Peer to Peer Networks

Professor Y. N. Singh

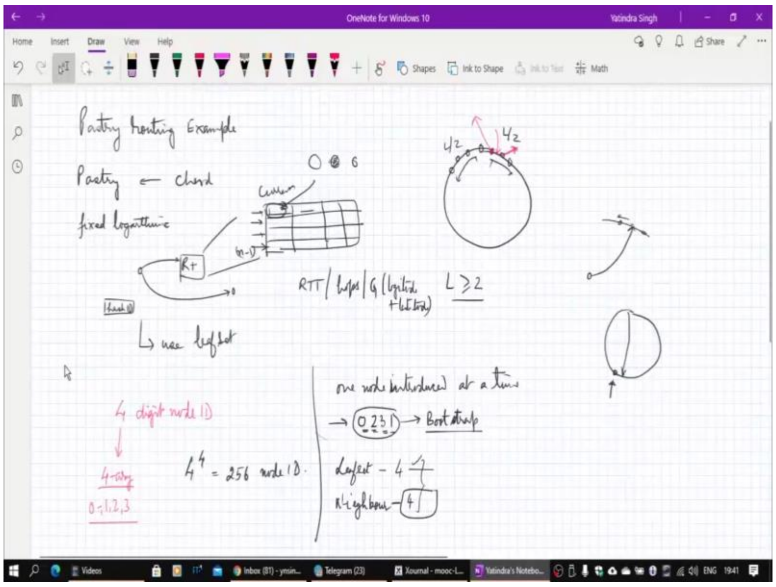
Department of Electrical Engineering

Indian Institute of Technology, Kanpur

Lecture 9

Understanding the PASTRY Protocol through Example

(Refer Slide Time: 00:13)



This is lecture number 9 for Peer to Peer Node. And we will be this time looking for a Pastry Routing example. That is our objective in this one. As we know, PASTRY was covered in the previous lecture; PASTRY was essentially derived from the chord. One important thing that they did was, the first time went to a fixed logarithmic partitioning. So, based on the digits, which were there, fixed logarithm partitioning was the one, which was introduced. Then, of course, each node first looks into the routing table based on this fixed logarithmic partitioning and decides which the next hop the query has to move.

So, the rule was that if you can get a better match to, with the Current Node ID, whatever is your hash, which you are searching for, you are searching for some hash ID, the root of this. So, you try to look at the maximum number of digits matching with the current node. So, the rule is that if you are going to find out the longer match, then you will hand over the query to that person.

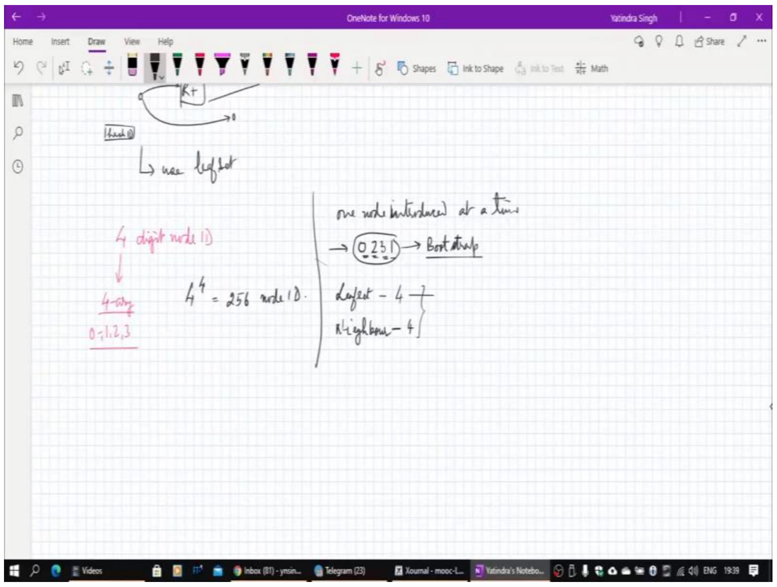
In this routing table, the routing table typically will look something like this, which we had already discussed that you would have only 1 digit with the current node, with only 1 digit, which is matching. No digit is matching; you will be maintaining any entries in that fashion. And when the 1 digit is matching with the current node, all those entries will be in the first row, 2 digits matching will be in the second row, and in the last one, n - 1, if they are n digits, n minus 1 digit will be matching, and the remaining nodes will be entered here.

Whenever you match the hash ID and go for a longer and longer match, so in this example, today, we will be showing that. And of course, if you do not get a longer match even if the same match is there, the numerical closeness, numerically closeness is looked into this, and you always pass on to the entry, your hash ID or the query to a node, which is numerically more closer to hash ID.

Once you cannot do so, routing essentially then moves on to the use of Leaf Sets. Because all the nodes are assumed to be organized in a circular fashion, the leaf set is the numerically closest L/ 2 higher nodes, and L/2 lower nodes. So, once you cannot move more to, these notes are essentially being used. You cannot move anymore to the routing, with the routing table, you use this, and find out your hash ID is closer to which node. If it is closer to this node, you will pick this up; if it is closer to this node, you will pick it up. If it is at an equal distance from both, you will pick up the entry numerically in this cyclic fashion.

So, essentially, the layout, the circular arrangement is the same as what is used in the chord. Let us now move to the example. This was the principle that is used.

(Refer Slide Time: 03:36)



So, for keeping the example simple, we are going to consider a 4-digit node ID. And each digit will be a 4-array digit, so if the number which can be used is 0, 1, 2 or 3 only. So, it is a modulo 4 numbering which will be used for each digit, which implies that you will have 4 raise power 4, which means 256 node IDs are only possible in this whole space. We have to use that in the way the, I will give this whole example is that 1 node will be introduced at a time, the system and 1 node, I will be assuming as a bootstrap node.

For our case bootstrap node number, I will mention that it will be 0231. So, these are 4 digits, and no digit can be more than 3. It has to be either 0 1 2 or 3. And this bootstrap node is fixed. So, normally in any peer to peer system, you will have bootstrap nodes. And these bootstrap nodes are used by any node to enter into the system.

Now, we also have to define that how what will be the leaf set size. So, leaf set technically can be of any size greater than 2 or similarly. We also have to define the neighbour table because we will use the proximity metric for doing a more efficient routing. So, in our case, I am taking both of them for this example, both of them to be 4.

So, normally, whatever is a base value, you are using four digits of that particular number is chosen or twice of that is chosen. So, that is normally the practice, but it can be anything. So, even if it is the leaf set of 2, it is okay; it will always work. So, as we have seen in the chord, in the actual chord, you are pointing to somewhere, so if the leaf sets are there with the node, you can always move towards a more optimal thing. So, that actual advantage is there.

In this case, because we are, when we are doing routing with this, we are just becoming closer and closer to the hash ID, but after some point when you cannot find anything further closer, you have to decide on the root node based on leaf set only. So, the leaf set is important; it has to be maintained. And L has to be at least it has to be equal to 2. So, L always has to be greater than or equal to 2.

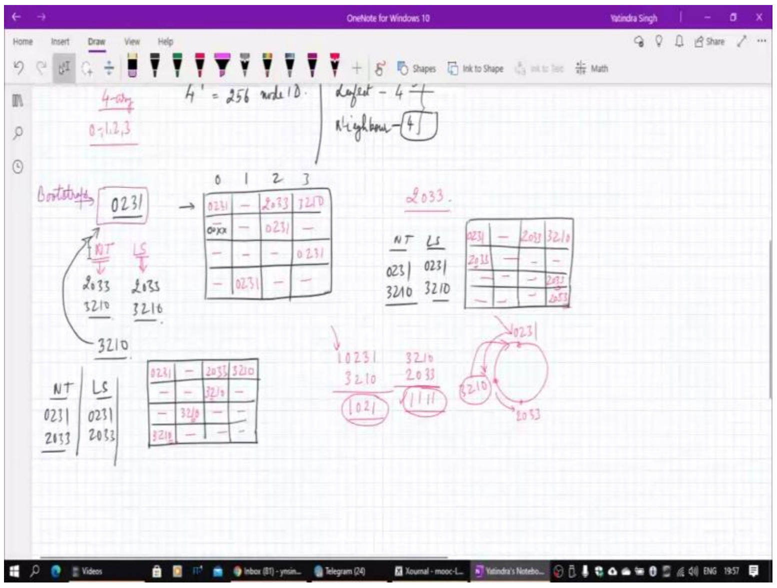
And the neighbour table, the larger we get, the better it is. So, in our case, I am taking just 4, but it need not be 4. So, why we are keeping these neighbour tables is because when I can actually put more than 1 entry at this place or maybe for that matter. Any other entry in the routing table, if more options are there, I will choose the option that will be closer to me in some physical sense. It can be RTT Round Trip Time, it can be hops, or it can be geographical coordinates, may be long or lat, longitude and latitude.

Based on that, I can find out the physical distance over the globe and then actually choose the ones closest to me. The idea is to forward the message to the query or responses to the geographically closer node whenever you are forwarding. While in the case of a chord, this is not your choice. Your routing table is; basically, you just find out the diametrically opposite one and whichever is the closest guy that becomes your, the finger table entry.

There is no choice; there is only one choice left, which you just keep on optimizing. Here, you have more than one choice in the routing table, so you become more efficient in using internet resources. You tend to cover less kilometres or consume less time to ensure that the query goes to the root node. And of course, we will do what we will do is we will; I will show you how the routing table entry moves get created as the nodes get added.

And at some point, I will assume some notes, and we will create a stable routing table. And based on that routing table, I will add 1 and see what will happen. And we will assume a Steady State Routing Table. But truly speaking, routing tables have to be in a steady-state; when a node joins or leaves is not required, it can be done at any point in time. The routing table will always move towards a steady-state situation or pretend to the proper position where it should converge.

(Refer Slide Time: 08:18)



So, the first node, as I told you, will be 0231. And there is no other entries or routing table of this will simply look. We have 4 digits, and each digit can only take four values, so the 4 values can be taken; there will be 4 columns deciding that it will be corresponding to 0 1 2 and 3, and this one is for each digit. So, in the first column, first row as we know that for 0231 and so there, no matching is required. So, all entries that do not match this will start with 0 can match this entry to put this thing directly here.

So, I am going to put this thing in red colour 0231. So, this is, there is no entry available here, there is no entry available here, there is no entry available here. And then, of course, I will put the remaining 4 rows similarly. When I start here, 0 is fixed; I have to look for all the entries with 0 0. The entries have to be 00xx, but since there is no entry, this will be empty with this thing. Similarly, this will be 01xx; it will be empty.

02xx matches with itself, so I will put 0231 here. Again, this entry will be empty. So, continuing with this thing, I can very quickly write 0231 and then I can write 0231 here. And that is where routing table entry when only 1 node is existing. Now, what we do is introduce another node. The second node we introduce is 2033, so let's introduce 2033 as the next node.

So, what will happen because of this? 2033 has to connect to somebody; it needs to find out some node. Since this is the first node, I will assume it to be the bootstrap node; I call it a bootstrap node. So, bootstrap nodes have the special nodes maintained in the system, so any new node that joins in can essentially enter into the system. Then we can update or optimize its routing table entry or leaf set, as well as for, as well as their neighbour tables. So, 2033 comes in, and 2033 will ask for, will communicate to 0231. So, it will come to know of; it will become aware of 0231. It will ask, forgive me, what will be my root node. So, the root node needs to be searched here.

So, in this case, 0231 does not only have this entry, so the first column only it can pass on, which can be used by 2033. The second column will only be useful if the first digit is going to match with 0231. 0231 will push this query on a search for the root node of 0231 towards the, depending on what entity it finds out. So, once 0231 searches for 2033, it does not find any match. The only closest match is no 0231. So, that is a root node.

So, the only first entry will be used by him. So, 2033 will populate its routing table based on the first entry, which is present, the first row, basically of which is being received. So, the first row contains 0231, which goes as it is. And the second entry is not available; there is no node with 1, starting with 1.

The third entry will be 2033, which is itself; it will fill up on its own. The fourth entry is not there. The remaining entry has to be done by 2033 because all nodes here have to start with 2, so there is only 1 node. So, 2033 will be coming here, and it's actually because this is 0, then 2033 will be coming here because this digit is 3. And this will be again in 2033 because this digit is 3.

And rest all entries will remain empty. That is a routing table after 2033 has entered into the system. 0231 will come to node 2033, so it has to update this entry. So, what it will do is, it will now find out where it can place this entry; if you can certainly place it here, it is not; the first digit itself is different. It will end up putting up this entry here, and you will have 2033, which will be the routing table after the first step.

Now, these also need to maintain their neighbourhood tables. So, this guy also has to have a neighbourhood table which needs to be populated; it also has to have a leaf set. There is no entry, but once it comes to 2033, that is the only entry that can be populated. So, it will be 2033 here and 2033 here.

Similarly, this guy will also maintain a neighbour table and a leaf set. The leaf set will be 0231 because only, and I can keep 4 entries remember, and this entry will be now also 0231. So, the next node, which we assume is 3210. So, what will happen with 3210? So, I am assuming 3210 is again actually talks to this bootstrapping node and 0231 will; it will request 0231 to find out who is the root node for 3210. Basically, for the leaf set population, that is what will be required.

And of course, along the process, as the note will keep on moving, they will keep on telling, giving them the routing table entries being passed on to 3210. So, if the first digit is not matching, the first row will be used. When the next guy, it will be forwarded to the next guy. The next guy will send the routing table. If the first 2 digits match, then that second row will be used if the first digit matches. If the first 2 matches, the third row will be used. So, in this case, 0231 will send its routing table back to 3210. So, 3210 will now populate its routing table. So, the row will come, so 0231 can be directly populated; there is no issue.

So, let me put it here. And of course, 2033 will go as it is, the first row. But the third row, it can place itself, so it is going to do that. So, 3210 is going to go here. This will remain blank.

The next row needs to have the first digit to be 3, so there is no other node. This routing table does not give him any other information, so it will keep it vacant. This only has come for the routing, this routing table, not this one so far. So, from this routing table, these 2 entries are being populated by. We will just maintain its entry, so the next entry has to be 3 2 1, which will come here, 3210 because 2, it will come in this row 0 1 2 and 3. This is blank.

The next one will be 3210, so it will be 3 2 1, because of that it will come here because of 2, you are here. And the last one is zero, so it will be 3210, with the last digit, this is the new routing table for n. 0231 will now search for 3210 and try to find out the match. It does not have any matches. So, what it will do is, it will essentially look for an entry, so there only these 2 entries, there is no match, so it will find out if, who is the closest numeric entry out of this. And it will try to just pass on to that.

So, 0231, we can find out the closest numeric entry, how this will be computed. 0231 and 3210, we find out the difference. If you arrange circularly, 0231 will come here, 2033 will come here, and 3210 says somewhere. So, this distance we need to find out, and we need to find out this distance. If this distance is smaller, so this will be the root node. If this is smaller, this will be the root node. So, we compute it. If I compute, so doing it actually, I need to put in 1 here to compute the distance because of the modulo arithmetic.

So, 1-1 is 0, 1-0 is 1, 3-1 is 2, 2-2 is 0, and this essentially will become, so 4-3 will be 1. So, this is 1021. And if I will now compute with that 3210 to 2033 and 3, I can do 3210 to 2033. Compute this, and this will become 7, no this cannot be 7; I am using a modulo 4 arithmetic. So, I have to use a proper base value. So, essentially it will be 4, because when you take the, borrow 1, so, 4- 3 is 1, this will be 0, 4 - 3 is one, so this will be 1, 1 - 0 is 1, 3 - 2 is 1. So, this is the entry.

So, which one is larger? So, this one is certainly larger; this one is larger. So, this means 0231. This is the root node for this. So, this node is going to be this guy. So, in the end, the search will terminate it will not be forwarded to 2033, so the routing table will get destabilized here.

Now, what will happen to 0231? And of course, I have further to mention the neighbour table and the leaf set. So, 3210 will now build up a leaf set, and it will invariably contain both these entries. Similarly, for the neighbour table, it will contain because all 4 is still not been there. So, there is no choice; you have to keep all the entries. So, this will be the entries for the neighbour table and leaf set. 0231 will come to know of this guy, so it will also now maintain 3210 and 3210 at both the places. So, they have populated.

Now, what will happen to this routing table? So, 0231 will find out that 3 2 10 does exist. So, 3210 can be populated here. It will simply create an entry 3210; nowhere else, the entry is going to change. Now, at this point, only 2 entries have been updated. How come 2033 will know of it? In the next thing, the algorithm is that 3210 will now exchange the leaf set values with all the members, which are there in the leaf set. So, 3210 will send this whole two pairs to 0231, 2033. 3 2 10 will also do the same thing with the neighbour tables.

And it will also request 0231 to send their neighbour table and 2033 to send their neighbour table. Or it sends the neighbour table to all these guys; it will keep on pulling and keep on sending. It will be done in both ways. So, for the leaf set also the same thing. You will then retrieve the leaf set of these nodes and send its only leaf set to work both of these. The same will be done for the routing table. 2033 will become aware of 3210 because they will send the information to 2033, which is there in the leaf set and the neighbouring table and the routing table.

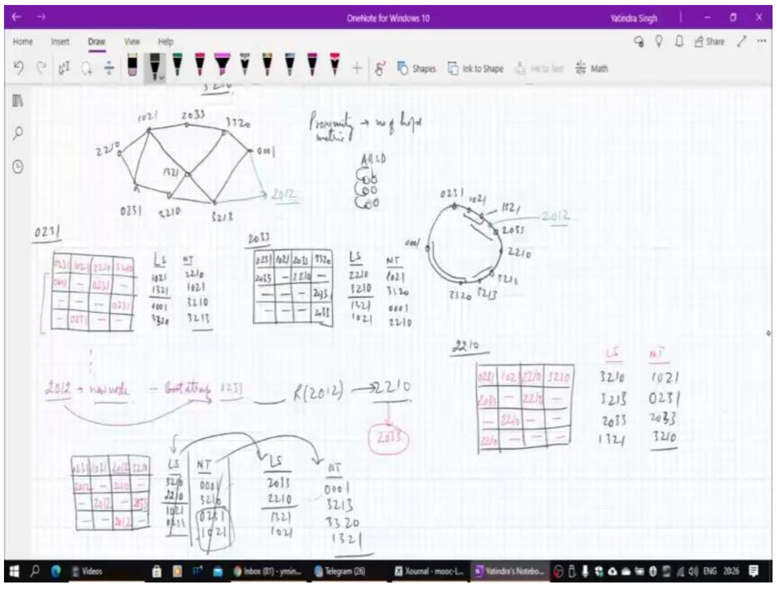
And, of course, 2033 will not give any new information to him, so there is no change. So, once 2033 comes to know of it, it will update the entries, and you will have, at this point, a new entry gets added, which is going to be 3210. And of course, you will also have because of this update in the leaf set, 3210 will be happening. Updated neighbour table 3210 will get added.

And you have now updated routing tables. And now, what we can do is we can add more nodes and build up a steady-state table. Essentially, when these more nodes keep coming, the tables will keep on getting updated for everybody in the same process. They will essentially always have come; if there is a bootstrap node, it will attach to this, ask to send the routing table back to them, and ask them to find the root node.

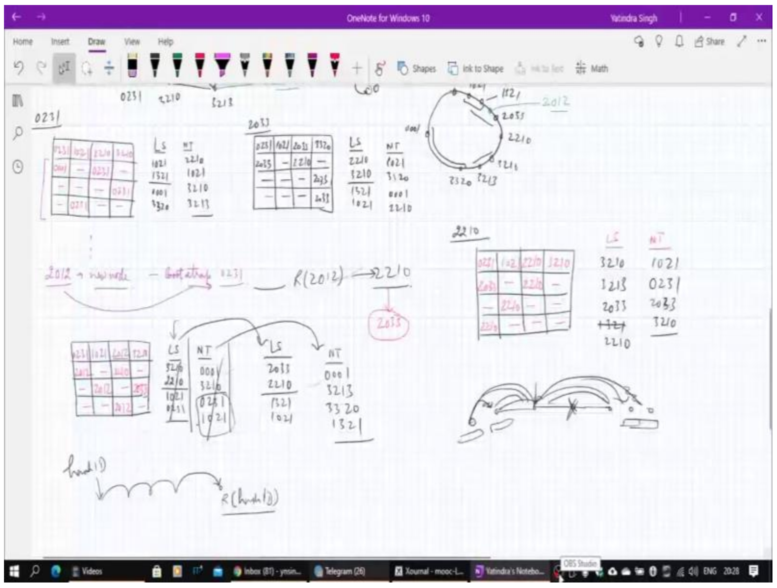
It will keep on finding out who is the best option to forward the query to that guy. That guy will also send the routing table information leaf set and neighbour table information back to the new node. So, they will keep on updating. And then, there are a periodic neighbour table and leaf set and routing table exchanges, which will be happening. So, everybody will now make better and better entries.

We need to know the topology for doing the neighbour table because we are now doing it based on the number of hops that are the physical distance, which we are assuming. So, let us see. First of all, we will try the addition of invert, and then, of course, you will also figure out what happens if we are going to remove a node correspondingly. The procedure almost kind of remains the same. So, we are going to add the following nodes more.

(Refer Slide Time: 22:06)



OneNote for Windows 10
Proximity → no of hops metric
2012 → new node — Bootstrap 0231 — R(2012) → 2210
LS
NT



So, 2210, we will be adding, 2033 was already present, 3320 is going to be added, 0001 will be added, at the time, these are the nodes that have come in. 1021 has come in. 3213, 1321, 1021, yes, 1021 has already been placed here, so it will now show up again. So, 2033 was already existing, and 0231 was also already existing. So, these are already existing nodes. In between, we also had 3210 being added, so I need to put that 3210 also. So, 3210 was already an existing node, which has been underlined.

So, we also need a topology as of now. So, I will; I am just taking some example topology. And we will be looking at hop distance, so I am writing the node IDs. And we are going to connect this and this, 3213. And of course, let's connect this. This will be 001. So, that is a topology which we are considering. And with this topology, and important thing, the proximity metric, we will take it as not RTT or something, its number of hops. When you have more nodes with the same number of hops, we will choose whichever one is numerically closer; we can choose that or choose the other way around.

So normally, the better is that if you have, you have A B C D. So, every time you choose a different value, if you are choosing 4 numerically closer ranges. So, this should be different as far as possible. If you do not have an option, keep this one as different as possible. That way, you have a diversity of choice in the proximity metric because that becomes important when you are trying to fetch the routing tables or their leaf sets or their neighbour tables.

So, your routing table entries will always be optimal in terms of the proximity metric. It is for that purpose; we have to do it that way. Now let us build up the routing tables for each one of them, how they will look like. So, under steady-state conditions, we will also have to write the leaf sets for each one of them. So, let's do this. So, for the leaf set, I am going to build up a 4 by 4. This is a routing table.

We will also have leaf set entries and neighbour table entries, all 4. Let's now create for the first one. And when the multiple possible entries are here. I will always be choosing the one which is closer in terms of the proximity metric. So, let's choose to start with any node, but we have to do for all 9. So, let us start with 0231 first.

The first entry will be 0231; let me put it in a different colour. And then, of course, we have to choose an entry which starts with 1. So, you have only 2 entries 1021 and 1 3 2 1. So obviously, it has to be 1021, which is 1 hop away. Other entries, 2 hops away. So, this will not be kept. And then we have to have the entries starting with 2. So, there are only 2 entries, and 2210 is the closer one. So, we will be keeping that unless we decide that is the way they should converge actually. With 3, we have 3 possibilities. So, 3210 obviously will be the choice in this case. The next entry has to start with 0 0.

So, 0 0 there is only one entry, which is 0 0 0 1, so I will retain that. We do not have any entry with 0 1. So, this is a dash. 0231, this itself will repeat. 0 3, there is no entry. Then we have 0 2, 0 2 there is no entry. This is itself. This will be 020. 0 2 0, there is no entry 0 2 1; there is no entry. 022, there is no entry. 0231 is there. After that, there will be no entry except one. So, this one will be the dash 0231 dashes. This will be the routing table.

The leaf set entry now, we can also put them in a circle and then see how they look. So, this is going to be, say, 0231. So, the next entry has to be starting with one, so this will be 1021.

The next entry will be 1321. After that, we will be having entry with 2, so it will be 2033. Then you have 2210.

All entries 2 are, with exhausted now, we will start with entries with 3. So, it will be 3210, 3213, 3320. And after that, it will be valued, again will come back to 0. So, we have a total 3 plus 3, 6, plus 9, total 9 entries in the ring. So, that is a numerical relationship between them. So, leaf set for 0231, you can conclude from here.

So, I need only 4, so these two will be the leaf set for 0231. So, I can write that to write in 1021 and 1321; one side and the other side will be 0, 0001 and 3320. So, neighbour tables will come from here. 0231 has 1 neighbour, is 2210. You had in 1021. You have in 3210. There is no entry with 0, which is why I have just to find out which one is closest to 1 2 and 3 has been done.

So, the next entry can either be this, this or this. There is no other entry, which is feasible. So, I can keep any one of them. So, I will try to keep entry with 0 possible, but there is no entry with zero possible. So, I will keep any one of them so that I can keep, say 3213. So that is a neighbour table, the 4 closest nodes for 0231.

Similarly, I can now build up tables for others. So, let's do it quickly. And, so let's do it, and then this one will be for 0231 has been done. Then we can start with 2033, which was our second node, and very quickly writing it down, 2033 will be here, 2033 will be here. So, I am filling up. I am not explaining how it is going to be done.

You understand it by now, but I just have to fill up the entries. Remember, from 2033, 2 entries can go with 0, but both are 2 hops away so that I can pick up any one of them. So, this is pretty fast, and of course, we have set, you can figure out from here 2033, so it will be 1321 and 1021. So, in the neighbour table, you can again compute by looking at this table. If you compute for 2033, the neighbour table will be 1021, obviously, 3320. Remember, most of the neighbour table entries, you have always trying to use it, or whoever is, are in the neighbour table of these members.

So, these members also have a neighbour table. So that is why it makes good sense to get the routing table from these guys and fill up those entries because they will be closer to what you always over time. That is what you will figure out. So, 2033 has got these 2; then it can pick up 2 hops away they are, this node, this node and this node. Anyone of them can be picked up. So, 1 and 3, we have already picked up.

So, we would like to pick up a guy with 0. So 0 obviously should be here. This could have been this, as well as this. Any one of them could have been picked up; I picked up in this case 0001 but, and then, of course, one more entry which has to be done. So, I can choose anybody and return to 2210 maybe. So, this is a neighbour table.

And so, doing it this way, this is, so other tables also can be done. So, I am not doing them here. What I will do is I will now assume that routing tables are there, and I will try to add. So, whenever it is required, I will compute the nodes' corresponding routing table. I need not because all routing tables will not participate when I try to add a new node. So, whichever will be required, and we will just populate that and use that.

So similarly, we have neighbour tables and then leaf sets, so we will also consider that. So, this basically, I am trying to give an example. So, you can work out on your own and then verify this particular thing. So, let's now add a new node. A new node, which we will add, is, let it be 2012, a new node, which we are adding, 2012 is a new node, which I want to add. Now, what will happen?

Now, bootstrap node, I have to decide. So, as usual, I will take the same bootstrap node, which everybody else had taken when I first started 0231 and with that now, let's find out. So, 0231, 2012, will now talk to 0231 and ask him to give me what will be my root node. Now, let us we will use that; I will just draw the routing table. You can verify whether it comes from a steady-state or not. 2012 will come first of all talk to 0231 because 2012 has the first digit as 2, this one a 0. So, the first row can be directly used by 2012.

So, let me build up a routing table for 2012. So, you do it, so 2012, we will use this routing table directly as it is except wherever this entry is there, it will put itself because this has to be closest to itself proxy using proximity metric. So, you will end up getting a 0231 here. So, I will now put it into different colours 0231, 1021, because it does not know anybody else. So, there is no question of optimization. So, it does not even have its own leaf set. So, it's 1021, so that is where the entry will change. Now, it will be 2012, and the change is going to happen here.

And then you will have 3210. The second entry cannot keep because they all have to start with 2, so I cannot use these entries of 0231. So now, 0231 will look for 2012, 2012 will come into its knowledge. So, it has to update its routing table if it is required. It will look into the, and it can only happen 2210 can be replaced by 2012. We are adding this entry, so I keep this entry and keep this entry here in 2012.

So, 2012 is far-far off compared to 0231, so it would not prefer that implemented 2210, which is 1 hop away okay. There is no change in the routing table of this, no change in the neighbour table. And of course, will there be any change in the leaf set? Yes. Let us see if there is going to be any change in the leaf set. That, 2012, will be getting inserted here; this will get inserted here.

So, there is no change even in the leaf set; the leaf set remains the same. Neighbor's table remains the same. So, there is no change in 0231. But using these neighbours, it will build its own leaf set and neighbour table. So, what will it be? So, as of now, it is only aware of the nodes which are present here.

Based on that, it will maintain its own leaf set. And whatever leaves that information, which will be retrieved from 0231, it will use that. So, using that, so if you look at 2012, it is aware of 2033, no it is not aware of that, 2210 is not aware of, it becomes aware of that, so we will just actually use these entries to 2012, anything which starts with 2 had to come here, 2210 will be the one entry. I am just increasing the entry 2210, and then after that, we have 3210. That is a leaf set, the higher side part. The lower side part will know anything lower than 2 because there is nothing with 2.

So, one will be there, so only 1021. So, it will just put 1021 here. And anything lower than that is 0, so it will be, put 0231. That will be the leaf set to begin with when the next hop, the routing in search of the root node, the packet is forwarded. So, that information will come again this everything will get updated.

The neighbor table will come from the network, which is there; it will do a test and find out the hawk. So, 2012 is, in fact, ideally speaking; if it is not, its internet hops, which you are talking about, and this is an internet of connectivity, okay? It can actually send a router and find out, later on, we will find out this guy is also part of the DHT. We will use that, but it's a hop in the IP layer. So, 2012 is going to be closer to whom? So, we have to figure out that.

So, I will just slightly reduce the size of this would become easier for me. And then I will come back to the same size. Yeah. So, in 2012, I am going to only look at these nodes. 001 is participating here, so it will become one of the entries, is one of the ways. 3213 is it present here? No, it is not present anywhere, so that would not come. Now, 3320, is it present there? No, it was not present in 1321 is not present. 3213 was not there. 3210 is there so that it will maintain 3210 as the next closest one.

So, these nodes are all exhausted, there, which are 2 hops away. So, now 2033, is it present here? No, it is not present in any one of them as of now. So, what will happen? So, 2033 will not come. And in 0231 it is present, so 0231 come 1021, so this will become the neighbour table, to begin with.

Now, this guy will search for 2012; the best match entry is 2210. So, it means this query will be of the root of 2012 will be handed over to 2210. So, I need a routing table of 2210 now. So, let's create that. So, by looking at all the information, I can create that very quickly. I will only be creating what is required.

So, this will be 2210, so 2210, the routing table will look like, the first entry has to be starting with a 0. So, 2210 is here, so 0231 obviously, there is the closest guy with 1, it is 1021, with 2, it is going to be 2033 obviously, and with of course 3, it will be 3210, with 2 0, if any entries, 2033 is there so that it will be maintaining 2033. Then we have 2 1; if there is anybody else with 2 1, there is nobody else, so it is a dash. With 2210, this will be itself. Now, I have to ensure that this 2033 cannot be kept; this has to be replaced by 2210 because that is the closest node to itself.

So, this needs a correction. And of course, the last entry will be 2 3, there is nobody, and there is nobody I think with anybody else. Wait. Sorry, this has, this has to be changed slightly. Is it the wrong place? 2022 will come here. This is 220, 221, and this has to be here, sorry. And this has to be 2210. So, this will be the routing table for 2210. So, when 2012 will come here, it will try to do the match. And the better match will be now 2033 and 2210. This will be the better match, which will get the second layer to hand over to 2033. So, this query will now be handed over to 2033.

But this routing table information will be sent back. So, the first entry has already been prepared. It will find out if there is a better option: to get out in 2012, which will keep itself. The remaining entries will be as it is actually. There is no new information. So, from the second row, it will find out, this itself had to be there in 2012.

The second entry will be a dash; there is no other node, no new information it can get from anybody. This one will be now 2210; this will be a dash, dash, 2012, dash, dash, no one else with 2, and only 2210, 2033. So, 2012 and 2033 will come here. And of course, the last entry will be 2012; with 201, there are no other nodes. 2033 as will come, so there is one more entry that has been populated. So, which knowledge will come from here and used in this case is a better match.

So, we have got this updated. And once you know these particular nodes, these leaf sets and neighbour tables have to be updated. Of course, this 2210 will also tell about its leaf set and neighbour tables. So, that information also has to be used. So, 2210 stands here; it is going to contain 3210, 3213. And of course, 2033 and 1321, and neighbour table again, it will compute from here. So, it will consist of 1021, 0231, and of course, you can put 2033 and 3210.

So, two guys 2 hops away and 1 person, which is 2 persons, which are 1 hop observing. So, once all this information comes, there is a lot of information based on that optimization that has to happen here. Now you have more knowledge, almost all nodes are now are actually, so 2012 is aware of these nodes, its aware of these nodes, because of 0231 it has already become, so it has become aware of everybody almost. So, it will essentially now do the hop count and optimize itself to find out the best leaf set by this time. So, everything will now be essentially merged. All information is there; this is all organized.

So, it will find out its own leaf set, and the leaf set will get corrected. So, the leaf set will now contain no more information. So, the leaf set will now change to, because more information is coming. So, this will now become 2033 and 2210. And on the other side, it will be 1321 and 1021. So, that will be the leaf sets.

The Neighbour table similarly will get updated now. It's basically by physical measurement; it will be coming out to be 0001 3213. Earlier, these 2 guys were being picked up because there was no knowledge of the intermediate guys, other guys. So, we can now put 3320 and any one of them so that it can be 1321. So, this will be the new routing table. And these guys will also get updated, so they also need to update, so for example, 2210 will come to know of this thing so that it will update its leaf set, so 1321 will be gone, and you will have 2210 now coming in.

Similarly, the neighbour table will also get updated. For 2210, it would not be making any difference, but this change will happen at all the nodes consequently. That is how the routing tables will keep on updating and keep on remaining in the optimal position. So, you will essentially be picking up whenever you want to do essentially routing for any hash ID. So, wherever, at whichever node you are, you will be first looking at the routing table, finding out the longest match or even the same match that it has to be numerically closer. Now, similarly, when you delete, the entries will essentially be parsed.

So, some entries from, for example, if you have two nodes: the leaf sets in front and 2 in the back, you are here, so if this guy dies off, you are seeking the leaf set from each one of them. So, all leaf sets are coming to you. So, this guy will be telling who is there the next two. The moment you figure out this guy is not there, and you will now get this particular set, you get this set, you get this particular set and based on that, you will choose. This entry will be parsed, and you will now make the entry to these 4 guys. So, this entry will get updated.

The same thing happens with the neighbour table continuously. So, anything, any better option will come, you keep on updating. So, with deletion also this will automatically repair, but this is a very fast mechanism. So, this was an example of PASTRY routing. So, next, we will be looking at the Kademlia Routing, which is another one and which works in a slightly different way because here, the distances are being computed in this fashion.

We do not worry about the numerical distances. So far, I can find out how you start at any node, and you can be routed, and you will always reach a unique node. So, given the hash ID, there is a unique mapping to a node, which we call the root of the hash ID. So far, this can be done; that system can be used for DHT-based networks. So, with that, we close and see you in the next lecture.