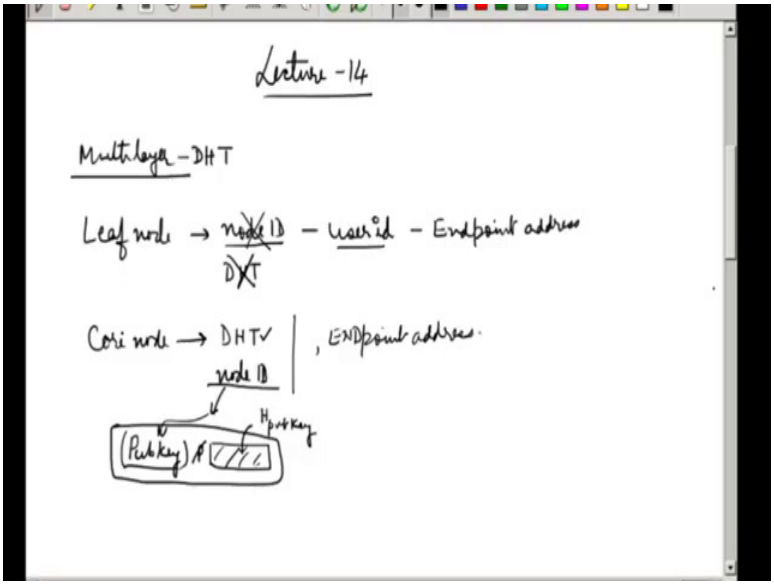
Peer to Peer Networks
Professor Y.N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Lecture 14
Multilayer DHT: A Design for Multiple Series

(Refer Slide Time: 00:14)



This is lecture number 14, which is for Peer to Peer Network. And so in today's lecture, We will be covering Multilayer DHT. So why we call it multilayer DHT is because single DHT network is not good enough sometimes for when you are running many services. So as far as only searching for a key value pair only one kind of key value pairs across the whole network and single DHT network is good enough.

But in any real life implementation, you probably have to go for multiple layers. So first thing is what will be this layers will look like? So what do you mean by layers actually? So one thing which is important is that every node which is participating may not be participating in DHT, if it is not participating it will be leaf node. If it is a leaf node, it may not even require a node ID.

Because nobody will be communicating via DHT network to it and it will be most likely there is no node ID for leaf thing. It will be going to have an user ID by which if somebody wants to communicate can communicate to them actually. So this will not be participating in DHT that is also clear. And this user ID can be access or this node can be reached only where end point address.

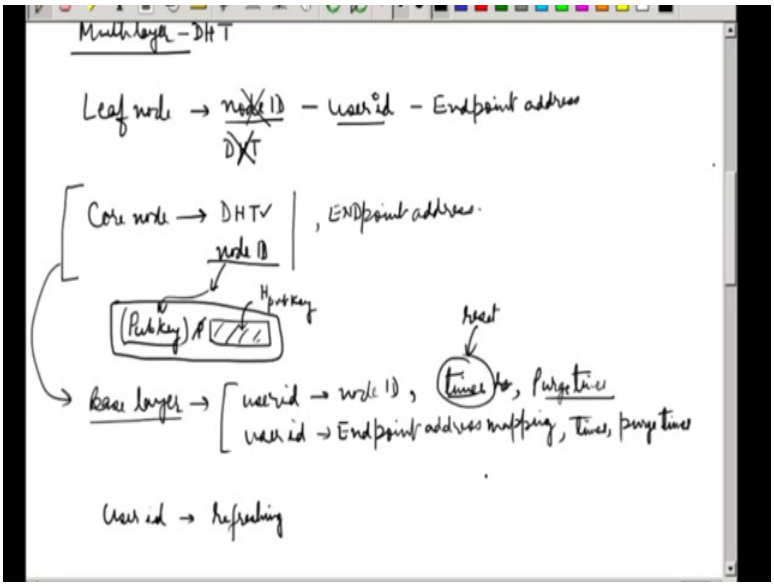
So this end point address from now should become discoverable, there is no user ID. So this will be one kind of nodes. I am only talking about the first base layer part. There will be some nodes which we call them core nodes. They will be participating in DHT. They will also have a node ID attached to them. So remember the node ID as we have discussed in earlier lecture consists of two parts.

So, one is the basically been generated using public-private key pair generation RSA algorithm. And the public key part is what is your node ID. This is this and whenever this is been presented, I need to always still sign it with private key and I can actually when I am signing I have to this is a sign hash which is going to be edit and I can represent this thing as hash which is sign by private key.

So this is what is going to be will be always will be available. Reason for doing this is that we will ensure that public key is always generated as a random number because both have to be regenerated together. You cannot arbitrary choose a public key. So if you decide on public key first then want to generate a private key that is not feasible.

So core node will do this, core node will also have endpoint address. There is no doubt but this endpoint address can be identified by sending a query directly to the core node. So this (node) never be kept in the database.

(Refer Slide Time: 03:36)



Now I will define the base layer where in the base layer all the node IDs which are participating in the DHT. So these nodes which are core nodes will be a part of this layer. So they will, these nodes will be now only storing either user ID to node ID mapping or they will be actually storing user ID to endpoint address mapping. So these two things can only be kept nothing else. Okay. So then these entries are kept.

Now the problem is a user can go off and after some time when new user comes in he by chance choose the same node ID. Then this entry has to be purged. This entry only has to be there till the user is there for some time. If user actually is down or user churn is there in that case we have to retain for some period. So normally the algorithm will be that if every user so he will be having user ID, he will be refreshing this entry. He will be republishing in the DHT.

Once it is done there has to be a timer which need to be associated. This timer will be reset. So this timer will be reset whenever this refresh or re-publishing will be happening this will be reset. If this timer expires this timer normally will be very large then of course this node does not exist that is what we assume but we can retain this entry still for some time so that if user comes in then actually he can just do the refresh.

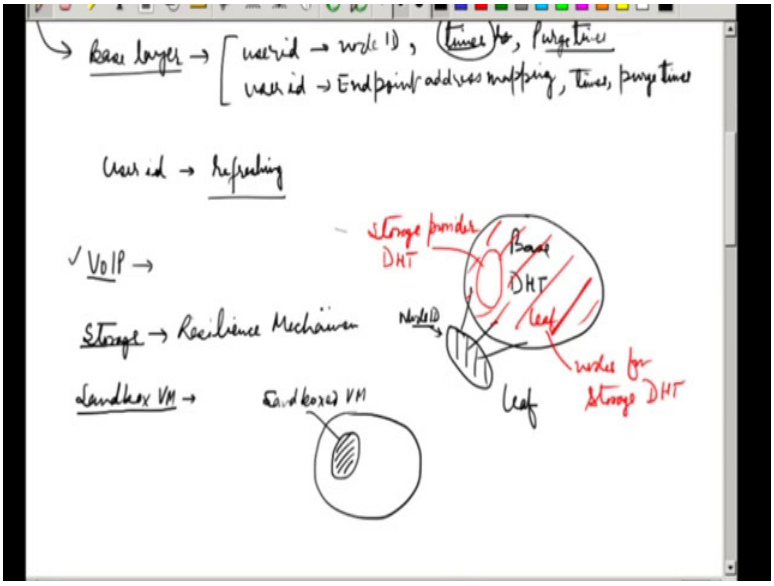
Of course I will not be using this entry for doing anything else as of now, okay. But some other users can actually identify that user. So what is the node ID for that if they want to have? There will be another timer which will actually be called the purged timer for this. So whenever the

reset this particular timer expires, then perched timer will be kept and after the perched timer is gone this entry will be removed from the table.

Similarly, there will also be a timer here which have to be maintained and there has to be a perched timer. So this is very similar very standard technique to of having two-timer. This is also used in distance vector routing protocol or removing the absolute entries same thing is also was done OSPF also. So more or less wherever the entries has to be refreshed we actually first of all disable it and after some time only we will actually remove it.

So the same procedure we are also following in our Brahaspati-4 design also. So every user need to keep on refreshing this particular entry.

(Refer Slide Time: 06:30)



So the refreshing is very important and there is a timer, okay. So now coming to what will be the other layers and then how will organize? So now this is not good enough this is for a voice over IP kind of thing. This may be okay. So from user ID, I will find out note ID and if that node is participating in the DHT, I can actually make a query for that node.

And node ID and that essentially query will terminate at the same node itself because that user has published his own node ID, okay. And then based on that then I can directly talk to him. Once I know I can that note ID will tell the IP address back and quint address back and then you can communicate.

If that guy is a leaf node, he will be actually publishing in the DHT network his user ID to endpoint address mapping. When I want to talk to him I will search in the DHT for this user ID. This time I will not get node ID, but I will get endpoint address and then you can directly communicate to him and then to the voice communication with the other endpoint. So now what will be the other kind of DHT? So one of the very important thing is a voice over IP is a service which is normally does not require a separate DHT this will be based on purely on this base layer then we can have storage.

So the idea here is that lot of people actually have extra storage space. So you can communicate you can donate it to the community. So everybody does it so I have actually get a large pool. So whenever you want to put your files, you will put your files you can actually fragment them and then these fragments based on their some key identifiers we will talk about this storage design later.

Based on that these fragments will be distributed across the network. So even if your machine comes off it dies, you can put another machine set up everything and then you will be able to access your system and we have to have a built-in resilience for reliability mechanism for this DHT layer. We will talk about this reliability later.

So storage can be a layer but not the base layer every node will be participating. But except some nodes which are the leaf nodes they may not be participating. So I have a base layer thing. So the nodes which are participating in the DHT are in this set. There will be some nodes which are outside. So this becomes the leaf nodes, which we will be using endpoint address. So they may be having the node IDs and they can communicate but nobody can communicate to them directly because they do not maintain node ID. They need not at the base layer.

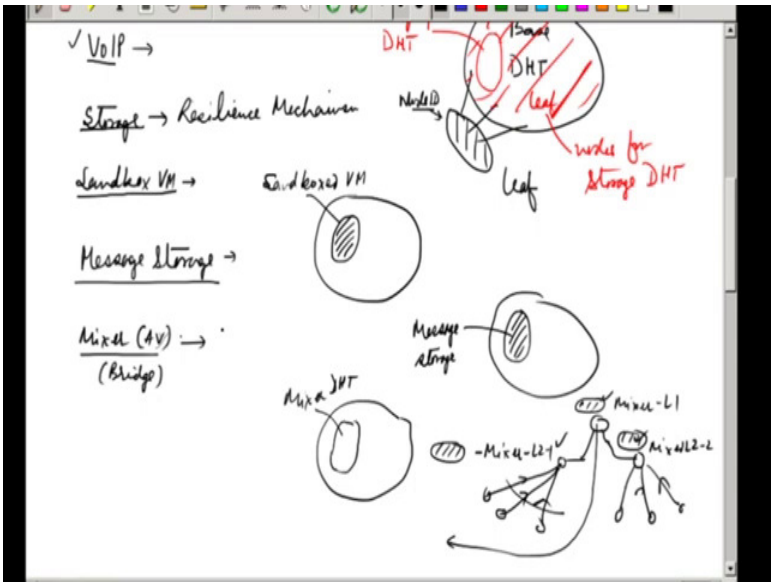
Now further taking of the concept out of this some nodes may decide that I am only going to do. These nodes only participate in providing the storage space. So, I will create their separate DHT for a storage providers. We will form a separate DHT. Now what will happen to this other nodes? They are not proving storage but they need to use the storage, so they will become now the leaf nodes for this particular part whatever is there will become the leaf nodes for the storage DHT.

When the same base layer, I can actually have even other services so I cannot provide for example M-box virtual machine so you can actually push in the compute job so these machines are. So, I have to draw a separate this is the base DHT. So some machines will be providing the sand box VM service. Okay.

So other guys will only simply be using the service provided by this, so even if one of your nodes dies, the load essentially get balanced depending on for which node who is going to be responsible in this that is what we will do through essentially mapping. Okay. We can have similarly intermediary services.

Now this service there is nothing like storage. Okay. There is no reliability which is required. If your node dies off I will just get something else. This is the storage here files are going to be stored. So or the fragment of your files will be stored across the whole network. So there need to be reliability. So there is going to be separate resilience mechanism which need to be built in this case. In this case, we will not be requiring it.

(Refer Slide Time: 11:23)



So we can also intermediary service for example messaging. So if the other endpoint the destination is off so where is messages is to be deposited. So there can be again in the same base layer DHT some nodes will decide that we will be acting like a message storage. But message are not going to be stored forever. So they normally you will be each node will be passing on my message if the other person is active at that point of time directly.

If it is not there then this message storage will be used. Okay. So we call it a message storage. We can have similarly another layer. For example say audio mixer. So you are doing video conferencing so there will be again problem that normally everybody's audio should not come to you and then you are not locally going to mix because then your bandwidth will be consumed. The 50 guys transmitting audio video and you get all 50 audio video that it is very cumbersome.

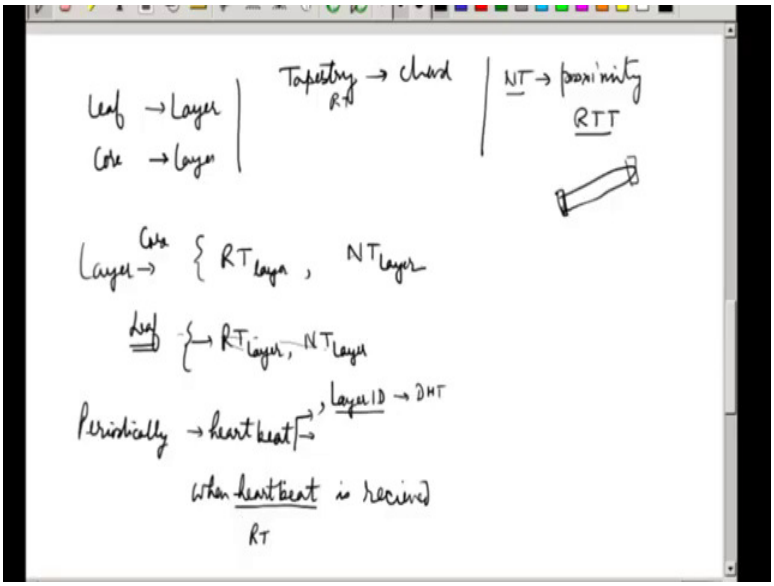
Normally it is a good idea that all audio videos can go to some nodes depending on whatever is the capacity. So 10 guys can connect to one, he will generate audio video. There is another guy which actually gets this thing. Okay, it also mixes it, two actually then mix audio videos are send to third guy which will go to final mixing and this gets broadcasted to everybody. So these are mixer nodes.

So we can actually define say mixer 1. There is layer 2-1, so I can define similarly ID for this mixer layer 2. This is mixer at layer 1. And then ofcourse, I need to append what is going to be your stream, which you are talking of the conference ID which you are talking about. So in voice over IP, this can be edit now, who will become the mixer now everybody here.

So then in some nodes you may say willing that I am going to become a mixer. So there going to be a mixer in DHT in that case which has to be there. They will act as a bridge, conference bridge. So everybody can communicate to them they will mix and then it is going to be broadcasted. How this body of broadcasting will be done I will again tell in one of the next lectures, okay.

So we can have this AV(audio-video) mixers. I should call it say AV mixers which will be acting as a bridge or I can also call them bridge and they can form a DHT. Now the moment you form such a DHT what has to be done? There is a problem here now. How the routing tables will be managed?

(Refer Slide Time: 14:17)



So one thing which we understand that in some layers you are a leaf node, in some layers you are a core node. And also now I remember we also we should we are implementing in Brahaspati-4 an optimization because we are using Tapestry algorithm. In fact Tapestry and we have combined the Chord thing to reduce the number of nodes in a column. So we only use predecessor, successors and middle value and based on that we do the routing at 1 digit level, okay.

And we are using hexa-decimal number. So we are also now going to maintain neighbour tables. Neighbour tables are based on proximity. May be round-trip time will be used here that is what we are planning as if now. So this takes care of because you are sending a message that is coming back it is actually taking care of the network load on both sides and based on that it is figuring out who is closest to you.

So it becomes some more efficient or basically time wise in least time all this communication will be happening in our DHT system in Brahaspati-4. So there will be neighbour tables also and there will be these routing tables. So what happens is for each layer so there is going to be a layer. I will be having a routing table for this layer. I will also be having a neighbour table for this layer. So routing table and neighbour table for each layer will be maintained.

Now, there will be nodes which are leaf nodes for this layer. They are just using the services so they will also be maintaining a routing table for that layer. They will also be making a

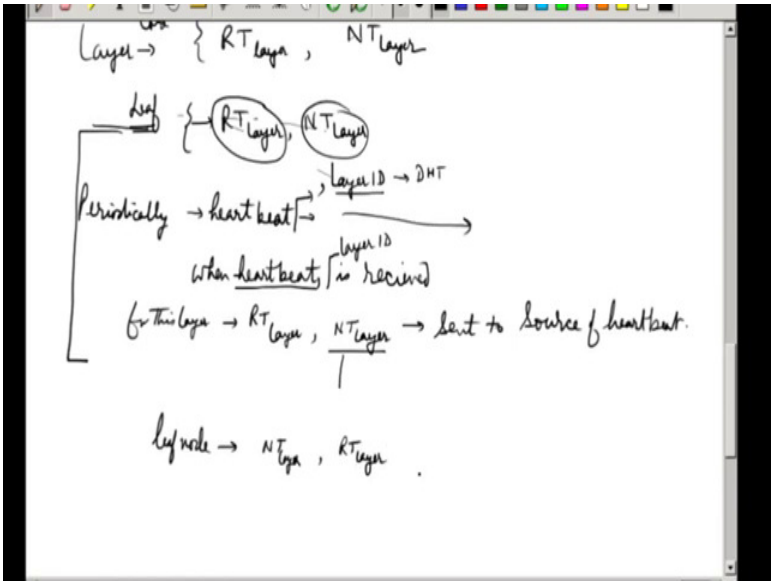
maintaining a neighbour table for that layer but this leaf nodes, this is for the core nodes this is for leaf nodes. But this leaf nodes cannot be used as a transit.

So nobody else need to keep this entry of this leaf node in their neighbour table for that layer as well as in the routing table for that layer. Leaf nodes will periodically so there is going to be again a timer associated with periodically will send only heartbeat that they are alive nothing more than that. That they are alive and they will mostly will be once the heartbeat is send then technically to all the guys who are there in routing table who are all the people who are in the neighbour table.

So whenever heartbeat is received, whenever the heart beat is received the local routing table. Now this heartbeat now has to be we have to identify which layer. So for different layer essentially now, it will be different heartbeats. You may be sending at the base layer to everybody but at a higher level only to some people. So the other way has to identify which routing table or which neighbour table has to be sent back.

So this heartbeat also now will contain. Now this is the change which is going to come layer ID okay or whatever DHT which I am going to define. So even all routing table will also have layer ID which is required so we can define multiple layer IDs for different purposes.

(Refer Slide Time: 17:58)



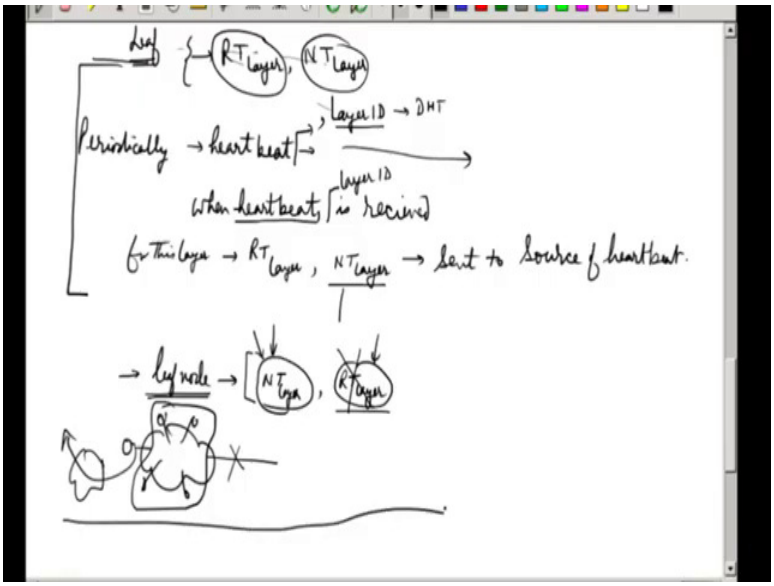
So when heartbeat is received layer ID will be identified. So what you will be receiving this as well as a layer ID. So RT for that layer, okay, so for this particular layer, which is for which

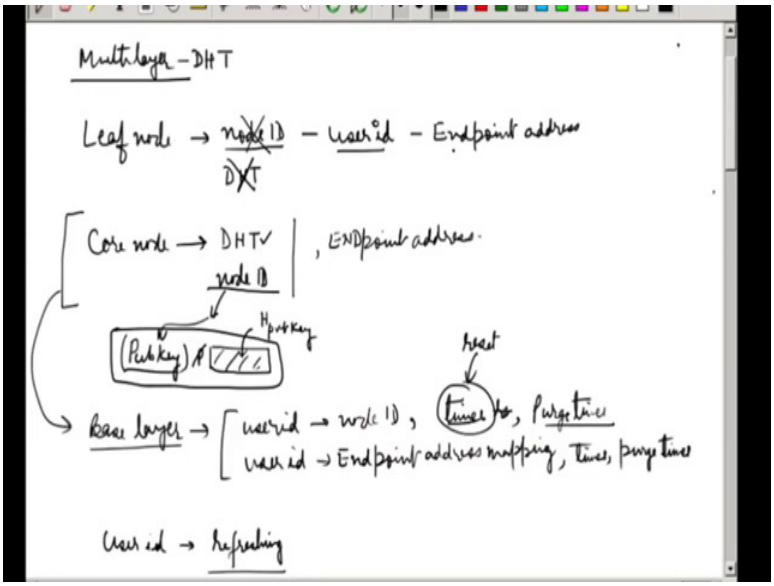
heartbeat has come. Copy of RT for that layer and neighbour table for that layer will be sent back. So over a source of heartbeat will send it to that.

So now I remember this and this heartbeat is being sent to whom? Sent to everybody who is there in your so this I am talking about leaf node as of now first, this is for leaf node. So all members who are present this we are sending this particular heartbeat to them. So I will be getting a lot of nodes. So I will update my local neighbour table.

So the leaf node for that layer the neighbour table which is there that will be updated. And it will also update the routing table for that layer. For all layers this has to be done. So if there are very few nodes I think very few routing table entries which will be there and that we will be keeping.

(Refer Slide Time: 19:31)





Now the routing table essentially depends on what is your node ID. When you are a leaf node in a layer then it is okay because you have a node ID and then you have to keep it. But if the leaf nodes for example I have told that leaf node in the base layer may choose not to have a node ID because they do not require this node ID. If they choose to keep it is going to be there but it cannot be used for routing. So this entry should not exist in any routing table, okay.

So routing table will always be present. So this routing table optimization basically depends on what is the current node IDs. So Leaf nodes, which do not actually have the node ID how they will maintain this routing neighbour routing table? This becomes a challenge. So this basically

depends RTT whatever is there for every layer it can keep on making a neighbour tables. But what about the routing layer?

They have to assume some node ID in that case. In fact, either they do not require if they maintain node IDs, they should maintain routing table if they do not they actually should simply assume a node ID which is by one of their neighbours. So essentially if this leaf node will only be maintaining this routing table if it is at the base layer if there is no node ID assigned to it. In that case it will not be requiring a routing table it has to just hand over to somebody.

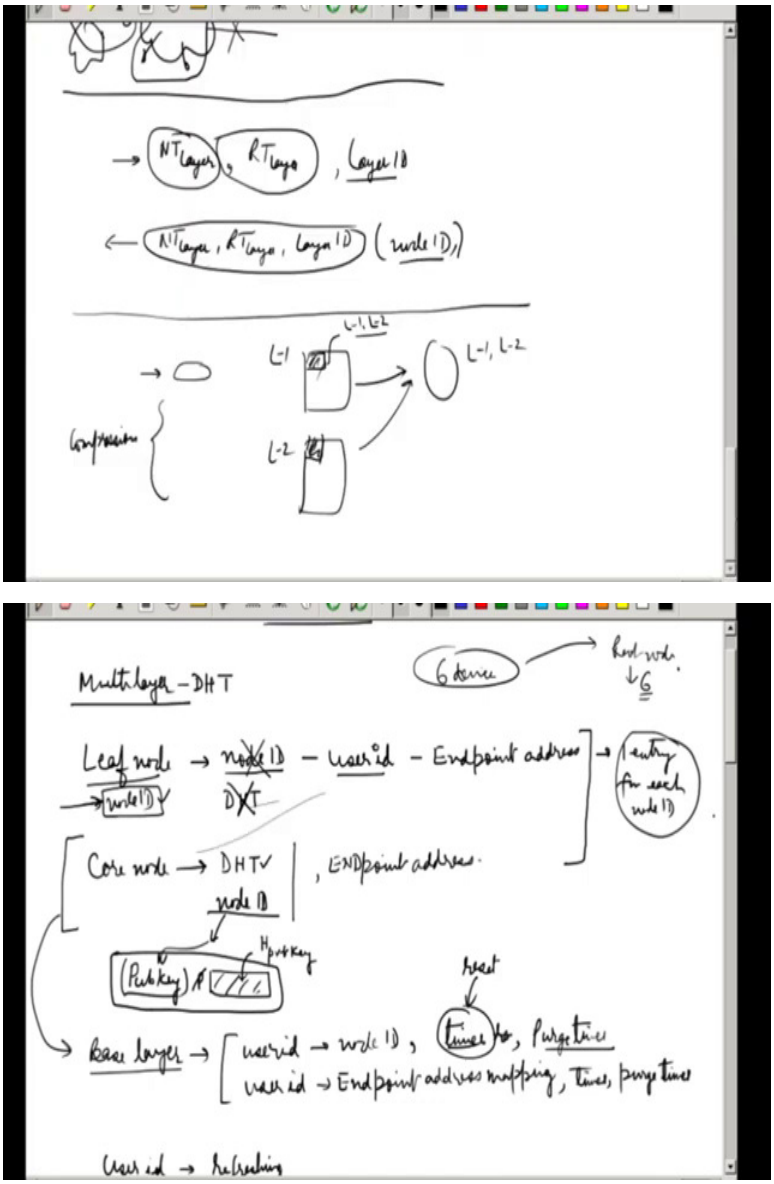
So it will just use of the neighbour table entries to forward the query and from there the query can further move. It only need to maintain this routing table actually will not be required if you are leaf node without node ID. If you are leaf node with a node ID, then you are going to maintain a routing table. Advantage of maintaining a routing table is only maintaining neighbour table there is a problem.

Because if you are for example in IIT Kanpur Campus, I have maintained only my neighbours and suddenly somehow what happens the network is dead. Okay and I am able to connect through some other network. But my only gateway was this I have to again start this whole neighbour table is waste for me now. I have no other entry I have to again start connecting to boot step node and from there get a routing table and then find out, then optimal entry for me.

Okay, again do a neighbour table search if I connect even through another network. So it is actually always good to maintain node ID because that way you will never be getting isolated from the network. Because you will have in the whole node ID space from every partition at you will be maintaining at least one representative. So even if you are connecting through another network from where you can start populating the things back.

So now for each layer separately the routing table exchanges will take place and for each layer we will also be having routing table. So this is for the leaf node that they actually should preferably have a node ID and correspondingly they should also maintain from neighbour tables as well as the routing tables for all layers. Now the nodes which are in the core what they will do? So let us come to that.

(Refer Slide Time: 22:47)



The nodes which were core nodes in that case you will not be sending heartbeats, but you will be actually sending your neighbour tables. So for each layer where you are actually there. If you are in a layer if you are a leaf node, then you will be sending heartbeat along with the layer ID.

In the layers in which for which you are actually updating your local tables, but if you are participating in a layer you will be now broadcasting your neighbour tables as well as your routing table at that layer so all the entries which are actually there listed in these two. Similarly from the other side you will also getting from when you are sending the your routing table in neighbour tables to all the entries maintained here.

You will also be now requesting from all these peoples to send their neighbour tables and routing tables. So in fact that you do not need to do explicitly the rule is you will also be getting a request from others. So other people will also be sending the neighbour tables for a certain layer. So remember when this information is sent, the layer ID need to be sent in all such messages. So it is a layer by layer communication, so when you receive a neighbour table as well as routing table along with the layer ID you will use these two things to further optimize your neighbour table as well as routing table both.

As well as whoever is sent this entry you will also consider that as also that node ID also as a potential candidate to be used in neighbour table as well as routing table both. While in case of when heart beat is received that source node is never considered. Here source node ID also need to be considered for this purpose that will be one difference which will be happening, okay.

And so you will be sending your tables to everybody who is listed there. So your source ID as well as all the entries which you have will be considered by the others for their optimization and you will be doing it whatever you are getting from other people. So their source node ID and whatever they are sending.

And for that every layer you will be doing this optimization, sometimes now either one optimization which can actually be done at this case. We have not thought about it. We as of now are planning to send for each layer separately the neighbour tables and routing tables both. But for certain entries for example are common to multiple layers. So same entry exists in one particular layer 1, layer 2, layer 3.

So I can actually now combine these and basically compress. We can additionally put a compression of data, so but this also slightly tricky because you need not send everybody a routing table for all the layers because you may not be participating all the layers, so when your leaf nodes you do not sent any routing table for that layer you only receive it. You only send the heartbeat packets.

So compression probably does not make sense it have to be much more cumbersome because you have to figure out which one layer has to go to which particular node ID and if you know that for all those node IDs who all those layer IDs only you will be compressing if they are

common entries you will be instead of multiple make only one common entry that is what you can do.

So technically if this is for layer 1, this is for layer 2. You have an entry here same entry comes here, so instead of repeating this entry twice by sending this tables separately and they both have to go to same node. So this node is also now layer 1 layer 2 subscription. So I will be now compressing and will only send it once but while mentioning it that is has to be layer 1 as well as in layer 2.

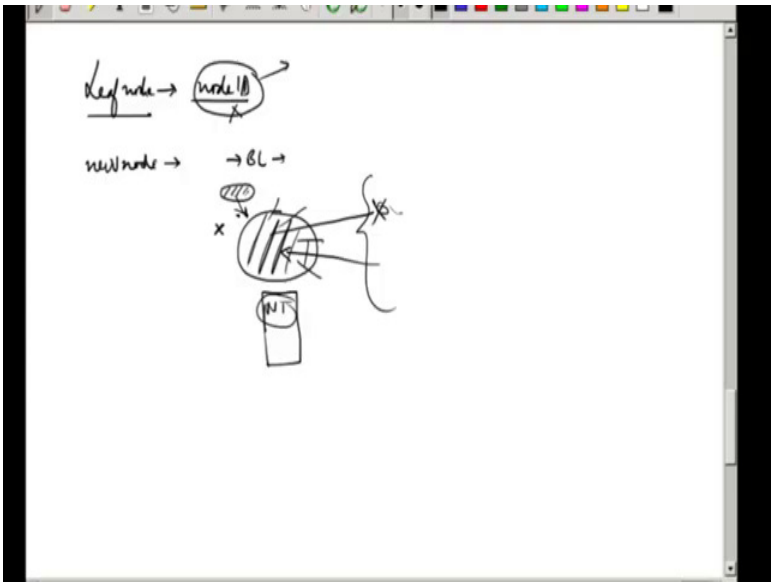
So that a kind of optimization which can be done. But this is independent of the DHT algorithm which we are talking about. So layering is very important and this need to be implemented so that various kind of services can be done implemented in the whole system. The only thing is that the base layer everybody has to participate and if you are leaf node, you do not want to store anything that is the only way you can become a leaf node.

But since if you carefully observe when I am talking about this I am talking about the user ID. So one device will have one user ID. So if a user going to have 6 devices, so user ID will map to only one particular root node, so that root note will maintain only 6 possible entries, 6 nodes ID here. But all these devices also have node IDs, they will also be maintaining on an average only whichever actually is they will become root for some of the people and they will be maintaining.

So on an average there will be one entry which will be maintained for each node which are the core nodes. And of course the leaf nodes will be now spread around where user ID to endpoint addresses will be maintained. There node IDs will not be there. So there user ID to endpoint address, but they will not be maintaining any entry on their own.

So on an average still remains one entry per each node ID and I am assuming now that for even leaf node we will be maintaining node IDs, they will have node IDs, but this will not be used. So that way but the problem with this kind of system is that other people will be able to figure out that can also reuse this particular node ID that is only problem. If it is hash than there should not be an issue. So ideally speaking everybody should participate in the core node and then of course should actually use to participate in the base layer DHT.

(Refer Slide Time: 28:39)



So finally in case If you are actually using the leaf node if it is going to use node ID then it actually can maintain routing table in an optimum way. Otherwise, it actually need not maintain and normally node ID if it is maintain the problem is somebody a duplication can happen and for avoiding this has to participate in a base layer DHT, okay.

So at least base layer because any new node ID any node which joins and which when actually searching or whether duplication of the whatever new node ID which it has generated. It can be attach to search in the base layer and base layer means everybody is participating iot can duplicates can be avoided.

In case if it is a leaf node even in base layer, then that node ID not be there. So the best strategy is that because there is no use of maintaining routing table. Only best what it can do is it can get routing tables from whatever entries are there and based on that based on only on neighbour tables it can actually do the updation and optimize it.

It cannot do anything because it does not have any node ID. So it cannot find out what should be the routing table to attach to essentially some other node ID which is participating in DHT as a proxy. So essentially if we copy the same routing table there so we do not even seek from everybody.

One alternative it can directly contact that guy, typed that same routing table and use that for forwarding it further and every time keep on periodically doing it. If this guy dies off it can fetch

some the routing table from somebody else and ideally it should fetch from somebody who is closest to it based on the neighbour table. So that the minimum network load is going to be there when leaf nodes are searching.

So with that we actually end our discussion on this multi-layered DHT and next we will be looking into how we will actually do resilience in the key-value pairs which are stored in a DHT layers, so how that gets actually implemented. So that will be the topic of our next lecture.