 1. And suppose I do not give any further connectivity, ok. If, in the graded design, I do this, then what will happen; now, suppose one wishes to get a trunk; I give this one this connectivity, this switch port. So, 1 gets trunk 1. Now, 2 want to have a trunk; these three trunks are free, but still, 2 cannot get because I have not introduced these switchings.

If these are not there, then 2 can only connect via this; but this is already occupied by 1; this is the blocking introduced by the switching matrix. Not that there is no availability; there are trunks available which could have carried data of 2, but I could not provide the

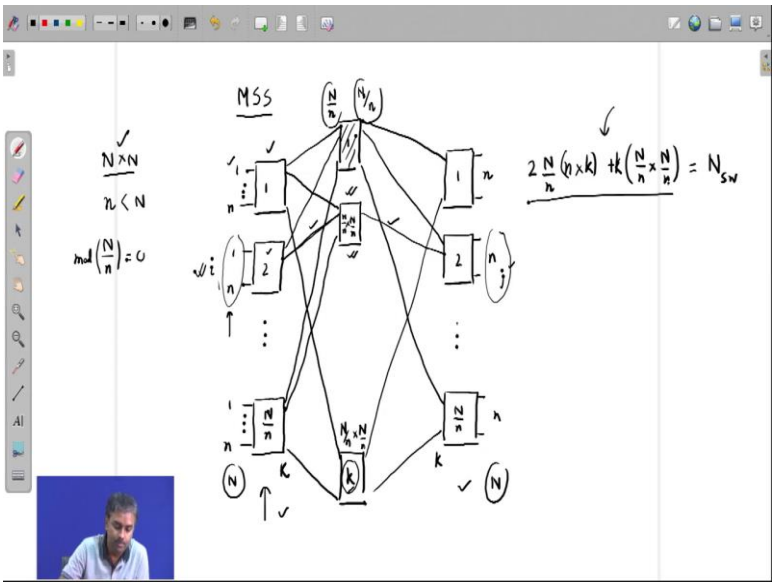
connectivity because, in the switching matrix, I wanted to reduce the number of switching elements, and because of that, what is happening now I do not get connectivity, ok.

You, later on, see this has happened because of the means improper reduction of switching elements; the same thing will be happening; we will see later on I will demonstrate that. If you start sharing, how do we do sharing? That is the first thing; so how do we actually, from this cross-point design, we would not be able to appreciate?

So, we will come to that structure, that multistage structure, where you can actually share switching elements. And once you share the switching elements, then we have to see that; because of sharing, somebody is already utilizing, and some more connectivity I need and input-output are free, whether we are capable of giving connectivity.

So, we have, while designing this shared architecture, we have to also make sure that enough switching matrices are given so that the switch becomes non-blocking by its own virtue; this will be the major design challenge for us, ok. So, if we just now go into a multistage switch, ok.

(Refer to Slide Time: 07:24)



So, this multistage switch, what is it? So, what I do, is suppose I have N connections N N cross N switching connections required. So, this was Clos's proposal that, if I wish to share, I need to make it multistage. What is that multistage, actually?

This N actually subdivides into smaller switching matrices. So, each of them is of length n; n is, of course, less than capital N and generally, N is divisible by small n. So, that means this N by n mod if I take, this should be 0, ok. So, capital N is completely divisible by small n, ok. So, this should be happening, ok.

What does this mean? That means, instead of having one switching element, ok, with the cross point of capital N; I will now have n switching elements at the input. So, at the input, I have n number of switching elements. So, I will be stacking multiple such as this one 1 to again n. Like this, how many will be required then? Of course, N is divisible by capital N is divisible by small n.

So, if I put this many, I will be getting an overall number that should be capital N ok; that is my input. And what will be the output? The output will be almost symmetric. Now, there will be small n output and similar things. So, again how many output ports will be there? Capital N; in between, in the middle, I will do the sharing; that is why it is called multistage.

So, it is generally whenever we talk about multistage; it will be this input and output will be like this, and there will be multiple middle stages, ok. So, you will see that the middle stage can start with a single middle stage. So, that is why it will become a three-stage switch.

It can be, from there, it can be again from 1 it can go to means middle stage can be 3, so it will be a five-stage switch and so on; we will see that structure later on, how to do that structure also, there is a generic rule of creating that structure. So, what now will we be doing, that sharing stage? So, we will create some middle stages; for the time being, let us create k number of stages, ok.

And how do we give connectivity? How is it being shared? It is like this. So, suppose this particular thing, one of the output ports will go to this switch; then let us know it is ok, and let us construct another switching element also. And what happens? His first port also will be going over there, and his first port also will be going over there.

So, how many ports will be having this middle stage 1 will be having? So, everybody's first port goes to his input. So, therefore, N by n number of input ports he will be having;

similarly, the structure is symmetric. So, I will be having all first ports getting connected to the output also, ok.

So, he will also have at the output N by n number of output; the same thing will be happening with all other things. So, this is being shared, ok; we will see how that sharing is being done. So, what will happen? This guy also the second port goes to him, the second port goes to him, and the second port of this goes to him. So, again he will have N by n, cross N by n, and so on.

Similarly, the kth number of ports goes to him. So, the kth port goes to him, and the kth port goes to him, ok. So, again this will be N by n and cross N by n; similarly, he will be also connected, ok. So, that is the switching matrix. So, what is the output port of this one? As you can see he has to connect to k switching elements, middle stage switching elements.

So, the output will be k, and similarly, over here, the input will be k, ok. So, what is happening? This middle one is being shared, ok; will come to that sharing concept, how this is being done, ok. And what will be the value of k that we will try to determine; we will try to see that if k affects the blocking or non-blocking part, we will give an argument given by Clos and, from there, will try to understand what should be the value of k, so that we have an unblocking switch.

This is something will; this is part of the criteria of the design also. So, so far, we have now understood the architecture of this multistage switch. Let us try to see how many switching elements are; after doing all these things, have we reduced the switching elements that something we have to see? So, let us try to see; for this N cross N switch, how many switching elements are required over here?

So, each one of them will assume that that is a square cross point or rectangular cross point, ok. So, if that is a rectangular cross point, how many switching elements are required over here, as you can see? In the first element, there are n input ports and k number of output ports. So, n into k number of ports are required for one switching element. How many such switching elements are there in this particular column?

So, N by n, this multiplied by N by n that many switching elements are required; this and this are symmetric, so this into 2 that many switching elements. In the middle stage, how

many switching elements are required? So, as you can see, each of them has N by n input and N by n output, ok. So, N by n cross N by n; How many such switching elements are there? K number of such switching elements k; so that is the overall number of switching elements we have, ok.

So, remember this equation; this is very important for us. And this is the whole thing that I will have to see whether I can reduce it and how I reduce it, ok. So, now, let us try to understand what exactly is happening over here. So, if I need to get connectivity, let us say some user 1 wishes to connect to some other user, ok, or let us take some ith user ok wishes to connect to some jth user over here, ok.

Now, first, you have to see that ith user block he is in, so that block you will have to identify, ok. And then, you have to see which output block the jth user is there, so that you have to identify. Now, from this block to this block, you need to keep switching. Now, this is the sharing part, the middle stage. I can choose any one of them; because each of these is connected to one-one middle stage, and each of these one is also connected to one-one middle stage, ok.

So, what am I eventually doing? So, let us say suppose I take; I choose this one. So, what will be happening? This is supposed the ith one I am doing. So, ith one is let us say, connected to the second switch, and the jth one, let us say, is connected also to the second switch of output. Now, from here to here, I have to do the switching.

So, what will I choose? If I choose this one, this element, the second element; then I connect to this one, and then that also will be connected to this, this port will be now occupied, I declare them occupied. Once this is occupied, now you can see none of the other users of this group or none of the other users of this group can further be connected via this switch; because this has only one output link and one input link to this middle stage switch.

That means if any other user from this user group has to connect, then it has to be via this one of the middle stages; once one middle stage is being taken, that middle stage is now shared, and it cannot further be taken by any of these users.

So, what we have seen understood that these users share the one-one middle stage; one middle stage is exclusively one of the users for these guys, and only one of them can

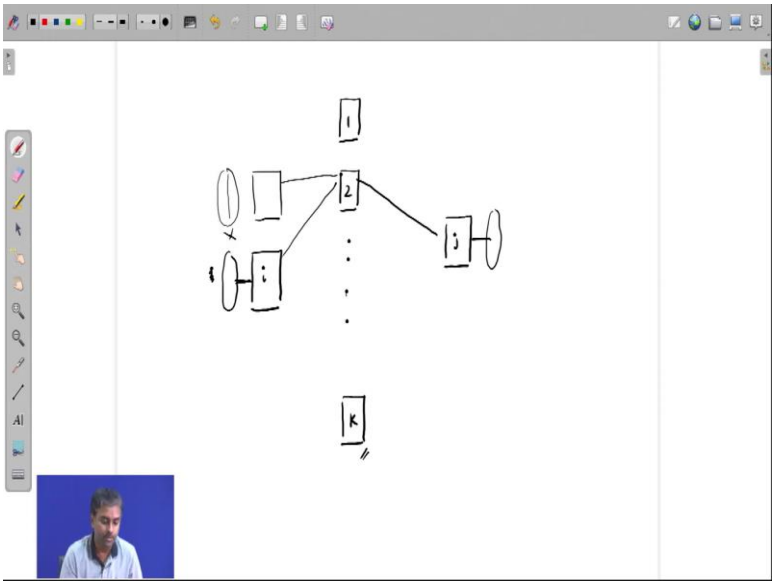
avail of this one middle stage; once that is occupied, any other user of this user group cannot take that particular one, he can take only other k elements.

Accordingly, we will have to see what should be the value of this k, ok. So, this is something we will try to; see with this sharing what is the effect of this sharing that is the first thing we will have to see and then we have to see; because this sharing is there a blocking condition, and if there is a blocking condition, how do I make it non-blocking, that is, the fundamental thing that will be now trying to discuss, ok.

So, for that, what you can now see is that the designing that I have to do is this k; because as I increase k, more on more elements will be there, more and more sharing means sharing elements are available, so I will have availability, ok. So, that is something that will be happening.

So, I hope this is clearly understood by everybody. So, now, what we will try to do, will try to design this as a non-blocking switch that something will try to do. So, if I just clear this ok and now, let us see how I design this non-blocking part.

(Refer to Slide Time: 18:59)



So, let us try to understand; let us say some ith user or some user let us say, is connected to the ith input block, and some user is connected to the jth output block; he needs connectivity, and there are k numbers of middle stages, ok.

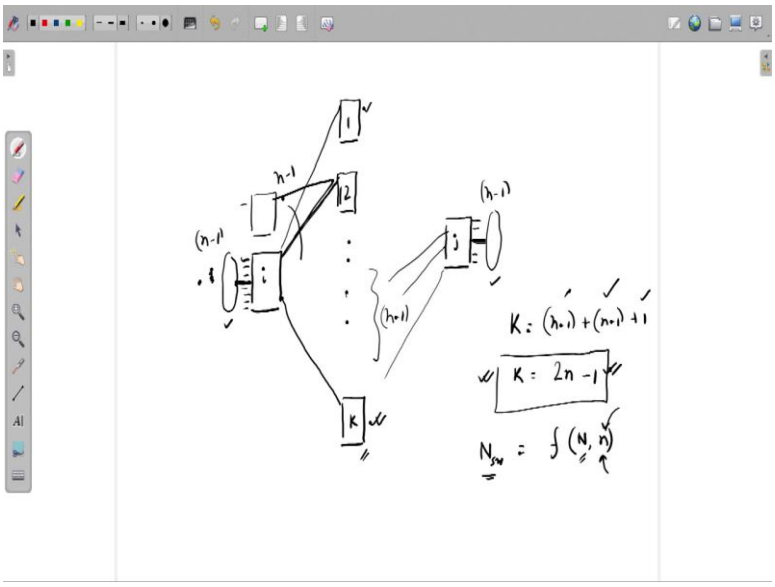
So, we have already seen the sharing nature. So, what does that mean? That a particular user connected to this ith block is willing to get connectivity to another user who is connected to this jth block ok, and there is k number of middle stages.

Now, I have to see that when my switch becomes non-blocking, what should be the value of k so that I can guarantee that whatever happens, it will always be non-blocking, ok. Let us try to first understand. So, the sharing is among these users, and among these users, that is where the middle stage is being shared.

So, if I just concentrate on this particular, because all other users from other blocks are not shared among these users, we have seen that because they have separate connectivity to the middle block.

So, suppose I take one connectivity like this, then some other blocks are there; they can still use this because they will have connectivity to this switch. So, they can still use it; even though this is being used, they can use it. So, I do not have any; all these users do not have on any conflict with these users. So, this conflict is resolved already because of the switching structure. So therefore, what I can do; I can forget about this portion, ok.

(Refer to Slide Time: 21:16)



So, now let us see; therefore, all I have to concentrate on is these users who are sharing and these users who are sharing. Now, when I want to mean to say non-blocking, I have

to think about the worst-case condition; at that point, also if a user comes with a request, can I still provide connectivity? Let us think about the worst-case condition.

So, what will be the worst-case condition? The worst case condition will be when the switch is maximally occupied; that means when I am making this connectivity, all others other n minus 1 are already connected, ok. So, what does that mean? It has k number of connectivity. So, whoever is connecting, they have to take one of these middle stages, and one user can only take one-one middle stage; because he has only one connectivity to this middle stage, each of these middle stages.

So, if there are n minus 1 other users who have occupied this, let us say they have already occupied n minus 1 such connections. So, if I just draw it that way. So, let us say n minus 1 is already occupied to accommodate the rest of the n minus 1 connections, which are already there; only one was free; that is my ith user actually, that is the user I am, means sorry that is not ith user, that is the user I am trying to now give connectivity, ok.

So, this n minus 1 is already occupied. Let us say this j also the same thing might happen. So, j what can happen? So, only one user he is targeting, this new connection is targeting; he also has n minus 1 other user, let us say they are also occupied. So, n minus 1 other users are occupied.

What is the worst-case scenario? Because I keep on doing this switching, so I can take any middle stage while doing the switching. So, this is the most unfortunate case that can happen is this j all these n minus 1 other users they are connected to some other blocks, and they have taken this n minus 1; other than this n minus 1, other n minus 1 blocks are occupied by this other n minus 1 user.

That is the worst-case scenario. At that point also, if this particular user comes and he wishes to connect to this particular user, both are free from the switching perspective; the input and output are free, so I need to give connectivity. So therefore, I still need to have another block free so that I can provide the connectivity.

So, in the worst case scenario, what will be the value of k, therefore? K must be this n minus 1 plus n minus 1; still, I need to have 1 more so that for the last connectivity also, I can give connectivity provide connectivity.

Therefore, for this switch to be non-blocking, my k should be this value. So, it is 2 n minus 1, which is the optimal value of k, so my switch is sure it will always become non-blocking. So, this part we have to see this very carefully; what is being shared is the first thing we need to understand.

So, what is being shared over here? Actually, these users, for a particular input switch block or for a particular output switch block; these users are sharing the middle stage switches of one of them because they only have one connectivity. So, if one of them takes it, others will not be able to use that middle block.

So, these middle block switches are being shared among the users of a particular input block switch, ok. So, users from here and users from another one are not actually sharing anything because they can still get connectivity. So, this guy can still get connectivity, and this guy can get connectivity through this switch only, so that is not being shared.

What is being shared is all the users of the same switch block, input switch block, or output switch block; they are sharing these middle stages, ok. So, when they are sharing this middle stage, we have to make sure that that sharing is proper. So, to do that, we have to see that in the worst case scenario also, I still have enough middle block so that I can give a non-blocking connectivity.

So, what is the worst case scenario? That is what we have done; this is the Clos argument to design that middle stage. So, basically, what he has argued is all n minus 1 in both the switch, input, and output switch might be occupied. And the unfortunate or worst-case scenario will be this n minus 1 will be using some n minus 1 middle stage, and mutually exclusively, other n minus 1 will be used by these guys.

If one of them is common, some less number switch will be required, ok, because we are trying to provide the worst case scenario, where still we are trying to provide non-blocking. So, I need to make sure that even if this n minus 1 and distinctly different n minus 1 are used, still for this last connectivity, I must have a switching element, a middle stage switching element.

So, that is when we come up with this number k equals 2 n minus 1. Once we have this k equal to 2 n minus 1, we know that it is strictly non-blocking; whatever happens,

whichever way you give connectivity, it does not matter; it will always be switch-wise, and there will be no blocking; this is ensured by this argument.

So, you can keep picking users and keep allocating elements; you do not have to see how many in the input I have given, which I have given in the output; so all those things you do not have to see, you can randomly give; even in the worst case scenario also, you will still have the last block available for giving connectivity to the last user.

And this will be true for every switching element; as you can see, they get separated out, and they are getting separate connectivity to the middle stage. So, the middle stage is differently shared; no problem with that one. So therefore, for everybody, as long as k is equal to 2 n minus 1, you have this non-blocking characteristic or non-blocking structure being ensured.

Once you get this k equals 2 n minus 1, then you remember we have talked about the overall switching elements; there, we had this value of k, which you will be replaced with 2 n minus 1. So, then the N switch will be a function of just capital N and small n, where capital N is something that is input; I cannot mean do anything with that, but small n is something that we can actually tinker with a bit, then this switch will be optimizing with respect to small n, this is something we will have to do.

In the next class, we will try to see what should be the optimal value of this n so that I get a very, which means a strictly non-blocking switch matrix, where the switching elements will be minimal possible. And that minimal number also, we will try to characterize and how it is different from a completely, complete matrix cross point switch ok, so square matrix kind of thing. So, that is something we will see in the next class, ok.

Thank you.