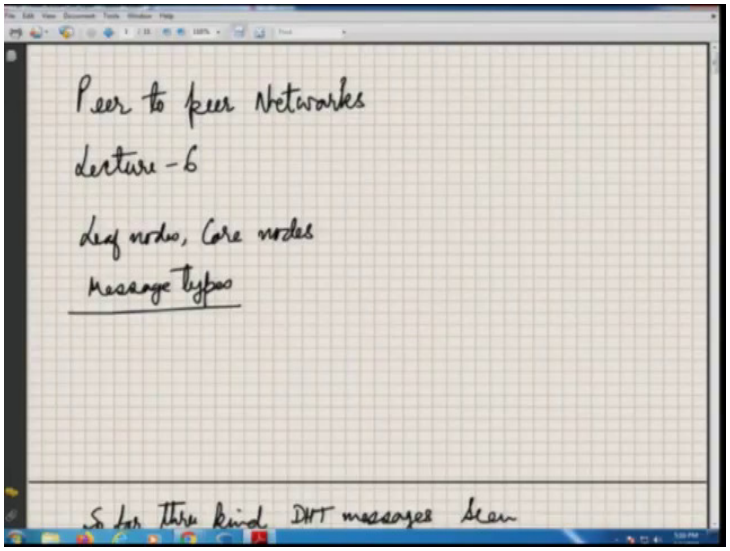
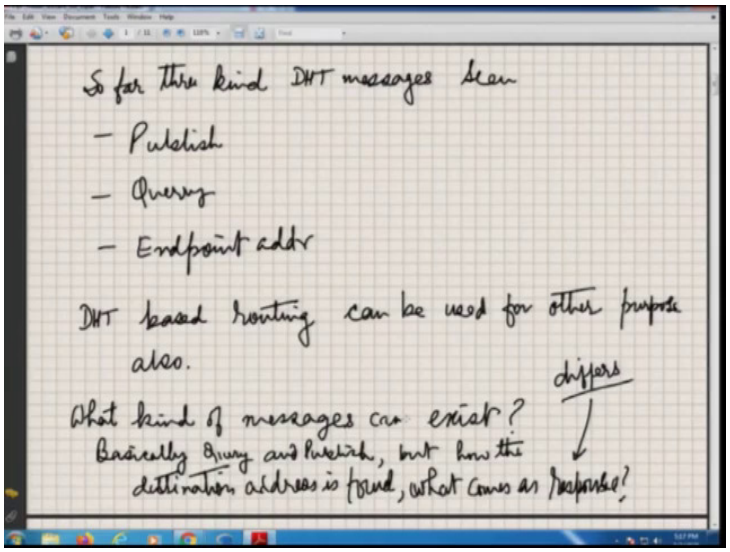
Peer To Peer Networks
Professor Y. N. Singh
Indian Institute of Technology, Kanpur
Department of Electrical Engineering
Lecture 6
Leaf Nodes, Core Nodes and Type of Messages in DHT Networks

(Refer Slide Time 00:23)



So, welcome to the lecture -6 for this MOOC on peer to peer networks. So, today we will be talking about the difference between leaf nodes and core nodes and how they operate. And we will also look at the different kinds of message types and how they can be classified.

(Refer Slide Time: 0:42)



So, so far whatever we have done till lecture number 5, we have looked at the various kind of DHT messages. So there were 3 basically DHT messages which we have looked at, 1 is a publish, where we were trying to put something into the DHT index in basically distributed hash table There was a query, so we are trying to find out if something exists in their distributed hash table or not. And third type which we looked at when discussing voice over IP over a peer to peer network was what is the corresponding endpoint address.

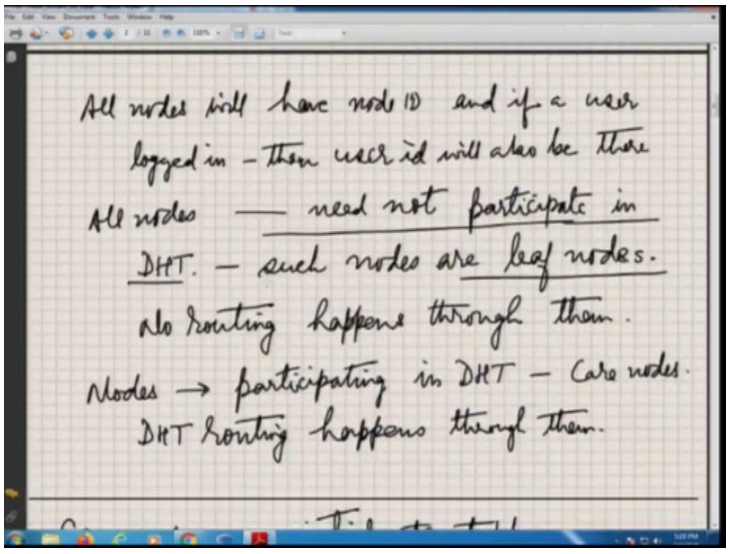
So there was no entry in the DHT distributed hash table but DHT routing was essentially used to send a query all the way to a destination and then asking what is the corresponding what is the endpoint address of that particular device. So, but we had assumed so far that all nodes are participating in distributed hash table.

Now, one important thing is it is not necessary that all nodes should participate in DHT. So there is a possibility that some nodes may not participate in DHT. And of course, DHT routing for example, here we have used to identify endpoint address directly from the destination, there is no entry in DHT.

We can also use it for some purposes, so we will be looking at those also. Now we have to identify what kind of messages can exist because we need to know and based on that we have to identify message types. So these have to be programmed when we are going to build up a system. So basically we will have query and publish, these are the basic two things.

In fact, the third one endpoint address is also query, but the query is not resolved from a distributed hash table or just not from the index, but it is being resolved directly by the destination. So, depending on what kind of destination address we are using, we will figure out what kind of response is going to come and that is essentially what will create various kinds of messages.

(Refer Slide Time 02:52)

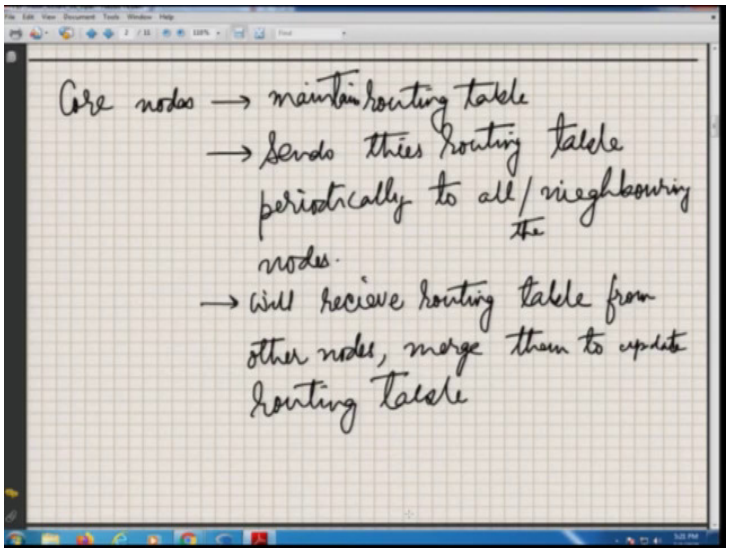


Now, we actually do know that all nodes will have some node IDs. And if a user is logged in, then it's user ID will also be exist, will also be there. It is possible that a node ID is there but no user has logged in, so the client is still running so node ID will be there but there is no corresponding user ID but the node itself is still participate in DHT that is still feasible.

Now, there is also a possibility as I mentioned that all nodes need not participate in DHT. And all such nodes which do not participate in DHT, but they are in the network they can provide service once in a while, they can also get requests for service, these kinds of nodes are known as leaf nodes, because they are just attached to some of the nodes who are participating in DHT.

So, they cannot route if a query or something comes to them, they cannot route it to anybody else. So they are not intermediary in routing, they are either originating or they are termination point of a request of any kind of message. No routing happens through them, but now it means the other nodes which are participating in DHT through whom the routing does happen these are called as core nodes. So there are leaf nodes and there are core nodes. So DHT routing does happen through the core nodes.

(Refer Slide Time 4:20)

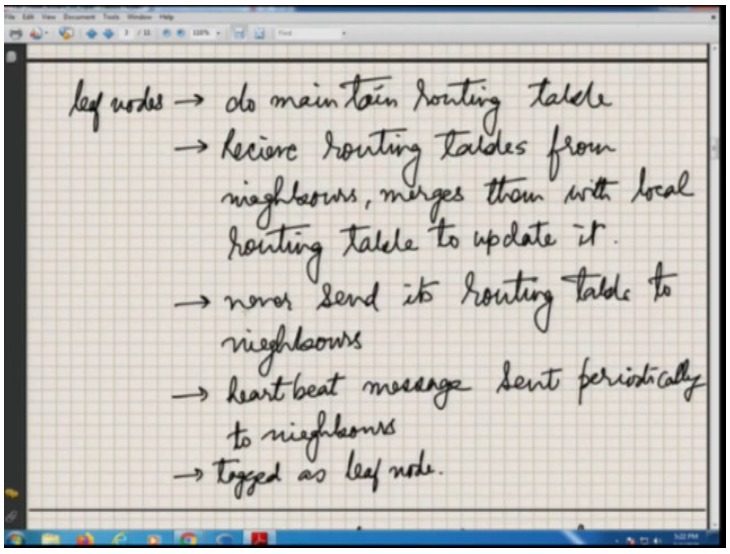


So, what the quarter notes are going to do, so obviously one thing if they are forwarding the messages they must be maintaining routing table and once they maintain the routing table this need to be updated periodically. So, as I had mentioned earlier also, the way we maintain or update routing tables is that everybody will send its own routing table copy to all of its neighbors so that actually means if you are a node you will be getting the routing table copies from all of your neighbors. So, they may contain nodes which may not be exist in your routing table and if such nodes are better candidates to be put in a routing table you will replace them.

So, your neighborhood relationship will keep on changing and you will become more and closer to the most appropriate node which should be there in your routing table in the whole node ID space. So, this way ultimately routing table stabilizes over time. So, this is basically a routing table action method which is used to dynamically evolve the routing table at all the nodes. So, if some node dies off, some new nodes join in, routing table again automatically optimizes and comes to a steady state of situation over time.

So, one of the functions is that any node where it happens is we call it overlay management. This is when the routing tables are received from neighbors you merge them into your own routing table making it more and more appropriate for the current node ID.

(Refer Slide Time 5:57)



Now, coming to the leaf node, so whatever I have told so far was about core nodes. Leaf nodes also do maintain routing table because if I have a query I am a leaf node I cannot, I have to at least route whatever queries which are originating from me, whatever queries are coming to me will not be routed any further, they have to be essentially handled by me directly.

Normally no DHT query will be coming to me. I will be only being called through only DHT address. So, leaf nodes cannot be reached by their node IDs, they have to be reached by their endpoint address. But the endpoint address to node ID mapping of these guys can be maintained at the route node of the node ID of, route of the node ID of leaf node.

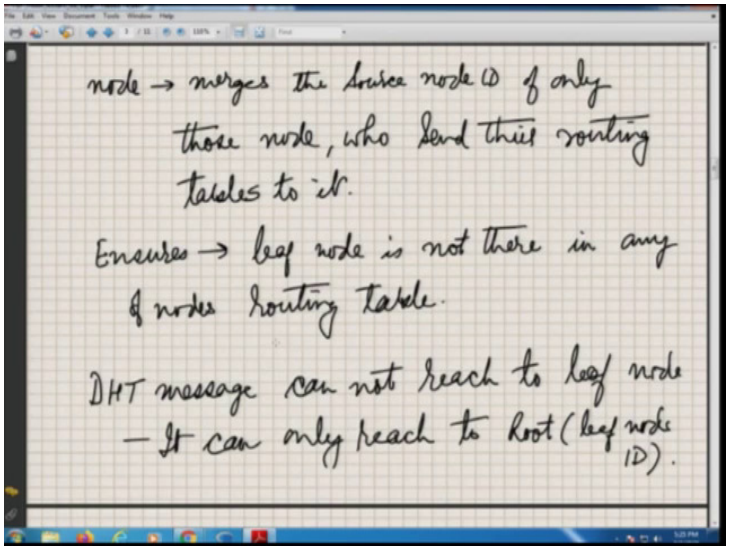
So, then if they do maintain routing table so what they do is they do receive routing tables from the neighbors which are there in the current routing table. So, it all starts when you log in, you will connect to a bootstrapping node so bootstrapping nodes routing table comes to you. Then, whatever nodes which are listed in bootstrapping nodes routing table you will get the routing table of those nodes. So you will now get in the next time their neighbors and so on, you will keep on updating and you will again optimize a routing table for yourself. But you will never be forwarding it any queries actually.

So it is important that you never send your own routing table to others because you are not forwarding so there is no need to send your routing table, you are only need to maintain your own local routing table. So when you are sending your routing table to neighbors, you are actually now advertising your reachability to other nodes so that somebody can probably

make you a next hop node in the routing table. So leaf nodes will never announce their routing table that is one exception, which is there. And it will also send heartbeat message periodically to all the neighbors so neighbors know this guy is alive but only heartbeat is coming but routing table updates are not coming that means that person is a leaf node.

I have to send him a routing table, but when I know that he is alive, I am not supposed to consider that node's node ID for my routing table optimization. So that is a trick which is used so to maintain the leaf nodes and the core nodes. So these are essentially tagged as leaf nodes. So therefore they are never considered for updating the routing tables.

(Refer Slide Time 8:24)

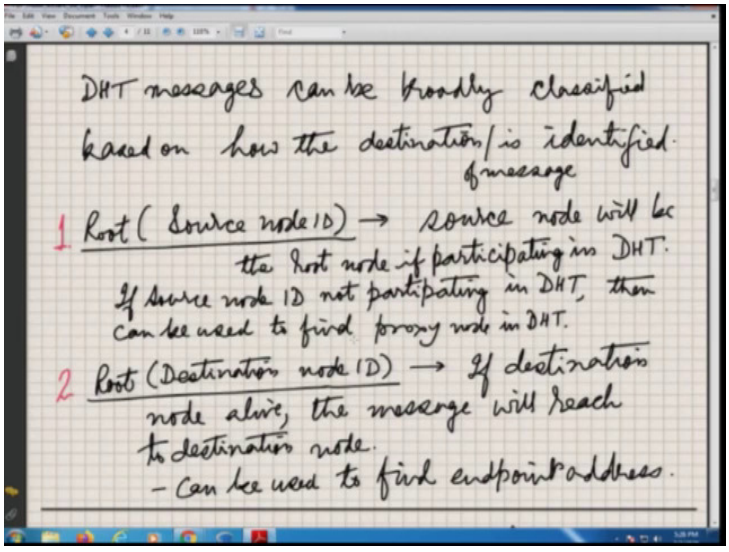


So every core node need to merge only the source node IDs of only those nodes who send their routing tables to you. If nobody is sending, somebody is not sending a routing table to you, but he is sending a heartbeat package, his routing table, his source node ID should not be used, should not be merged in my routing table.

Anyway, he will not be sending routing tables of that merger of that does not arises. So this ensures that leaf node is not there in any of the nodes routing table who are the part of DHT. So DHT messages cannot reach the leaf node it can only reach to the root of the leaf node.

So in that case, the leaf node itself can make a query to its own root. But whenever it is generating a query so return address is not going to be node ID, it has to be an endpoint address which is going to be IP address, port number and transport triplet which is required to be put as a source identifier.

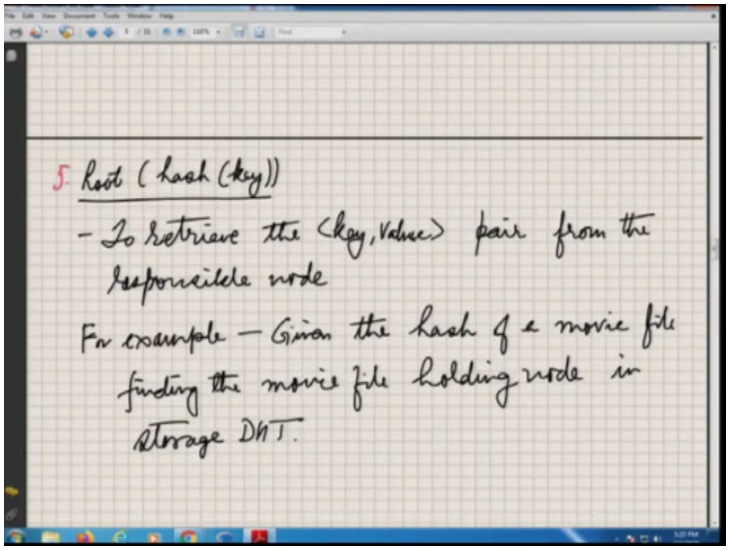
(Refer Slide Time: 9:31)



So based on this understanding of root and leaf node, now let us come to how the DHT messages can be broadly classified. Now this leaf node is a special node. If they do not exist, if they everybody is core node, then one of the messages will never be required. So first thing is that what kind of identifiers for the nodes does exist?

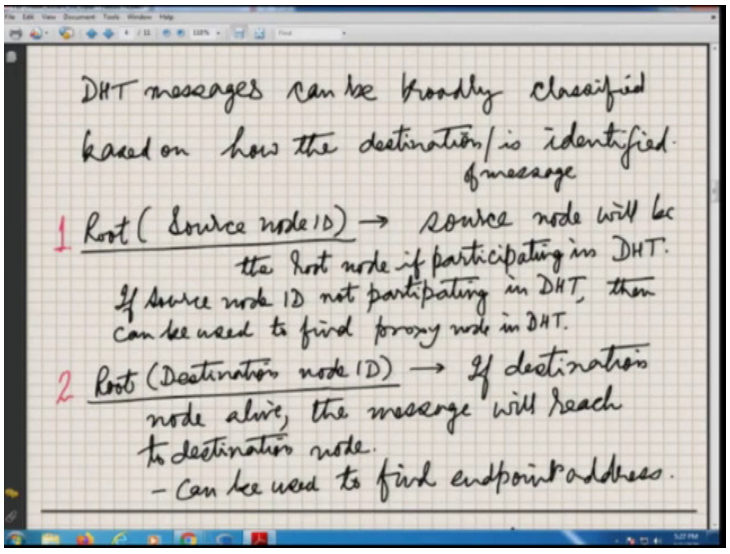
We actually have node, source node IDs, we have destination node IDs, we have source user ID, the guy who has logged in onto that system. So, same user ID can be logged in at multiple node IDs also that is feasible, so that we have to also take care. Then we have destination user ID and then we will have ultimately at the end.

(Refer Slide Time 10:16)



So these are basically for example, in the distributed file system, what is going to be the I node address. I node it is description of that particular file or the directory in the file system space. So these are basically five messages which will exist, let us move to the first one.

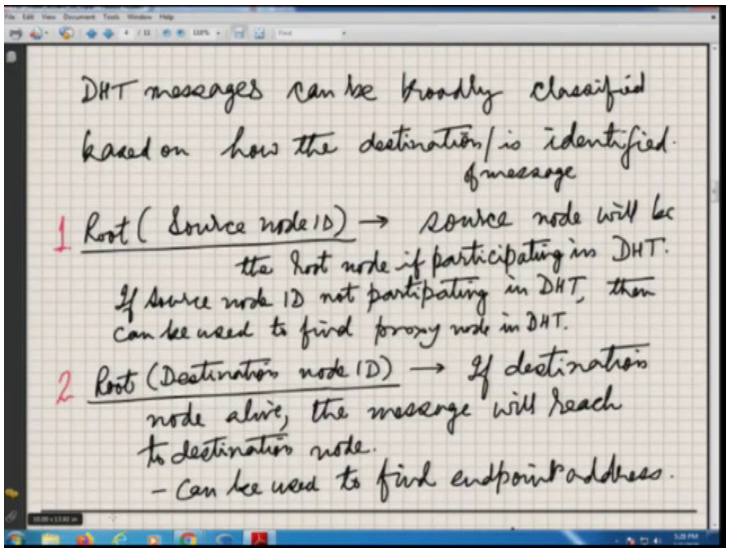
(Refer Slide Time 10:37)



So in this case, the root of source node ID has to be found out, who will be the source node ID is actually existing. And if it is participating in DHT, root of source node ID will be the source node ID itself because that is closest to itself. So source of node ID, source node will be participating in the DHT then it does not make any sense. So it will be root of itself, so this kind of message will not be actually used. But if the source node is not past is not participating in DHT, in that case if it is going to fire a query, any kind of message to this destination, it will go to the root of this particular source node ID because it's not part of DHT.

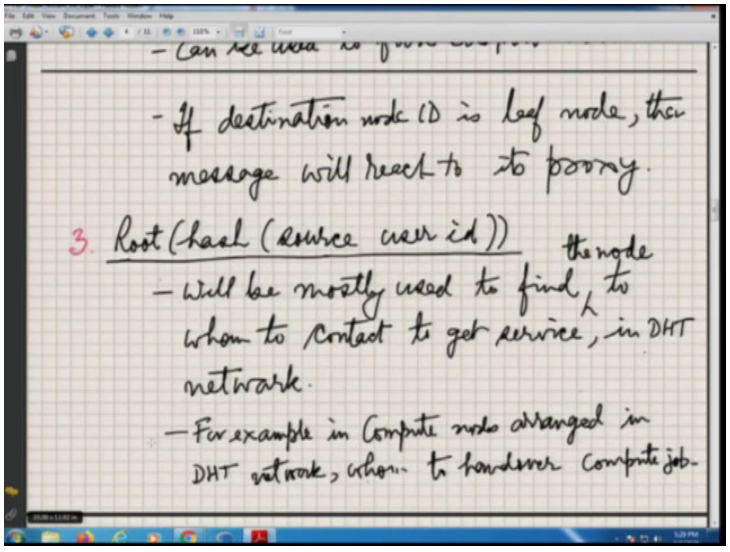
So, in the DHT whichever is a node ID, which is closest to it, the query will essentially go to that or the message will go to that and we call it as a proxy node of DHT. So, if you want to find out who is your proxy, because you are a DHT, so probably, you are not in the DHT. So in the DHT network somebody is there who can receive messages on your behalf. So that will be acting like your proxy. So identifying those, for that essentially this kind of destination address can be used. We will look at the examples of each one of these scenarios.

(Refer Slide Time 11:56)



2nd kind of messages will be when we will find out that we will look at the destination node ID, now remember I am not talking about user IDs here, I am talking about node IDs. So root of destination node ID that can also be used as the destination address. In this case, if the destination node is alive, the message will certainly reach to that particular node ID itself. So this typically will be used if you want to find out what are the corresponding IP address, port number and transport to communicate directly to that node ID to their destination node. So endpoint query, for example will come in this particular category.

(Refer Slide Time 12:39)

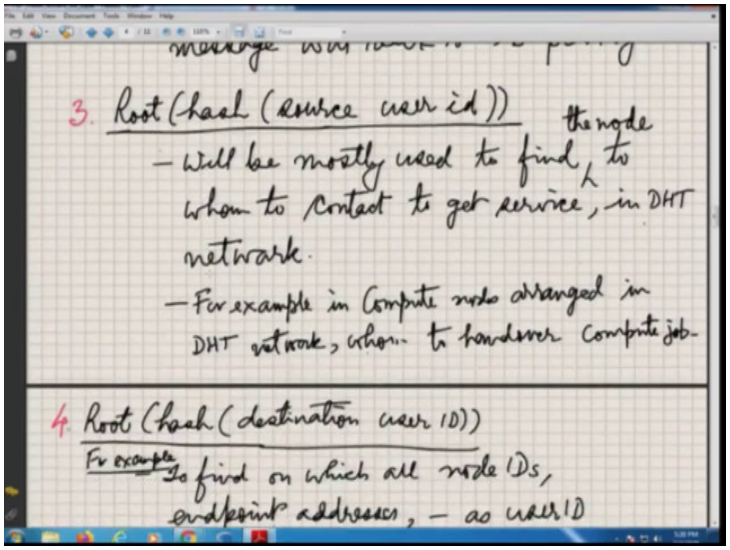


So now the third kind of thing, there is also an issue if the destination node ID is the leaf node, then the message will reach to its proxy. That is a very important thing, if the

destination node ID is leaf node, then the message will reach to its proxy. So this basically you want to communicate some leaf node. You do not know its endpoint address, you can actually deposit the message with the proxy and that leaf node will use the first kind of message to retrieve. So, this will be a publish and the first one will be the query thing so by which it can retrieve the messages destined for him in the DHT.

Now this kind of scenario we will be using when I will talk about multi layer DHT, I have not talked about that thing so far, but for peer to peer mailing system design in Brahspati-4 (B4), we also have peer to peer mailing system, there we are actually using this kind of structure. In fact, there is a more refined version which we have implemented.

(Refer Slide Time 13:36)



So the third kind of message is when now I am not looking at node IDs, I am looking at the user IDs. So if the source user ID so my user ID is there. So I have to compute the hash of that. It is like a key, it is not a node ID so I cannot directly compute the route. So I will do the hashing, whatever hash string will come hash ID will come, I will find out the root so who is closest to it.

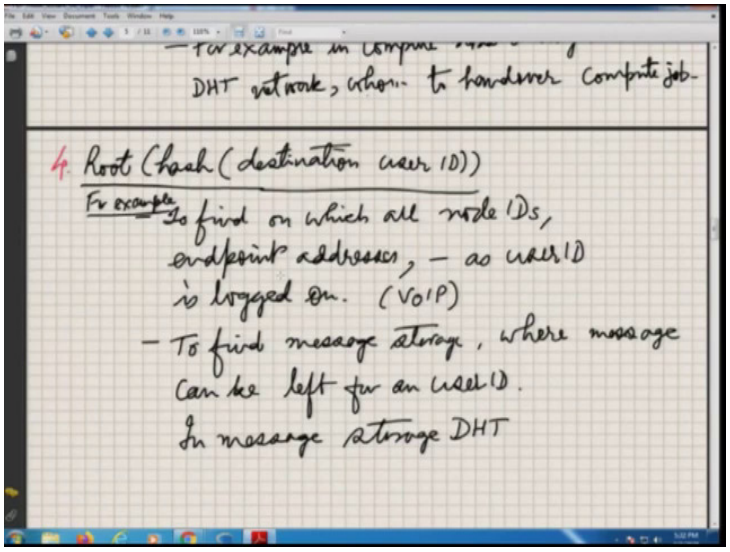
It will basically be used to find out the node to whom to get a service. So for example, I as a user ID would like to find out, want to do some computing, there are various nodes in the DHT, which are providing compute service, a virtual machine. So to which virtual machine I should deposit my job.

So to uniformly distribute all the requesters across all the nodes who are willing to provide a virtual machine service, I will now compute the hash of my source user ID and then we will

find out the root of that in that DHT which is of the virtual machines and we will then send a message to them that I want to now put a job to you. So this message will essentially reach to the root so that way compute jobs generated by all the nodes will get uniformly distributed across the DHT, across the nodes participating in the DHT of virtual machines. So,this kind of example, which can be used.

And the fourth kind of message will be when I am going to use a destination user ID and I will compute the hash of that destinations user ID and we will submit, we will find out the root of that and send my message to it.

(Refer Slide Time 15:22)



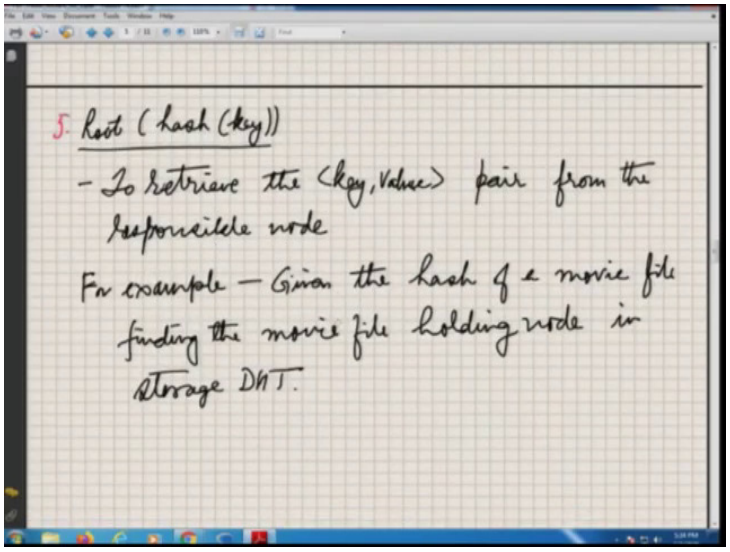
So, this is basically, for example, you have this in a Voice over IP. So I am supposed to make a call to a user not to a device. So I will search for destination user ID, which is the phone number or the email ID of the other person to whom I want to communicate. I will compute the hash of it and then we will try to find out the root of that one. And this message will go to the root so all the devices on which that destination user ID is logged in would have registered there.

They would have registered for this user ID, this is a device node ID, this user ID, and this is a device node ID. So I will be able to get to all the list of all the device node IDs on which this user is logged in. So, this is an example of query which can be made with this kind of scenario.

We can also have a kind of, I want to send a mail to a certain user, so in message storage DHT, the nodes which are participating in message storage so whoever is not participating will become a leaf node for this message storage DHT. I can again find out using destination user IDs, hash value finding out just root node. So, the node which is there is responsible for keeping index of the fragments, all fragments of the mails which I deposited in a separate storage space.

I will talk about it when I will come to the mailing system or maybe I can directly dump the message on this message storage, the root node in the message storage DHT and later on the destination user ID can always withdraw it. And since it has been digitally signed, so nobody in between can read it, only the destination user can actually read this particular message.

(Refer Slide Time 17:05)



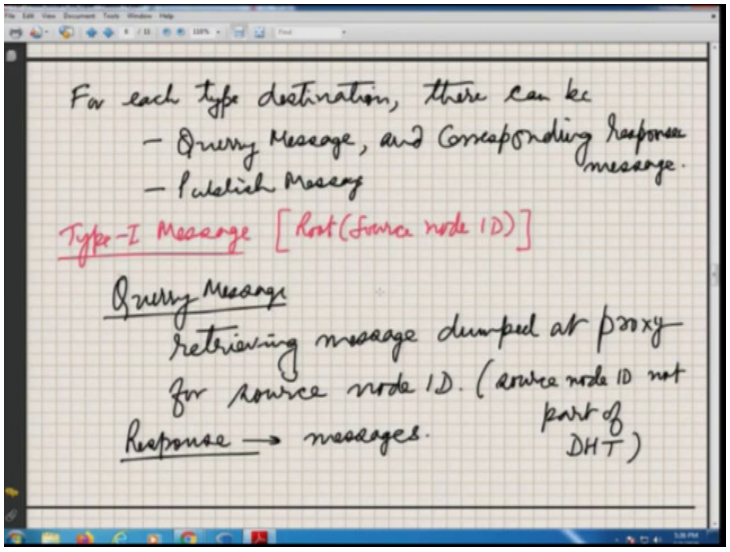
Now, the fifth kind of message is where I will be taking the key value peers will be stored. So, you have a distributed file system for example, so my file is again I will be talking about this file system thing later on how to implement it in a distributed system or maybe a file which contains banana, so I actually deposit a file and a banana there as a key word. So, whenever everybody who has something to say about banana will deposit the file to a hash of the banana string and then finding out the root node and dumping there. So, it may not be exactly files, it may be pointers to the various nodes in the network where actually these files are located.

So, when somebody searches for banana, he will also make a query to the same thing, he will get all key value peers. So, values will give the pointers where you can get the files which contain banana as a keyword. Then I can directly talk to those nodes and retrieve the files. So, that is another example where key value peer can be used. We can actually use it for DFS, we can use it for identifying a service.

So, for example, you want to find out lecture for P2P MOOC so, you can set P2P MOOC by Y.N. Singh, put it as a key, compute the hash, find out the root node and then there it will be giving you an index from which place I will be able to download my lectures. There may be multiple nodes hosting my lectures, but we are going to do a better job than this actually, the mechanism which I am showing where nodes are hosting, it will be hosting all the content in the DHT itself where the resilience is built in.

Even the nodes go away and they join in, my content will still reside in the network and can always be accessed, ultimately we have to build that kind of system. Now, for each one of these type of destinations which I have, there can be query message or there can be a published message.

(Refer Slide Time 19:02)



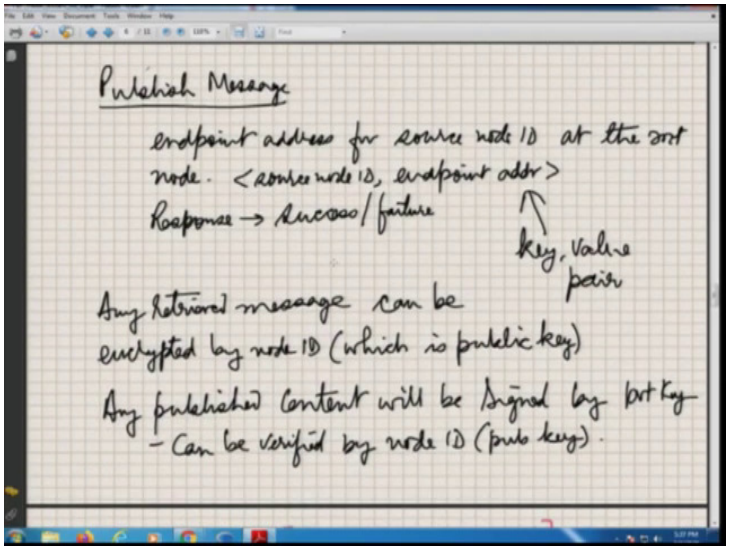
So I think now I am becoming more explicit for each type of message, I am also giving them a type number now and I am going to now try to find out if it is a query message for what purpose probably it can be used. If it is a published message, what purpose it can be used?

We have actually identified in the beginning that there are only two basic messages; either it is publish or it is a query, there are only two things which has to be done. So, type-1 message

again come back to the same thing, source node ID being used and I am finding out the root node of that. If source node ID is part of DHT this itself will be the root node. So, if there is a query message, so in this kind of category with this kind of destination, it will be mostly for retrieving the messages dumped at the proxy for source node ID. Source node ID will only be making query for this, there is nothing else which can be done and source code ID is not part of the DHT, so for a message storage for example.

So response will always be the messages which will be coming back which I have to then decrypt and then find out what they contain.

(Refer Slide Time 20:13)



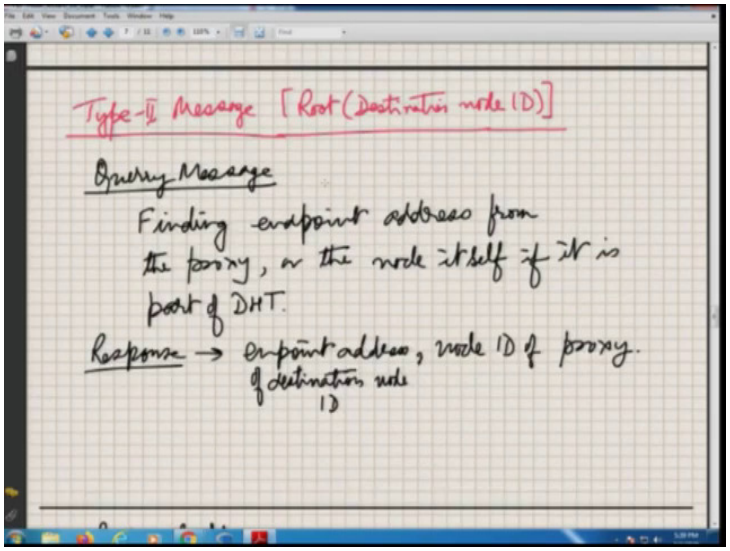
If there is a published message in this thing, so what I will be publishing is, it means this will be done by the node because earlier I was retrieving it, but I have to tell it to others that how they can communicate directly to me. So, I may actually publish the endpoint address of mine at my root node. So this will typically be a source node ID dash endpoint address key value pair, response will only be success or failure. So that is a typically a publish message for type-1 messages.

And any of the retrieved messages can be encrypted by the generator through the my public key so that I am the only one who can actually decrypt it using my private key, so that is typically through which messages can be transferred, is we call an intermediary based transfer actually of the messages, while keeping them encrypted and secure.

And of course, any published content which is coming to me has to be digitally signed by the sender, so I can verify who the guy who has actually sent it was. So it is a public key of the

node ID or the source ID depending who should I need what kind of information about the sender. If it is the user which is important, it will be public key of the user, which will be used to verify and the private key will be used to digitally sign it.

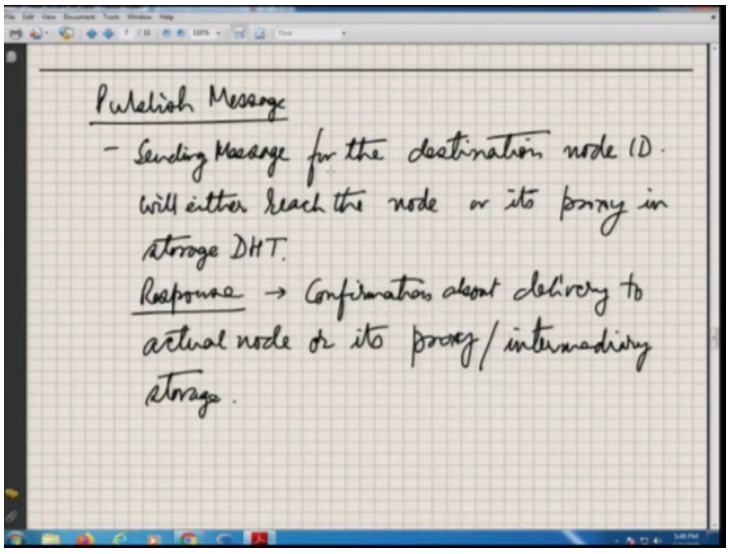
(Refer Slide Time 21:38)



So type-2 messages, so type 2 messages are typically I will be now using destination node ID and computing the root node of that that will be the destination. If I want to do a query on this kind of thing. So, why I am doing it? So query is normally for finding the endpoint addresses from the proxy or the route of the destination node ID and if destination node ID is not part of the DHT.

And if it is part of the DHT, I am just trying to send a message to the node itself. So either the response, you will get the endpoint address or you will get the node ID of the proxy of the destination node ID. Only two things depending on, if the destination node ID is part of DHT or not.

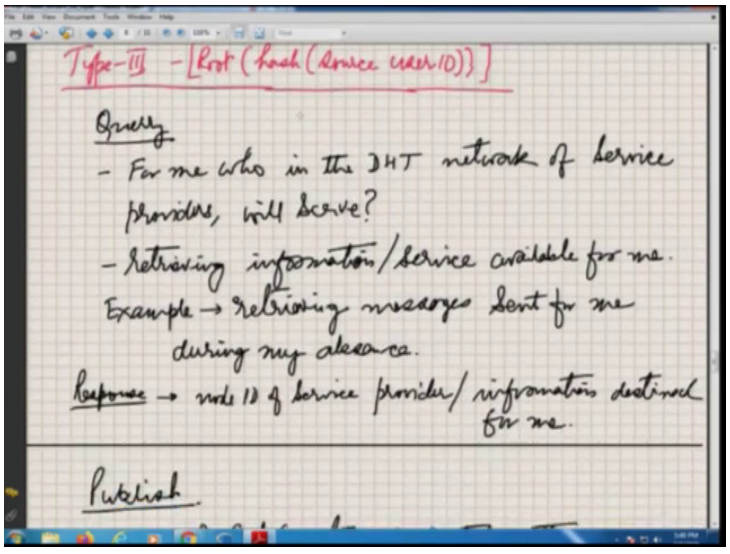
(Refer Slide Time 22:20)



When you do a publishing, so typically it will be sending a message for the destination node ID, it will either reach directly to the node, or it will reach its proxy in the storage DHT, or the responsible node for that particular node ID. So response will always be confirmation about delivery to the actual node or its proxy or intermediary storage.

So this is typically when I want to send a mail to somebody, I will find out from the user ID, what is the corresponding node ID of the device, and then I will use that to directly send a message to him. So he will receive the message and that is a publishing, it is not going DHT but it is being communicated directly to him.

(Refer Slide Time 23:04)

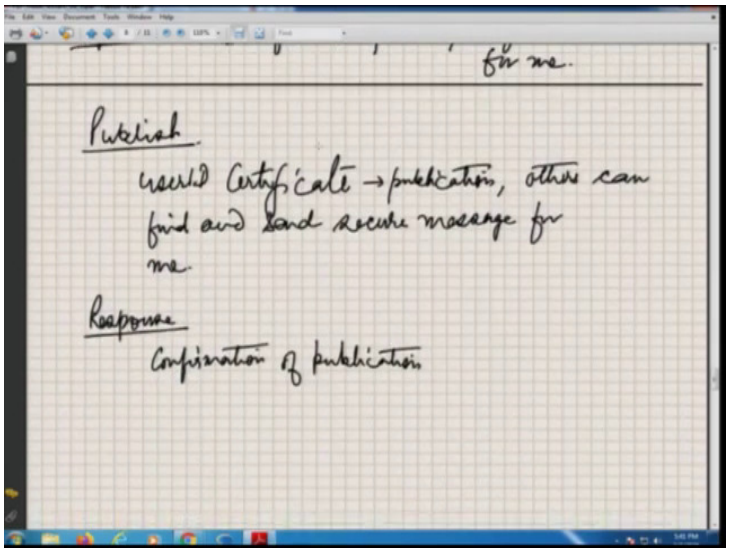


If I am going to have a type-3 message, in this case, source user ID is being used not the node ID now, and we compute the hash of it and then find the root. So, if there is a query, so query typically will actually example is for me who is in the DHT network of service provider who will provide me the service. So that is a question being asked.

So in a DHT of virtual machines, service provider who is going to provide me the virtual machine for my compute or you can retrieve the information or service available for me. Example is retrieving messages sent for me during my absence typically. So it is a user ID specific message, not with the node ID.

And response will be node ID of the service provider or it will be the information, which is destined for me. So VM was a service provider DHT and VM service who is going to provide me we were asking that.

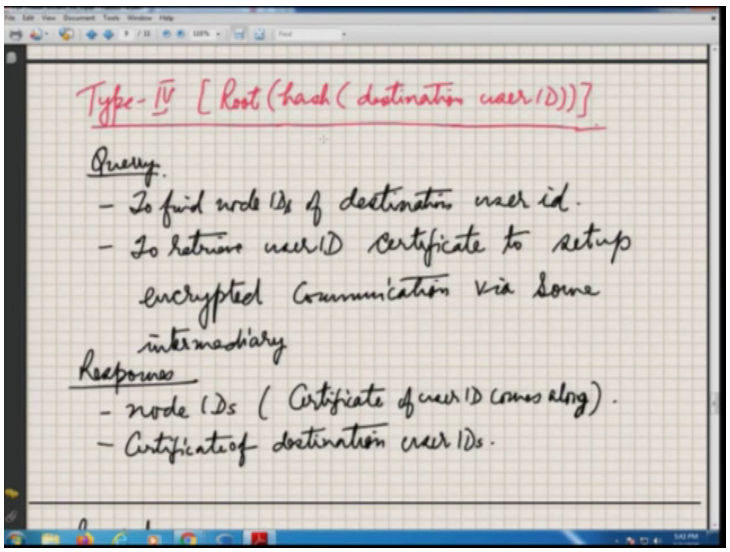
(Refer Slide Time 24:08)



So let us come to the next one, the published thing, what you will be publishing here. So remember, I am actually talking about type three message which is based on source user IDs, hash and then it's root.

So, normally you will be publishing user IDs certificate, you will publish it so that others can find and send a secure message for you. So they will always be searching from your source user ID, they need to get your certificate first. So even if you are not alive, they need to get your certificate so they will get a certificate, but you have to publish your certificate with source user IDs, hash value and then finding out the root of that. So it will be only a response will be confirmation of the publication in this case.

(Refer Slide Time 24:48)

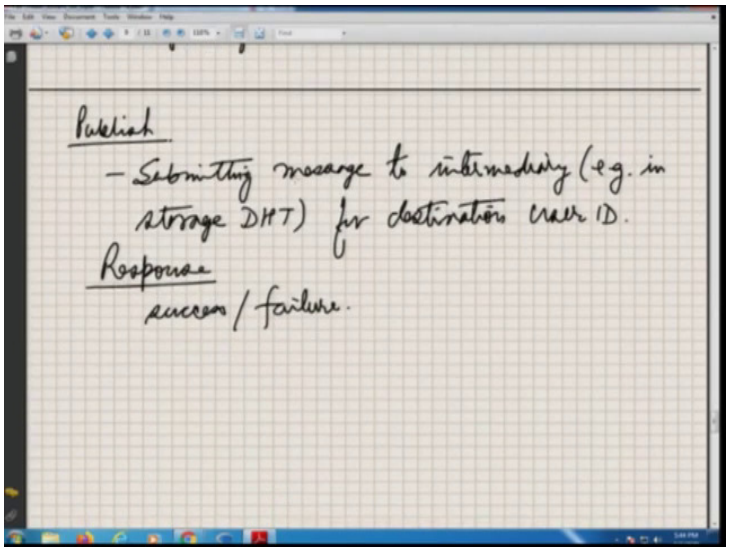


Then there is a type-4, again I am looking at the example. So query type messages, so here the destination user ID is going to be used. I will use a hash of that and find out the root node. Remember, it is a destination user ID, what kind of queries I can make. I can make a query to find out the node IDs, which always the destination user ID is logged in, Voice over IP is an example.

Or to retrieve the user ID certificate to set up an encrypted communication via some intermediary. I can also do it to find out who is going to be the intermediary in the message stored DHT. So responses will be either node IDs, certificate of user ID will come along with that. So whenever I am searching for user ID, or it can be purely a certificate of destination user ID, so both things will come depending on what kind of combination has been published, whether the user ID has published only the certificate or the node ID.

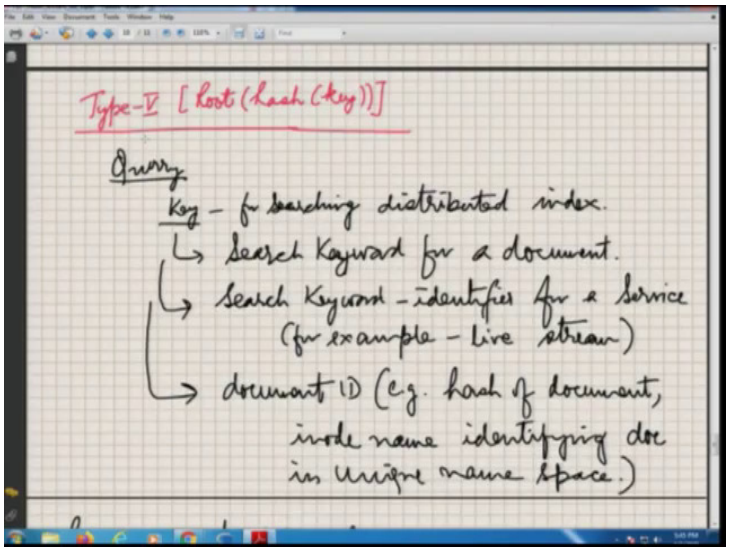
Normally, I will prefer that user ID should publish certificate separately and user ID should publish their node IDs separately. With each node ID it should not publish a separate certificate, so there is only one certificate so there should not be multiple copies. So, I will actually save on the unnecessary replication of data in that case. So trying to fragment and keep separate key value peers. So there should not be key value one, value two, there has to be key value, key value, key value kind of thing which should be used. A value can be a certificate; value can be node IDs so that can be encoded in the messages. So that is typically the response to the query.

(Refer Slide Time 26:29)



If we want to publish, so this will typically be like I am using now type-4 which is destination user ID, you want to send a message to somebody and the guy is not alive, it has to be done through intermediary. So you can submit the message to the intermediary in the storage DHT for destination user ID which that guy will retrieve later on. So only responses you will get is success or failure. This is an example of publish for the message type-4.

(Refer Slide Time 27:03)

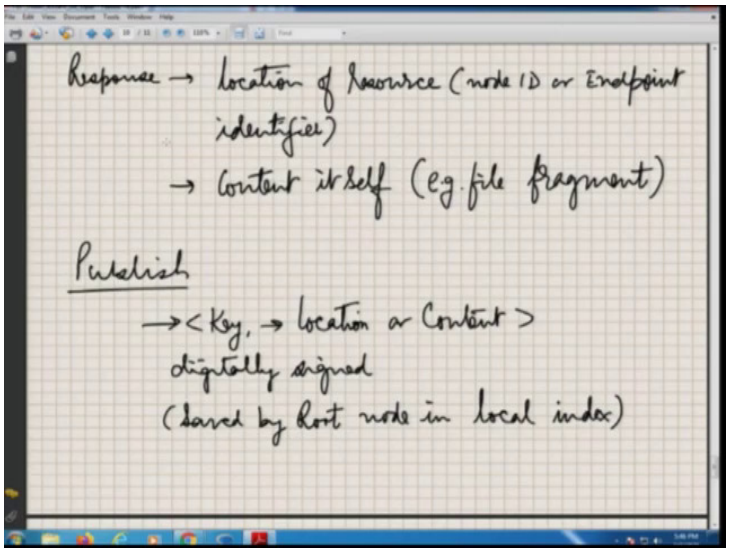


Now coming to the message type-5, So, this is very generic and this is based on key. So key will be essentially for searching the distributed index. So it can be a search keyword for a document, it can be search keyword identifier for a service, for example, a live stream, so I am actually delivering a lecture you want to find out so this will be identified by a live stream identifier, you will search for it and you will go to the hash of this particular string and the root of that and we will tell who all are the guys who can provide me the feed for this particular lecture that is an example of a keyword identifier based service discovery.

OR, it can be a document ID, for example a movie, so movie's hash value is going to be the key and movie will be there with people. You will search with the hash value you will find out where is the movie in the storage DHT itself, or which guys are actually holding the corresponding movie.

So once the movie gets downloaded do you have to compute the hash and then verify whether that actually move is indeed the same one, and then you can play back that movie. So that is a key value search based thing, so these are typical queries, which will you will be making.

(Refer Slide Time 28:15)

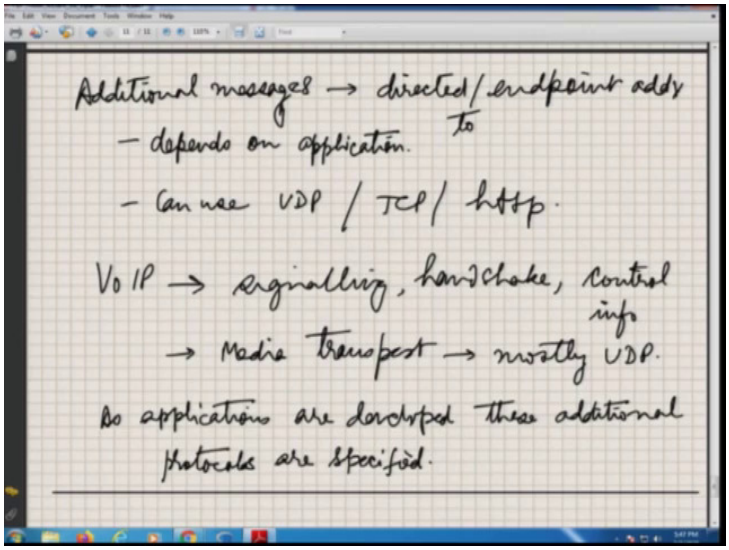


Responses will be the location of resource, node ID or endpoint identifier or the content itself any one of the two. You can also publish actually the same thing, it will be key and you will be either putting a location or the content itself will be published. It has to be digitally signed depending on who is publishing it.

It has to be saved by the root node in the local index there with the root node. And of course, they can be now directly additional messages once you know the endpoint address, you might have to do a peer to peer negotiation for example, call setup, Voice over IP call setup or maybe identifying an API by which you can do a VM service can be subscribed to, by that you can actually dump the messages.

So these are communications which are not passing through DHT, but which are direct Point to Point communications. So, there has to be separate messages which have to be identified for this purpose. It is basically a protocol definition for the service.

(Refer Slide Time: 29:13)



So in Voice over IP for example, signaling, handshake, control info, etc has to be defined, we have to define whether it will be on UDP, TCP or HTTP kind of thing, we have to define media transport which is mostly the UDP actually for Voice over IP systems.

And as you develop applications, you have to essentially now identify these additional types of messages and put in your documentation for implementation. So, that is typically the way the message actually does work. And I broadly classified various kinds of messages which are used in the system.

We will look into now multi layer DHT here further and then more on DHT algorithms which we will be doing, we have done so far on CHORD, but cod is not the only algorithm there are many others. So, I would like to discuss ultimately by after discussing few algorithms, a more generic framework on which the distributed hash tables actually operate. So, I will continue with the more content in the next lecture.