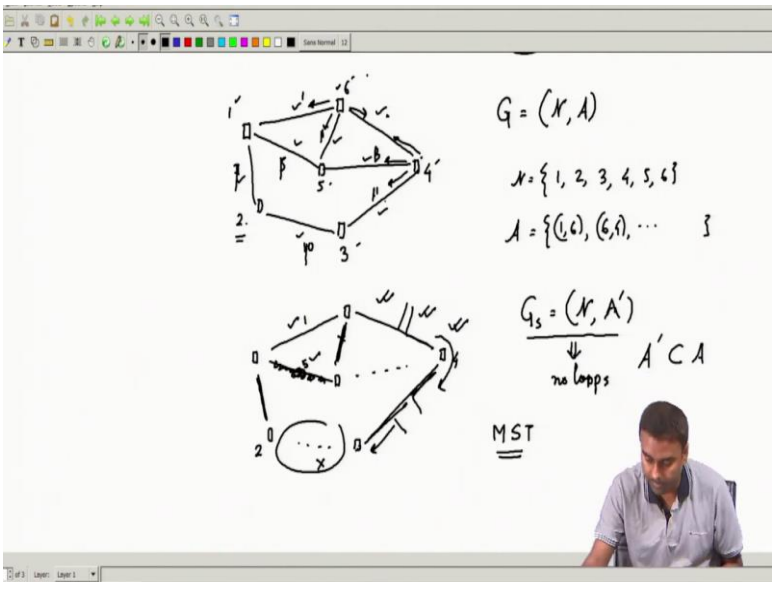
Communication Networks
Prof. Goutam Das
G.S. Sanyal School of Telecommunication
Indian Institute of Technology, Kharagpur

Module - 11
Bridges
Lecture - 52
Distributed Spanning Tree

Ok. So, now, we have already discussed the problems of learning Bridges, if they are connected in a loop. So, for that, we have to facilitate the bridge with an algorithm, which is called the Distributed Spanning Tree Algorithm.

So, we will try to see how this distributed spanning tree algorithm has been devised. And how the bridges in a distributed fashion learn by themselves what should be the corresponding spanning tree, ok. So, let us try to understand the definition of a spanning tree.

(Refer Slide Time: 01:03)



So, spanning tree is, if you have a kind of mesh network, ok. So, take particular network connectivity, ok, they have some connectivity where which is not loop-free, ok. Let us say this kind of connectivity we have. It has a loop, multiple loops as you can see. So, now, if we wish to make this to be loop-free, we have to also span the entire network;

that means, suppose this is a particular graph we have taken, it is of course, a network, so, its a graph, which is a graph is nothing but, some set of node and some set of arc, ok.

So, let us say I name them 1, 2, 3, 4, 5, 6. So, that is the Bridge IDs let us say and the connectivity between them. So, this node set will be nothing but, all the nodes. So, 1, 2, 3 4, 5, 6. So, those and the arc set will be all the connectivity between them. So, like links 1 to 6, ok this one. The links 6 to 4 and so on, all the links you list, ok. So, every link 1, 2, 3, 4, 5, 6, 7, 8 all 8 links. So, 6 nodes and 8 links make the whole graph, ok.

Now from this graph, we have to construct a tree structure where there is no loop. But, we have to span the entire network; that means, every node means, whatever link we will be taking, all nodes should be there and they must be connected via some links, ok. So, if I take it this way, let us say I take 6, 1, 2, 3, 5 and 4 and I give a connectivity like this, I take this link I take this link I take this link this link and this link. So, that is a spanning tree.

So, why it is a spanning tree? As you can see, I have taken a subset of this graph, where this subset is all the node-set should be there, the entire N should be there and some A dash, where this A dash is a subset of A. So that means, the sum of the subset of the links I have chosen such that, in this new graph there are no loops. If this happens, then this is called the spanning tree of the other network or other graph.

So, that is what we will have to construct. My algorithm should now construct a spanning tree, from this particular graph; that means, it should span the entire network. So, all the nodes should be included, but I must take a subset of nodes. So, all the nodes are connected as you can see this node is connected; from any node starting from any node, I can reach other nodes via all those links.

So, basically, it is a connected graph. If I had not taken this one, then it is a disconnected graph, that is not a spanning tree, ok? So, it must be constructing a tree structure. So, everybody is connected to each other, but they mean and also we should also ensure that all the nodes are taken into account, then it is called a spanning tree.

Now, among this spanning tree, we can also have a concept of Minimum Spanning Tree MST. So, Minimum Spanning Tree means, if now these links are having some cost value, ok. So, if I just say this is 1, 2, 3, 1, 5, 7, 10, 11 something like that. If I give the

cost value, now I have to construct or now the minimum spanning tree is a unique one, ok, in most of the cases that will be the case there might be some tie, but it will be some set of links which will which I will be taking.

It will still construct a spanning tree, but overall link cost, and added link cost will be minimal among all possible means, all possible spanning trees that I can construct. Because over here as you can see, multiple spanning tree I can construct. Among them, the link cost should be minimal.

Suppose, if you see, I can also have a spanning tree like this. Instead of taking this link, I could have taken this link and this link I could have taken out, ok, this is their assumption. This is also a spanning tree; because all the nodes are connected to this link only, I have taken it out, ok. If I do that, immediately you can see, that instead of link 5, I have now included another link which is costing only 1. So, therefore, this cost is much less than or at least 4 units less than the previous cost.

So, like this, a minimum spanning tree also can be constructed depending on the cost of these values, ok? Generally, there are some postulates which can be proven we would not be doing that over here, that how many numbers of links have to be taken to construct a spanning tree? That is always fixed. As you can see, as many nodes are there. 1 less link has to be there. So, a spanning tree will always have that many number of links. So, that is always true, ok.

So, always I can replace a link with another link because all spanning trees will have eventually the same number of links, link by link whichever is the cheapest. So, with the cost-effective or minimum-cost spanning tree, I will be able to construct. So, there are algorithms to construct these kinds of things. Now, you will be asking, what is this cost? What do I mean by the cost of the link? So, the cost of the link might be, the actual cost of the link; suppose you have leased this link, ok or this might be the lease of that LAN using that LAN. If you try to forward data using that LAN, you might have to pay some cost.

So, it might be that cost, of course, it is the economic cost. It might be other things also; cost might be inversely proportional to the delay. So, in that LAN, suppose a huge amount of traffic is there. So, therefore, the delay is a little higher. So, you can make that

LAN the least costly because you do not want to include that in your network because that will increase your delay.

So, therefore, like that, you can devise the cost, ok? So, it might be inversely proportional to delay. Sometimes it might be, just the number of hops. So, every link costs only 1. If the link cost is all link costs are 1; that means, they are all uniform. So, therefore, the minimum spanning tree will be equivalent to any spanning tree. Because a number of links are always fixed in a spanning tree.

So, you will be taking only this many links and then anyone you take, you replace this one with this one, it is still the same cost because each of the links costs 1 unit only, ok? So, how many links you are taking? If that is the cost of each link is costing only 1 then any spanning tree is a minimum spanning tree, ok.

So, the cost is dependent on how you decide your cost. If it is related to delay, then separate costs might be there. If it is an economic cost, then a separate cost might be there. If it is just hop count, then only one cost unit per link will be coming into the picture. So, depending on that your spanning tree algorithm will be devised. So, whatever the cost is, that is something which will be a metric which will be given.

So, for every graph, each link will have it is associated cost. So, this will be supplied to you and with that supply you will be now deciding about your cost. So, the example for the spanning tree algorithm that we will be devising, for the time being, of course, any generic cost we can take for the time being we will take hop count as the cost, ok?

But it can be generalized to any other cost factor, ok. So, now, let us try to see, how we devise this particular kind of protocol. So, that should be our next target. So, for devising this particular kind of protocol, you need to remember that there are 2 issues, one is this has to be done distributedly. So, basically, each of the stations must take its own responsibility for deciding these things and how do they decide?

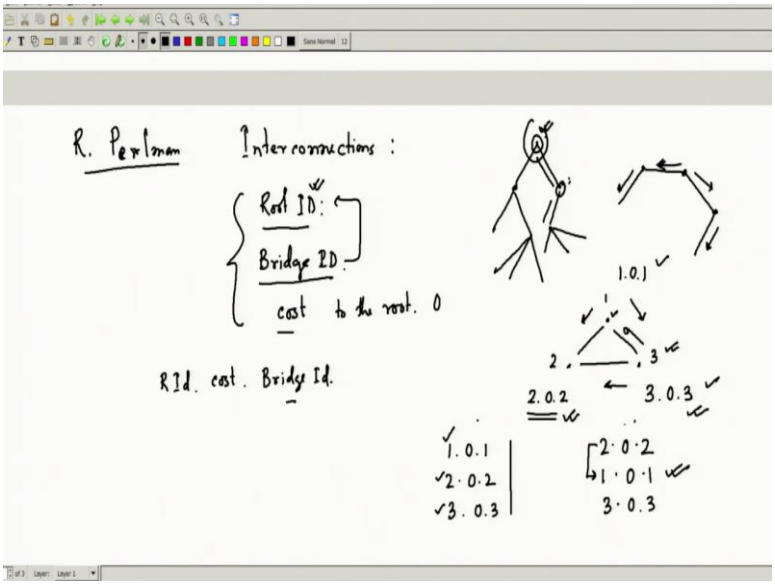
Because it is still centralized things. The spanning tree is spanning over the entire graph. So, every node of the graph must know which link to be activated, ok. But they are they should collaborate, towards finding a unified spanning tree. If 1 node thinks that this must be my spanning tree and other nodes decide another spanning tree, then they will be

clashing with each other. So, that will not actually be loop-free, then they will again create a loop.

So, therefore, they must collaborate also. So, it is a distributed decision, but it must be a collaborative unique decision, that must come out, ok. So, that is why they must be or there must be some provision of message passing among themselves. And with those messages only, they will be deciding or they will be running their algorithm and through that, they will be uniquely coming out with a unique solution, ok?

So, in a distributed manner, in a unified manner also, they will be coming to a unique solution that is what we want to see. So, therefore, we need to first understand, what kind of message they will be passing to each other. So, this the protocol that I will be describing that is of course, comes from the original centralized protocol of Prim's and Kruskal's whatever greedy approach in spanning tree you can take, because it has an underlying theory that any greedy approach is good for spanning tree algorithm or minimum spanning tree algorithm. So, it comes from that particular viewpoint.

(Refer Slide Time: 12:08)



And this was devised by Radia Perlman. And you can find it in her book only, the book I am following. So, it is called Interconnection Bridges Router Switches and Inter Networking Protocols, ok Perlman it is not a, its P e r l, ok, Interconnections Bridges Router Switches and Inter Networking Protocols. So, it is and she only has devised this protocol. So, this which has been in use. So, we will start describing what was the

message that she started thinking about and how those messages helped to build the whole protocol, ok?

So, each bridge locally has to just decide over here as you can see. Each bridge has to decide which node or which LAN he takes for forwarding the packet and which LANs he actually does not forward the packet, ok. So, over here as you can see this is the connectivity.

So, there are 6 bridges. So, 1, 2, 1, 6, 4, 5, 3 and 2 and all these 6 bridges are connected to all these 8 LANs that we have constructed. And among them, let us say bridge 2 decides, to utilize this LAN for forwarding, but not this LAN for forwarding. So, this is how they decide it, ok. Similarly, bridge 4, decides this LAN to forward packet this LAN to forward packet, but not this LAN to forward packet, ok.

So, they will have to just in a distributed fashion, they should make a unified decision, that is all we need to do. And they should make this decision on which LAN to forward and which LAN not to forward. So, whenever the packet comes over here, they will always decide not to forward it to a particular LAN, they will never do that, ok? So, their learning will also have this additional learning mechanism.

So, even though they have LANs in different ports connected, some of the LANs will decide not to forward packets. So, they will decide that these LANs any packet that comes I will never forward, I will never whatever packets are coming in that particular LAN, and I will never learn them either.

So, which stations are connected over there I will never learn that. Only the path or the LANs I have decided that this must be the designated LAN or this must be the LAN where I will be forwarding my packet there only, I will continue learning my things like the learning bridges. Because once it is a tree, he can start learning. So, this is a tree structure, this structure is a tree structure. Now he can learn.

He will be learning, see this particular LAN he will not learn, this LAN he will learn which stations are connected. From this LAN whatever will be forwarded through the bridge, will come to him he will also learn which stations are connected beyond that. So, all these things he will learn, but he will stop learning on those discarded LANs, ok?

So, this is what we want to facilitate and we try to see how this was done by Radia Perlman, ok? So for that, she decided there must be some messages.

So, what are those messages? So, the first message is called the Root ID. Whoever has done a little bit of data structure or whoever has done a little bit of graph theory, in any tree, you can always define the tree with a root and then all others are branches of that, ok. And then from root, it comes to a child and then from there next set of children and so on. The tree will never have a loop. So, it will always have branches. So, that is why it is called Tree.

So, basically tree as if like it has a root and from that root it has branches, again next child from their branches, some branches might be there some might not be there and so on, it just goes on like this, ok. Not necessarily a binary tree, it can have multiple branches also, ok something like that. So, that should be the tree. Always it is like this and there should be a root; from where it all starts, ok. Even a linear thing, that also I can decide to be a tree, these 2 are branches.

Then it comes over, here from there another branch another branch and so on. So, it just keeps on going from one branch to another branch or leaf to another leaf and so on, ok? So, therefore, there is a concept of root, I can decide and in a tree anything you can decide as root. So, I can decide this to be root and then from there, I can start taking branches.

So, basically, anybody can be root. So, this the Root ID, therefore, is a very important thing, which has to be encoded in the message, so, that is the first thing. Then, the Bridge ID. So, Bridge ID is whichever bridge is transmitting that message, because we are talking about what message they can pass. So, whenever somebody is passing some message at that time, he has a belief that this must be the root and this is my Bridge ID.

So, he is transmitting. So, therefore, he will suppose he is transmitting and he believes this is the root. So, he will give his ID as Root ID and his own ID as Bridge ID, in the message he will encode these 2 things. What are the third things? The third thing is the cost he has estimated. So, far to the root, ok.

In the message, what she has proven only these 3 things we take, it is good enough if they keep on updating these things in a recursive fashion. It is good enough that after

sometimes they will learn, locally and globally and they will learn the same thing they will come up with the same tree structure, spanning tree structure. So, let us try to see how it is being initiated.

So, basically every node has to pass this message to its neighbour. So, he will be connected via some port. So, let us say if this is the situation, then, he has to pass this message to all his neighbours. So on, he has to also pass this message to all his neighbours. Now, how he constructs that message and how he initializes this message, should be part of the algorithm, ok?

So, for initialization, every node will assume that it is the root. So, basically whatever his Bridge ID, that only he will put in his Root ID. Immediately, what is the cost? So, he is the Bridge ID; from him, he wants to reach the root and what is the cost, The cost should be 0, because he is the Bridge ID. So, from him to root, the cost is 0, because if you can reach him you can reach the root.

So, therefore, the cost is 0. So, that should be the initial message. Everybody has their own Bridge ID, ok. So, that Bridge ID, they take as Root ID and take the cost to be 0 and give it to everybody, ok.

Now, what will be happening then? If everybody is actually broadcasting this to their neighbour, they will be getting multiple copies of such messages from all their neighbours. So, let us say this guy; will generate his own message, with his Root ID his Bridge ID as Root ID and 0 as cost. But he will also get messages from this guy this guy and this guy.

What he needs to do is, the next message he will be constructing by seeing all these messages and comparing this message and there is a method for comparing these messages. So, by comparing these messages, he will be constructing a new message, ok. So, from the initialized message, he will be now taking all this information. So, the local data he will be taking is a locally greedy algorithm. So, that is why you are just taking local data; all neighbours data, not anything more than that.

So, the local data you will be taking compare them with certain rules that we will be discussing. So, you compare them and from there you come up with another message. And then, you start again broadcasting those messages among your neighbours and this

has to be done by everybody. And we will be showing, by the comparing technique that they will be slowly coming up towards a convergence of a particular structure.

So, let us try to see how they compare. So, we have already seen that they have Root ID, Bridge ID and cost. Now we have to come up with the unique Root ID, right? What happens, in the initial what will be happening? Everybody assumes themselves to be the root. If they assume themselves to be root, their ID will be coming as Root ID. Now, they have all unique Bridge IDs because it is a MAC ID which is unique.

So, if they have these things, I can now compare and I can say my comparison method will be whoever has the lowest ID that must be the root. If I declare that, then I know that, once the messages are passed from neighbour to neighbour to neighbour, then this Root ID will prevail. So, whoever in the entire network has the lowest Root ID that should become the Root ID automatically for everybody.

So, if I just for comparison take these things for, if 2 messages I see among them whichever has a lower ID, that must be my means now designated Root ID. So, if I compare my comparison should be like that. So, let us try to see 2 messages, and how they can be compared.

So, the first thing is this root you have to compare. Once you, if there is a possibility that root will be tied, ok. So, both of them think that this is the root. So, it might how that might happen? So let us say, I have this kind of connectivity. This has a Root ID 1, this has 2 this has 3. So, the initial message of him, what that will be? That will be, let us say I put the message in this fashion Root ID dot cost dot Bridge ID, ok.

So, what will be his message? He thinks his Bridge ID is 3. So, the Root ID is 3 3 dot 0 dot 3. What will be his message? That should be 0 2 dot 0 dot 2. What will be his message? 1 dot 0 dot 1. Now, he will broadcast this message to both of them. So, now, what this guy gets and he will also broadcast these 2 messages to both of them and so on, ok, everybody broadcast this.

So, everybody will get 3 messages, 1 is his own message. So, let us say think about him. So, his message is 1 0 1, which he gets from his next neighbour 2 0 2 and the next neighbour he gets 3 0 3, ok.

So, after receiving these 3 messages, if I now compare these 3 messages, ok. So, what I can see? Among them, Root ID wise if node 1 thinks, he thinks that mine 1 is lesser than these 2. So, therefore, I must be the root, ok? Now, think about what happens to 2? 2 has his own message 2 0 2 and then he gets it from his neighbour 1, 1 0 1. And from his other neighbour 3, 3 0 3. Among them, he can see that Root ID this 1 is lower. So, he thinks 1 is root. Now, as you can see they are converging.

1 thinks he is the root, 2 also thinks that he is the root, for 3 also the same thing will happen. Now, all 3 if they compare in this fashion, will be coming to a unique conclusion; that this guy must be the root and then they start operating the protocol next. Now, among the messages while comparing, what do they do? They try to first see this Root ID. It might happen sometimes.

Suppose, after this first iteration, what will what will be is that 2 and 3 both will be thinking of 1 as root. So, now, among his messages, 2 also thinks 1 is root, and 1 also thinks 1 is root. So, if he gets the next message root will be the same. Now how did he discriminate? Now, the discrimination will be cost.

So, then he will if Root IDs are the same, then he thinks about cost. Whichever has a lower cost to root that, must be my, because I want to also go to the root with the smallest possible distance, ok. Because it is a minimum-spanning tree. So, I want to reach the root, every node should target to reach the root with the smallest possible number of hops. So, if hop count is the cost then the smallest possible hop.

So, next, I will be deciding on cost, if the cost in 2 of these messages whichever is lower, I think that via him only I will be reaching the root fast. So, therefore, I should take via him as my designated root, towards the root, whatever I am thinking, ok. If even the costs are the same; that means, I see that via this guy also, I take 2 hops I take 2 hops and both of them go to the same root.

So, costs are the same; the roots are the same, and the Bridge ID is the same. Now, I have to just break the tie, somehow. So, I take it because I have to still take unique. So, therefore, I take the lowest Bridge ID and through them, I will root my things. So, that becomes my designated root, the other 1 will not be, I will be discarding that, ok?

So, therefore, The breaker is with this fashion, first, you see Root ID, among the messages, whatever message you have received from others and whatever message locally you have generated, you will be comparing them. So, comparison techniques are first you do by Root ID, then you do by cost and then you do by Bridge ID. So, you see all these 3 things and break the tie, whichever wins, that now happens to be your message, how you construct that message after comparison will come back to that later, ok?

So, this is what is the comparison that is being enforced. After this comparison, as you can see it is slowly building up towards finding a unique spanning tree in a distributed fashion.

So, we will come back to the rest of the protocols after comparison how do we come up with a message and then how do I designate the ports which 1 should be designated 1, whose packet I should forward, whose packet I should not forward, who should be in which direction my designated root is connected to. So, all these things how he decides, every node in a distributed fashion from this messages and this comparison only they will be deciding. So, that part we will be trying to see in the next class, ok?

Thank you.