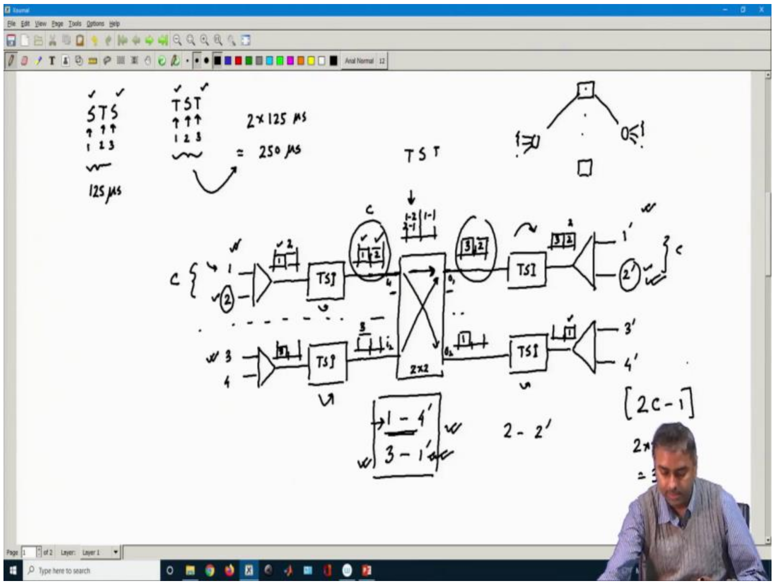
Communication Networks
Prof. Goutam Das
G. S. Sanyal School of Telecommunication
Indian Institute of Technology, Kharagpur

Module - 03
Circuit Switched Network
Lecture - 12
Space-Time Switch cont'd

So, in the last class, probably, we started discussing this multi-stage Space Time Switch combination. So, what will you do? This particular class also will be discussing the same thing we have left over a particular configuration of TST switch.

So, we will continue on the blocking performance of that kind of switch and how do I design that switch, do that same thing over here. So, let us go back to the design diagram that we have drawn in the last class.

(Refer to Slide Time: 00:55)



So, basically, what we have done so far is we have given 2 connectivity, and that 2 connectivity how we have configured that also we have mentioned over here. So, basically, we have connectivity of 1 to 4' which is something we have shown already. How do I do that what will be the exact configuration at the first TSI?

So, first, TSI, I do not move anything. So, I keep it as it is, then I do cross connectivity in the first slot 1 to 2, and then I actually swap it in the next one, and I get to 4'. In the second one, 3 to 1', I want to connect. So, first, TSI means randomly choosing means it is it should not be on means I should not see what has happened ok.

So, I should be able to do switching whichever way I wish to. So, that should be the motto of switching. So, basically, I choose to swap it in the first one. So, I put it in the second slot; in the second slot, I give a means cross connectivity ok, so that means it goes over here in the second slot and then I do a further time swapping. So, the third one comes into the first slot, and it goes out from 1 dash. So, this is what we have already configured.

Now, for the blocking case. So, now, I want connectivity between 2 to 2'; let us see whether I can do that. So, 2's data will be over here. I have no option for this, and if 2 to 2', I have to connect. So, 2's data should come over here in the second slot only.

So, what option do I have at the TSI? Because 1's data is already occupying the first slot after TSI, so I must put 2 over here. So, there is no other way I have to do it. So, if this is the case, what I have to do, is in the second slot, I have to give this connectivity, but that will collide with this data?

I have no option now. So, I cannot do switching from 2 to 2' even though these 2 ports and 2' ports are free. I have somehow blocked myself because of this connectivity. You might be asking if maybe can we do something again like the rearrangeable of the non-blocking space switch we can do something. So, the second connectivity we have made, we might rectify that. So, what we can do we might rectify this connectivity ok.

So, this is something we can rectify. So, 3 was over here. I could have put my 3 over here and then, at the first instance, what was the switching option. So, 1 to 2 ok. So, if 1 to 2 I can do, I can also do this cross.

So, if 1 to 2 is connected, I can always do 2 to 1 nobody blocks me because that is the cross connectivity I can always do that one input port to another distinct input port the mapping is being happening. So, I can do that. So, 3 will come over here.

And then at the TSI, I do not do anything. So, 3 come over here, 3 goes out from here. No problem, if I do that switching now, you see I have the option of the next connectivity 2 to 2’ if I want to connect. So, 2 comes over here, 2 I keep it over here because 2 is free now. Now the second is because of this connectivity actually I do not need this connectivity. This was gone. So, now, at the second instance, I can keep connectivity like this 1-to-1 connectivity. So, in the bar state, I connect the switch.

So, 2 goes over here, I keep 2 over here, and 2 goes out from here. So, as you can see now, just by reconfiguring the previous connection, so, this is called a rearrangeably non-blocking switch I could take out that blocking state ok.

So that means, I can randomly whenever a connection comes, I can randomly give connectivity, I can adjust TSI, I can adjust space switch configuration, and I can give connectivity randomly. So, at that point whatever looks best to me like in space switching also we were doing, we just do that connectivity.

But whenever there is a blocking situation, I try to see who is actually blocking. So, as you have seen, this connectivity is 1 to 4’ and 3 to 1’, these two were actually blocking this 2 to 2 dash connectivity.

So, again with the clause argument, as I have told you, you will see inputs that are actually disturbing him. So, 2 to 2 dash connectivity at the input 1 is the disturbance creator. So, this connection is 1 to 4’, and at the output, it is 2’; so, 1’ this connection was creating a disturbance.

If there are multiple c’s, so all of them. Try to see those connections and rearrange those connections. So, that you get a free slot over the entire switching matrix and you can provide connectivity, this is something you can do. So, this is called rearranging non-blocking, but can I now talk about how Clos has given that argument?

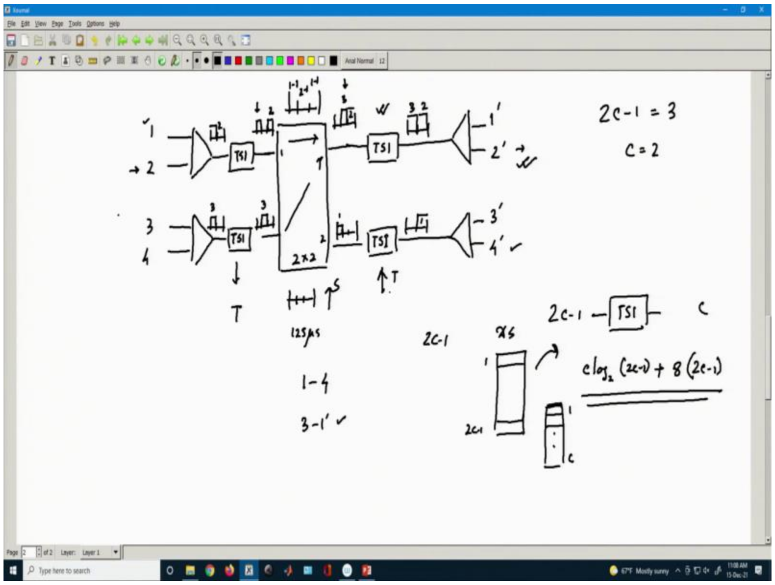
Can I now talk about a non-blocking switch, strictly non-blocking I do not have to rearrange anybody I can give any random connections as long as those connections are viable? I will still be able to give connectivity to the next one that is coming to me, ok.

So, this is something that I have to try to do; let us see whether we can do this. So, for that I need you to remember in the sharing stage. So, where exactly is it getting shared? As you have seen, it is actually getting shared in the middle stage. So, this 2 to 2 dash I am talking about that is actually sharing this middle stage almost like that one of the stage middle stage they were sharing that is where it is getting shared ok.

So, you have the because 2 to 2’ you have to give connectivity; that means, whatever happens, they must be because 2 dash is associated with this TSI, 2 is associated with this TSI. So, it must be occupying some of this first port of the space switch. So, in the first port of the space switch, what are the sharing options? We are sharing the slots so, probably why it is getting blocked is because there are not enough slots.

So, I have this c number of input things and c number of output things which are sharing those slots ok. How many slots do I have, c that is not sufficient? If I can now give the similar argument as Clos has given 2 c minus 1 number of slots. So, over here, my c is 2, so it should be 2 into 2 minus 1, so 3 number of slots; if I can provide, I will probably see a non-blocking switch. What does that mean? How do I do the connectivity? Let me draw a fresh diagram then it will be clear, probably.

(Refer to Slide Time: 08:33)



So, let us try to see let us quickly draw the diagram. So, now, all I have to do is in the middle stage. So, you remember we had these 2 slots all synchronized; in between, we have put 2 slots. So, within these 125 microseconds now, we will be putting actually instead of 2 3 slots because my 2 c minus 1 is 3 now if c is equal to 2 ok. So, I put 3 slots over here at the space switch output also, I put 3 slots. So, basically, the space switch now has within 125 microseconds 3 switch switching options ok.

And then, from there 2 slots ok. So, the TSI is a little tilted now, it is not symmetric. So, at the input of TSI the data slots that are coming that is 2 c minus 1 and at the output of this TSI 2 c slots are going out which means, in 125 microseconds, I am reading it faster because 2 c minus 1 over here it is 3, 3 data has to be read whereas, while writing I will be writing little slow. So, I will be writing. So, this is sorry, this is c ok.

I will be writing c number of bytes, ok. So, 2 c minus 1 number of bytes are coming in. So, while writing I have to write it faster, while reading at the output I will be reading a little slower with the c number of bytes in 125 microseconds. So, that is the TSI we will be designing now.

At the same time, this TSI is also tilted, it actually has a lower writing speed ok and faster reading speed over here c with the c bytes coming at 125 microseconds within 125 microseconds and 2 c minus 1 are being fetched ok.

So, what will be happening a few things which will be happening now one is this TSI how many locations it will have? This TSI now, TSI memory will be a little bit more because whether you read or write, you have to keep data storage up to c minus 1. So, earlier, it was c. Now you have to keep data storage up to c minus 1.

So, TSI will have data storage which is 1 up to c minus 1 ok. So, that much data storage will be there and then depending on the input associated and output sorry random write and sequential read or sequential read and random write.

Depending on that, what will be your memory these things will be dependent on that, ok. So, where you will be fetching that is something that will be dependent. So, now, let us try to see if we are fetching them. So, means if we are trying to write them so that means, what is happening whatever comes.

So, let us say I am designing this switch ok. So, 2 c minus 1 data are coming I am just storing them one after another. Now there is a control, ok. So, in the control, what I have to do is, I have to how many bytes should be there in the control because while actually reading them, I only need c number of locations.

So, therefore, there will be a c number of locations, but for each location, how many bytes have to be or how many bits have to be there because each location should be specifying one of these 2 c minus 1.

So, that should be log 2 c minus 1, ok. These many bits are required, and there are c number of locations. So, c into this plus 8 into 2 c minus 1. So, that should be the memory requirement of these TSI switches now, as you have understood. So, the TSI switch gets a little bit tilted, but let us now try to appreciate what this has given me. Has this given me an unblocking case? So, earlier if we see. So, we had a configuration of let us go back to that.

So, we had a configuration of 1 to 4' and 3 to 1'. So, 1 to 4' and 3 to 1' ok. So, let us try to do this 1 to 4'. So, the way we have done earlier. So, 1 this 1 to 4', we will have to do so once data comes, now I have 3 options anywhere I can put ok. Let us put it in the first slot, ok. So, I put it over here remember the space switching cannot do time readjustment. So, if the input to the space switch is 3 slots then the output also should be 3 slots.

If it is 2 c minus 1 output should be 2 c minus 1. So, if I put it in the first slot at the first slot of this configuration, I can do this 1 to 2 configuration, and I can put it over here. So, the data of 1 comes over here, ok. Now I have to put it in 4'. So, I do this swapping once data comes over here that comes from here, good, very good, no problem with that. Now the next one, 3 to 1 dash. So, 3 3's data comes over here, so that is 3. Now over here, I might choose to swap it.

So, let us say I put it in the second slot. So, that is where my 3 comes in. So, 3 then should go over here. So, it will come in the second slot because the space switch cannot swap time. So, I have to do 2 to 1 connectivity in the second slot because in the second slot only I have to push this data to this 3 to 1'. So, again this has to be swapped, and I put it over here that will go out from this one. Now suppose I want to give this connectivity 2 to 2'.

Now, you can see it is very nice because whatever I do, I have occupied the first slot, and at the output, I have occupied the second slot, I still have the third one remaining. So, this was clause argument that when I am sharing this middle stage only I am sharing, if I have if there are c number of this one and c number of this one in the worst case scenario all other c are coming c minus 1 are coming from the others, and this one all c minus 1 are connected to some others which are happening over here ok.

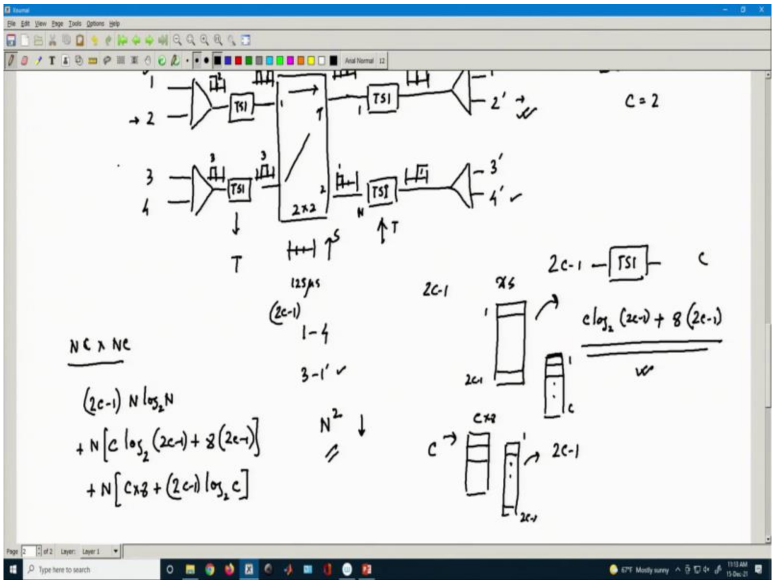
So, 1 to 3' from others and 4 to 1' from others, it is coming to this, and then still, I am trying to get the connectivity if I have 2 c minus 1, which is 3 over here. So, I still get the last one to be filled up, and do I can do the corresponding switching. So, I have a strictly non-blocking switch. So, now, 2 to 2', I can put 2 over here, and I can put the 2 after time switch interchange over here if I do that, I can do this bar connectivity at the last one. So, 1-to-1 connectivity I can do because it is free.

So, my 2 comes over here, 2 I put over here all are free, and it comes out from 2'. So, this is how I can still make a rearrangeably, sorry, a strictly non-blocking switch if, in the middle stage of TSI, I inflate. So, inflation is TSI switch will now no longer have a symmetric input to output.

So, if it has a c number of slots coming in, I am now having 2 c minus 1 slot going out, but things are all getting symmetric because, on the other side of TSI, I will be just doing the reverse thing. So, from 2 c minus 1 in, I am getting c out because of this middle-stage inflation or expansion of 2 c minus 1 I have all the possibilities.

So, whichever way now I do switching it is strictly non-blocking I do not have to rearrange anything I can get any kind of switch connectivity blindly if I keep allocating things. So, this is the beauty of a particular TSI or TST kind of switch. So, first stage T, second stage S, and third stage T. Now, if you ask me what is the overall amount of memory that is required, can you now do this calculation let us try to do this calculation ok.

(Refer to Slide Time: 18:28)



Let us see what is the memory requirement in the space switch what is the memory requirement? Let us say I am doing N C cross N C switch with 3 stages let us say in the space. So, now, how many slots are there? As there is an expansion so there will be 2 c minus 1 slot.

So, for each of these 2 c minus 1, what is the control logic that is required? Control logic because it is an N x N switch. So, N log 2 N is the control logic that will be required, and that is required for 2c-1 slots because there is a slot expansion.

So, each slot must have a particular associated control logic memory that has to be stored. So, that I can do the control at the particular instance, things are all synchronized. So, I have no issues every slot means every time slot boundary I go, I do that switching, and the next slot I fetch from memory, I do that switching. So, I keep doing this kind of switching.

So, from I fetch data from that control and fed it to the multiplexer accordingly the switching is being done ok. So, it will keep on repeating every 125 microseconds. So, these many memories are required for doing the space control.

Now, let us talk about the TSI this second stage of TSI, you have already seen that 2 c minus 1 number of slots are coming, and I am reading c. So, for that, I have already calculated. So, it should be c log 2 2 c minus 1 plus 8 into 2 c minus 1. How many such TSIs are required? 1 to N. So, N number of such TSIs are required. So, that is the memory that I will be requiring. Now at the input stage so, input stage what is happening oh, sorry, this is the; this is the memory requirement and data requirement.

Now, this input stage switches a little differently because over here, what is happening c the number of bits will be coming in, and I have to read this 2c-1. So, now, let us see how much data storage I require. I require a c for each one of them 8. So, c into 8, right? So, N number of TSIs are there c into 8, that will be the requirement. Now while reading, I still need this whole thing right I have to read a particular these things. So, let us try to see how many such memory requirements will be required.

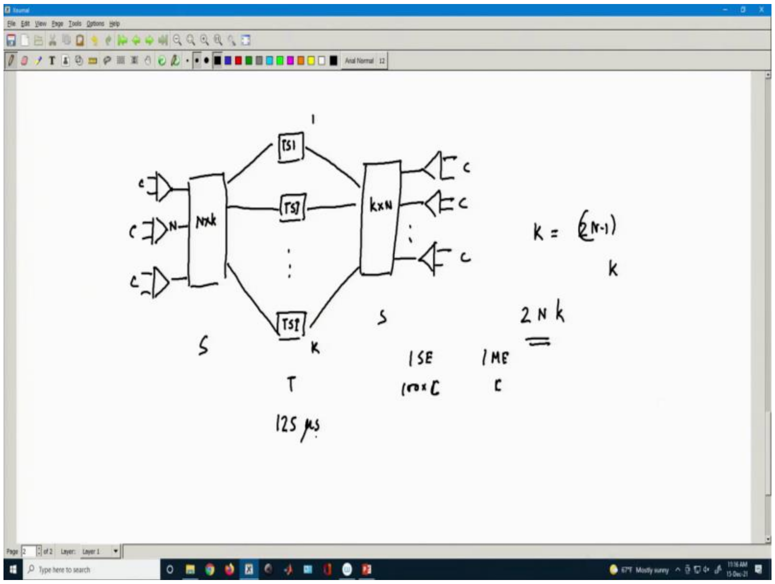
Because now I have to distinctly read 2 c minus 1. So, that requirement will be 2 c minus 1. So, there will be a 2c-1 memory place. So, 1 to 2 c minus 1. So, 2 c minus 1 memory place will be there, and for each one of them, I need to specify which location it is there. So, that is, there are only c locations. So, log 2 c and N such because there are N TSI.

So, this will be the overall memory requirement of this kind of complex switch; once I have that: What is the overall delay of this switch? It should be 250 microseconds because for every switching that you do, you need to go through the first TSI, and the last TSI you have to do it. One stage of TSI and followed by the third stage of TSI you will have to do that. So, therefore, 125 microseconds in 2 so, there will always be 250 microseconds switching this one time or switching delay that will be happening always.

This is the overall complexity; that means, this is the overall memory requirement which is the space requirement or switching blocks requirements. So, N x N switch I have done if I do not do any multi-stage in the middle stage also. So, it will be N square which is the requirement. I can further reduce it if this space is switched again I do a 3-stage or 5-stage or 7-stage 9-stage switching then further this will be reduced ok.

So, that is something I will be able to do. So, this is possible. So, this is something which we can also do. So, now, you have seen how I do this strictly non-blocking, rearrangeably non-blocking 3-stage switches with the combination of space and time switches. So, we will later on also talk about this STS switch.

(Refer to Slide Time: 23:29)



So, let me just draw a quick diagram for the STS switch. So, that first stage will be space, the second stage also will be space, and in between, there will be TSI. Now again, if we just so this is the space stage, now let us say its N stages, of course, they are followed by a multiplexer of c each. So, it is a dimension of N c, and then middle there will be a 1 to K stage. So, this K, again, you will be able to prove that if I wish to do strictly non-blocking, this K has to be whatever that number is there, let us say N.

So, it should be 2 N minus 1. So, that has to be there. So, this is something we will be able to again prove if you wish to do a strictly non-blocking switch because these are getting this middle stage are getting shared.

So, you can do this kind of space switching. So, that is STS switching you can always do that of course, this would be N cross K space switches and K cross N. So, basically, the amount of space switching will be much higher because it will be 2 N into K if your K is 2 N minus 1.

Then that is the switching dimension that you will be requiring. So, this will be much higher. So, space switching elements that will be required will be much higher over here of course, you will be able to see that the memory requirement will be less, but of course, memory is much cheaper than the switching elements.

So, there is a rule of thumb that every memory element is 100 times cheaper than the switching elements. So, if you combine these two. So, if there are one switching element and one memory element, the cost of that will be this switching element will be 100 times more costlier. So, ok.

So, for some unit cost, if I see that it will be 100 c, this will be only c, one memory element, one memory element, and one switching element. So, this is the cost equation that will be happening, ok. So, accordingly, if you try to minimize the cost, you will have to see it in this fashion, but of course, over here, there is one advantage because only one TSI you are doing. So, it will always give you 125 microseconds delay, it will not be giving you anything more than that, ok. So, that is something you can do.

So, now what we will try to do in the next class is try to see if we can extend the blocking probability calculation that we have done through these graphs to these things also. It is very easy; we can actually do that. So, now, we will also try to evaluate the blocking probability of these kinds of switches which are space-time-space or time-space-time or just TS switch.

So, that those kinds of switches, we will try to see what I do for evaluating blocking probability. So, Lee’s graph essentially can be applicable similarly over here that something will be appreciating.

But next, we will try to do another switch that we have not talked about so far is a trunk switch. The blocking probability calculation of the trunks switch is not as straightforward or as easy as this one. So, we will try to see if there is a very rich literature on queuing theory that will be borrowed to evaluate the blocking probability of trunk switches, and those are famously known as Erlang B or Erlang C formula.

So, we will try to appreciate what kind of minimal knowledge of queuing theory we can introduce. So, that can be applied to evaluating the blocking probability of trunk switches.

These are actually typical local subscriber loops or local circuit space switching that we were appreciating so far. So, we will now try to see trunks switch where the output can be anything we, it does not really matter that we have to go to a particular certain output port output can be anything. So, they are like servers; any of that servers I can take, whichever is available. So, that is trunk switching for you, and we will try to see whether we can evaluate the blocking probability of that ok.

Thank you.