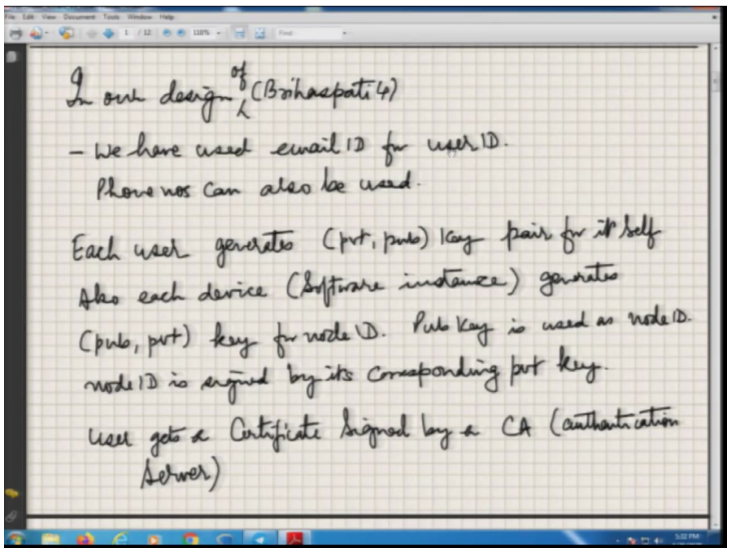
Peer to Peer Networks
Professor Y.N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Lecture 05
Implementation of Voice over Internet Telephony in P2P Way

So, Welcome to the lecture number 05 for Peer to Peer Network MOOC course. So, in this lecture we will be actually using whatever we have learned in the earlier lectures and , I will try to describe how the Voice over IP is implemented in P2P way using peer to peer mechanism in our system which we have been building.

So, as I have mentioned earlier also, we have been building a project called Brihaspati-4 and the code actually resides on GitHub, so the Voice over IP is implemented in that. And, as I go along, I will tell what innovations we actually have done. So, few of them actually, I will now tell today.

(Refer Slide Time: 01:03)



So, first thing in this Brihaspati-4 design, we have used the email ID for the user ID. In fact, phone numbers also can be used; though then it will be like the conventional SIP telephony. So, normally there has to be unique identification of every user and which has to be unique to everybody. It should not; no two persons can have the same ID. So, phone number is one such thing. Similarly, email ID could be one such thing. So, we have actually implemented with the mail ID, but system can be upgraded for phone number. Conceptually, the technique remains the same.

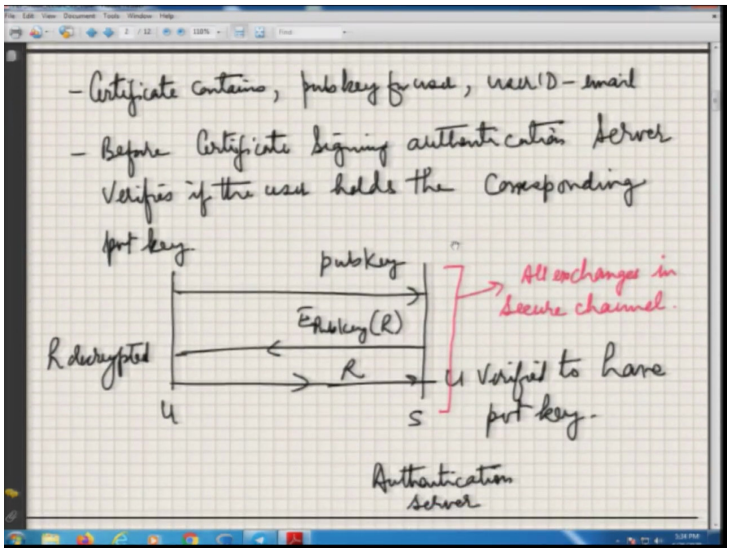
So, each user is supposed to generate a private and public key, a pair of it. And this will be for itself. And this will be basically user identity this is for that and for also each device. So, maybe you are going to have a phone, you maybe have the same thing on the laptop. Maybe you want to also the same instance running on the similar user ID on a phone, SIP phone or it might be running on a desktop machine but you want the same user ID to be there.

So, it is possible that when somebody makes a call to you, so the ring will come on all 4 places, and you can pick up the phone from all those. But in a system, we should be able to uniquely identify each device. So that is why we every user, every device will have two identities. So, one is the user identity which will be same depending on who is the user. So, if I am as Y N Singh logged in into my phone, same Y N Singh, I am logged into my laptop, so my user certificate private-public key pair for user ID will remain the same.

Okay, but for device, this will keep on changing. So, and the public key part of the device ID which will be used as a node ID or maybe if the node ID has lower number of bits, we will compute a hash of it and that will be used as a node ID, and this node ID will be signed by the corresponding private key of the node. So, node is the device.

And why we do it is because we want a random generation of node ID. This we had discussed in the earlier lectures. And the user will generate the public-private key. This has to be signed by a Certification Authority (CA). And we call it authentication server in peer to peer SIP RFC or in any peer to peer system. So, we also have a Certification Authority. We call it B-4 server; B-4 stands for Brihaspati-4.

(Refer Slide Time: 03:38)



So let us move ahead. Now, the certificate will contain the public key of the user. The user ID which will be unique ID in this case the email. So, before the certificate is signed by the authentication server, the authentication server needs to verify whether the user who actually has given the corresponding public key; private key is never shared with anybody. It is with the user only whether that guy has the corresponding private key also or not with him.

And secondly, it will send an OTP over the email ID in this case or it can be SMS, if it we are going to use a phone number. And that OTP has to be put in and then we will be verifying whether the guy actually have access to email ID. So, if you have access to certain email ID, you can be assumed to have that identity email ID identity, actually. So that is what we use.

And in this case, this is the user on one side; this is the authentication server. So, 1st thing you will be sending a public key of the, to the user to the basically, this server and this server will now send this public key. This public key is essentially sent by the user. This has to go

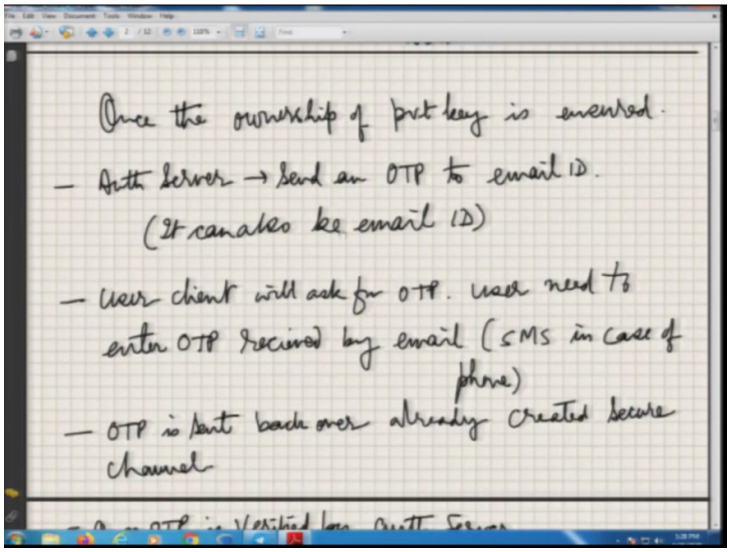
into the certificate. So, a random number will be generated by the server which will be sent back by encrypting this with public key and this user will be able to decrypt this R, the random number. And, this R will be sent back again back to the server.

Now, this server an user already have HTTPS channel. So, this server certificate has been used to create a secure channel, this is going to happen over this. Except for this Epubkey (R) the second step need not require that thing because this is anyway going encrypted back on the public key which has been given by user. This first step, the public key will go into a secure channel. Similarly, the third step that random number which is coming back to server coming in a secure channel.

This we had done earlier, how the server, you create a secure channel to a server essentially using the same technique. And once this is received R is same as this, so you verify you know that user does actually hold the corresponding private key. Another way is that when this public key an email ID is being sent for creating signature, you generated digital signatures of on this particular message, and that signature will be done through a private key. So, if the signature are verified, so private key is they are with the user. That is ensured.

And of course, you can always put a random number so that memory attack cannot happen in this case. So, every time there is a new random number, so you generate a new, the hash value is going to change. It will not be same even if you send the same public key and same email ID again and again. Every attempt it will be different. So, replay attacks cannot happen in this case. So, this is how you verify whether the user does hold the private key or not.

(Refer Slide Time: 06:41)

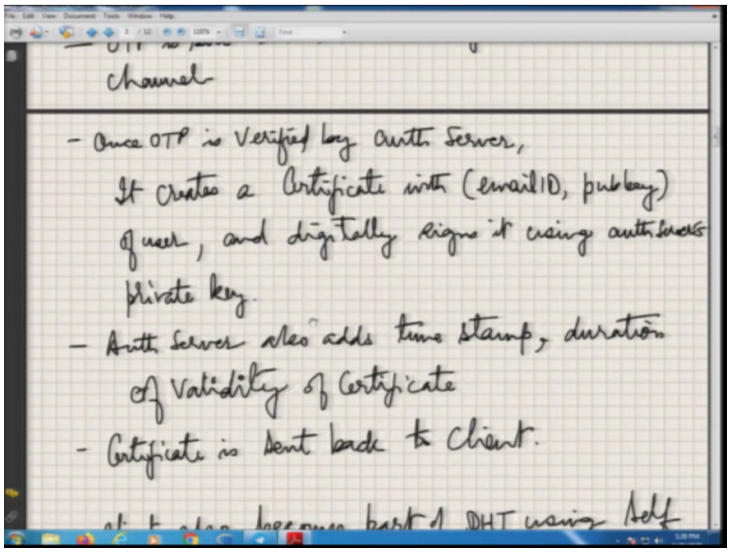


And once the ownership of the private key is ensured in this method, the Auth server should send an OTP to the email ID or it will can be an SMS sent to over a phone number. Now, the user-client, the machine, the software which is running at the user side should ask for the OTP from the user. So, you would have written on your email. You will pick up that string and will enter that thing in your client. And client on a secure channel will send it back to the server.

So server know that you actually does have an access to email ID. So, email ID is yours. So, he will then sign the certificate and he will, the certificate will contain some information which will be public key, the user ID, and other things which I will mention later. And this will be signed by the private key of the server.

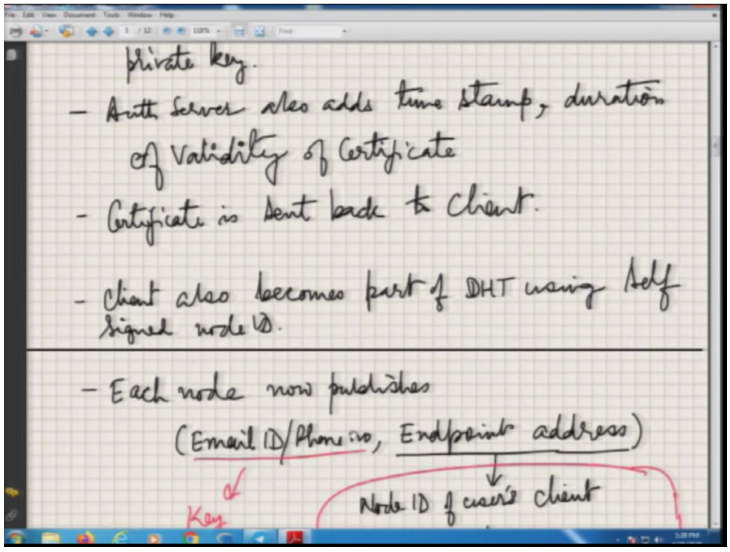
Again, this only remains with this thing. And server certificate is already there with all users. That is an assumption that actually is mostly true, that is distributed as the authentication server keys or CA keys. This is being periodically updated in browsers or this can be built into our product which we will be distributing as a client.

(Refer Slide Time: 07:52)



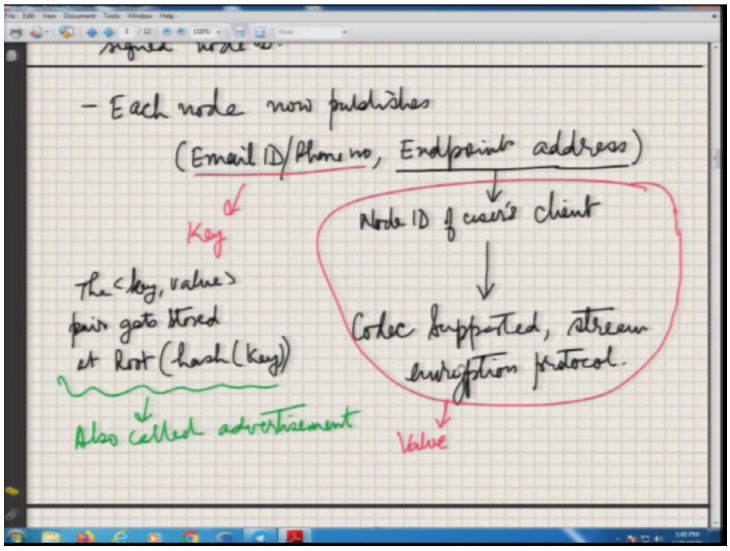
So, once the OTP gets verified, the certificate will be generated, the email ID and public key of the user until we digitally signed, and Auth server will also timestamp it. It will put at this particular moment of time I have signed it, duration of the validity of the certificate is this, so it will expire say after 6 months and this certificate will be sent back to the client. And this certificate is what will be used for mutual authentication.

(Refer Slide Time: 08:17)



And now the client once he gets a certificate will become part of the DHT, distributed hash table created by all the nodes using self-signed node ID. Remember, each device will have a separate unique node ID, but if the same user using multiple devices user IDs will remain the same. But node IDs will be different for each one of them. It has to be unique.

(Refer Slide Time: 8:40)



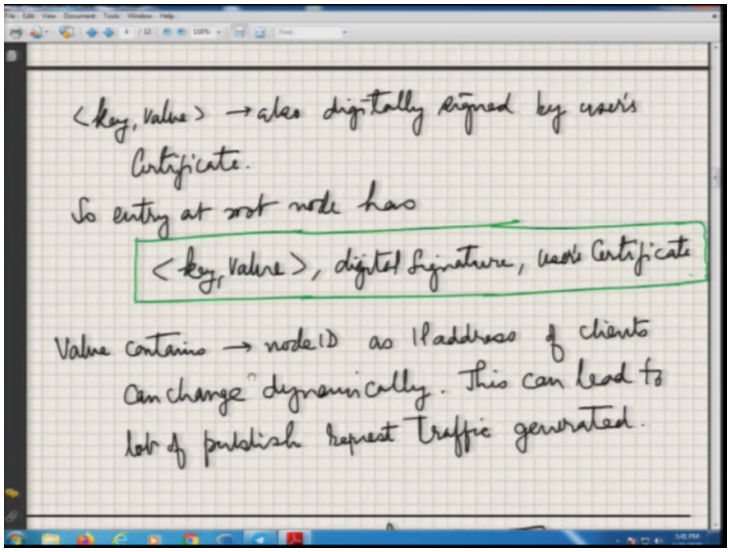
Now the node has to publish its email ID or phone number to the DHT. So that when somebody wants to talk to him, he should be able to find out where this guy is actually. So, he will actually put an email ID or phone number and endpoint address. That is what I explained in my earlier lectures. We are going to make some change here. There is a reason for doing that.

So this email ID or phone number access a key. So, I will generally compute the hash of this, find out the route node of that that will be responsible for storing these two key value pair. Value earlier endpoint address I mentioned should be IP address, port number, transport kind of thing, but it need not be. Because what happens device's IP address can keep on changing depending on the DHCP.

So, if you are roaming around so depending on to which base station you latch in a Wi-Fi network or in a GPRS network or 3G or 4G network, your IP address can keep on changing. And every time you cannot keep on publishing again and again your key value endpoint address. So best most likely the node ID will remain same. Node ID is rarely the clash will happen. So, once you acquire it, you normally keep on using it. So, email ID to node ID is what is going to be put as an endpoint address in the key value pair.

And you can further put what kind of Codec is supported, stream encryption protocol, stream cipher which is being used. And this will be stored at, as I shown here, at the route of hash of the key. And we also call this particular publication also as an advertisement. You are advertising yourself so by which other people can find out where you are.

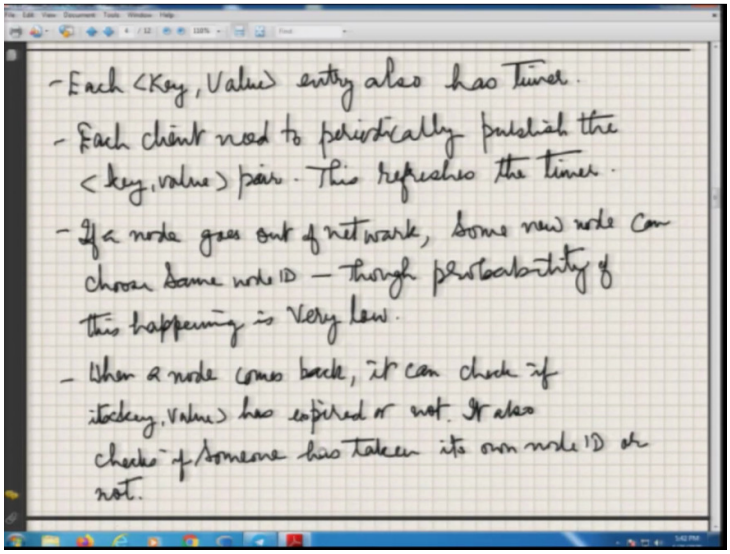
(Refer Slide Time: 10:16)



And this key value will also be digitally signed by the user's private key. So, user certificate is already there, the public key; so, you can always verify it. So, you know, that is a user's private key has used it so it is a genuine request it is not somebody else who have dumped the key value pair for a certain reason. Otherwise, anybody can do it for anybody else which should not be allowed. You should be able to advertise only for yourself, not for somebody else. This also I had mentioned in one of my earlier lectures.

So, it will be key value pair, the digital signatures, and user certificate. This will be the entry which will be there at each route node, and the value will contain node ID as IP address can change. This I have already mentioned. So, we are avoiding the traffic. We do not want a periodic publish request happening and leading to generation of a lot of traffic.

(Refer Slide Time: 11:07)



Now each key value pair which is published will also have a timer. And each client because the timer is here, it has to expire. So, it has to keep on periodically publishing the key value pair. This will refresh the timer value. So simply when I do a republish it goes there, figures out the same entry which is mentioned already exists. It is only just reset the timer. You need not rewrite into the database, just change the timer value.

And if a node goes out of the network, some new node can choose the same node ID. Though the probability will be very small for this, and normally after a timer expires, if timer has not expired, when a new node comes in, it will search for that node ID and it would not actually find out that node ID. So, it would like to take that node ID and when it will try to publish its advisement, that time the new thing entry will get published. The older one will get expired after some time and only new one will remain.

There is a small vulnerability period. So normally when you acquire a new node ID, you have to essentially now wait for some time so that the entry, older entry might have expired and then you should push in your entry actually. And you should keep on periodically refreshing it. If you are not refreshing it, then only timer will expire and entry will get purged.

So, if a node comes back again, it can check if it is key value pair already exists, it has expired or not. So, it can be either it can expire or it can be replaced by some other entry. If some other entry has come in, then certainly you have to find out a new node ID in that case. Okay. So, this normally happens for the node IDs only. Node ID to IP address. So, node ID search will always be done. I think hash key value pair, the email ID and the node ID which has been used is going to be present in the root node.

Now if the node actually goes out that node ID will no more be there. So, any new node which comes in when it will search for that node ID it would not find because that guy is out. So, it can acquire that new node ID. So now it has to because new node ID has been acquired, it will now put his own email id and that node ID will be published. The older entry will essentially now will get purged off, okay. This new node ID will be now presenting a new user ID.

So older user ID to same node ID entry will automatically will expire that also would have been digitally signed, but this will expire after some time. So, if a node goes out of the network and when it comes back in and it will try to find out its own node ID, try to use the

reuse one, so it will find that new node is node ID is already taken off. So, it will find out a new one. This is same thing which I had explained in one of my earlier lectures.

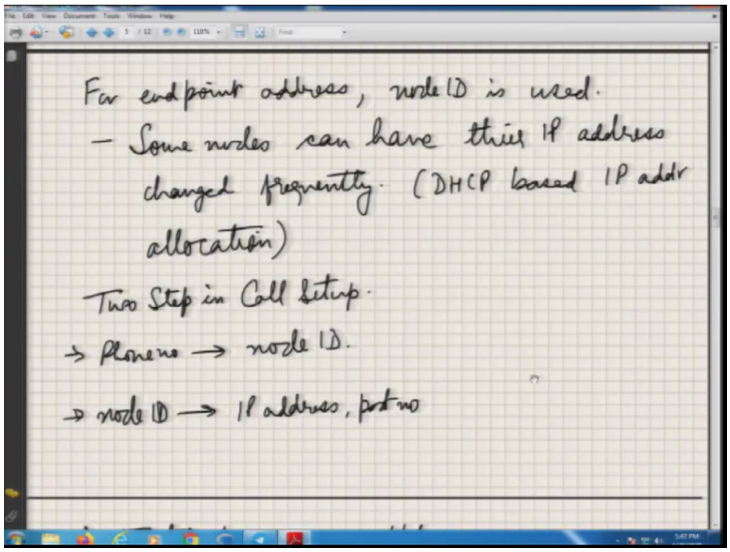
It will pick up some new random node IDs and of course, the probability of this happening is extremely low may not never be actually happening over the age of the network actually. So, it will find out a new node ID and will now use a different key value, same email ID, but a new node ID signed by the user ID certificate which would not change. And it will again, make another advertisement; the older one will get purged off.

So normally, when the key values are searched, these are done through email IDs. So, at some point of time, it is possible that older email ID and the same node ID already exist. In fact, two different email IDs to same node ID can exist at any point of time, but the probability of that happening is extremely, rarely. It will be a rare event, maybe once in few hundred years kind of thing. So, we can actually live with it.

So, but in that case, it is possible that when you make a call to a certain email ID, you end up in calling to some other node ID, which actually is not the original user. But that probability is very small, but that does exist. So essentially, it has to you have to keep on replenishing yourself. That is why this expiry mechanism has been actually put.

So, if you want to be safe that nobody is going to call me thinking somebody else, then you should wait before you publish you start taking up the calls. You publish it immediately your email ID to node ID mapping, but wait for some time before you take up the call. Or you check for if old entries does exists for the same node ID or not. But you cannot actually remove that entry because that is assigned by a different user. You have to wait for the timer in that case. So, this timer can be say 15 minutes, so 15 minutes waiting is okay when you come in. And, since, it is a rare event, so we can even bypass that rare event which happens once in blue moon.

(Refer Slide Time: 15:47)

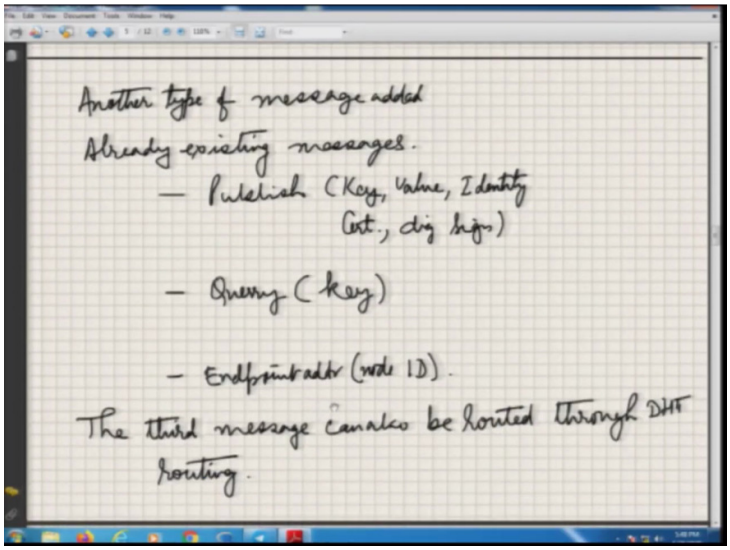


Now coming to the endpoint address how that will be done? We have made a modification that email ID and then no; there is no email ID and value is node ID. That is what we had done instead of endpoint address. But then I cannot set up a call unless I have an endpoint address. So, what we will do is for each endpoint address, in this case node ID is used I will use this node ID to find out what is the actual IP address, port number and other things.

The reason we did is because of the DHCP protocol being used for mostly the IP address allocation. These are not static IPs. So, because IPs are dynamically allocated same user same device can actually get different IP at different points of time. So normally, there will be 2 steps in any phone call setup.

So, first of all, you will put a phone number or an email ID actually you can put it. I will find out what is the node ID correspondingly. Once I have node ID, I will find out what is the corresponding IP address and port number on which the call has to be made.

(Refer Slide Time: 16:44)

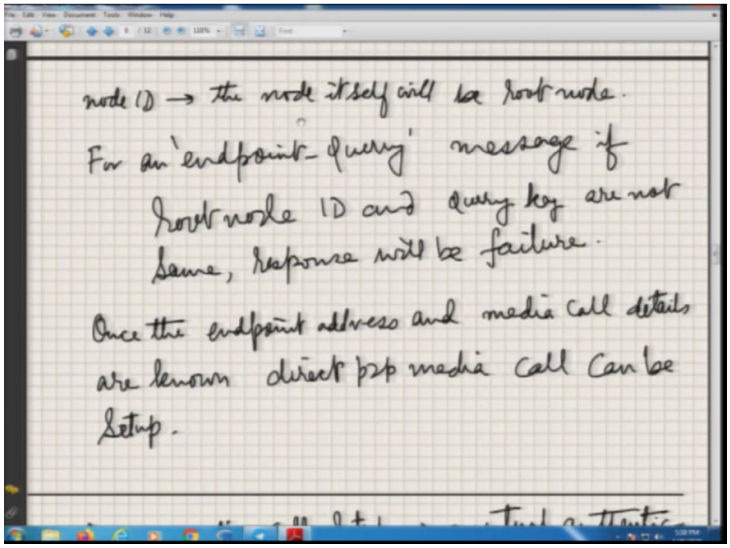


So, for this we need to change the messages which can be put in the DHT. So far there are already existing message are publish and query. So, in publish we will put key, the value, the identity certificate, the digital signatures. So, you we will find the hash of key, find out the root node and there its key and the remaining 4 entries has been signed as in, published as a value. You can also have a query. So, when you do a query, you will get the value identity certificate and digital signatures back in the response.

Now we do 3rd thing which is endpoint address. Actually, it is not published in DHT. What it happens is when you give endpoint address query basically and you give an argument node ID, the DHT routing is still used till that node, this endpoint address request goes to the node ID itself. So, if a node ID exists, it should be route of itself.

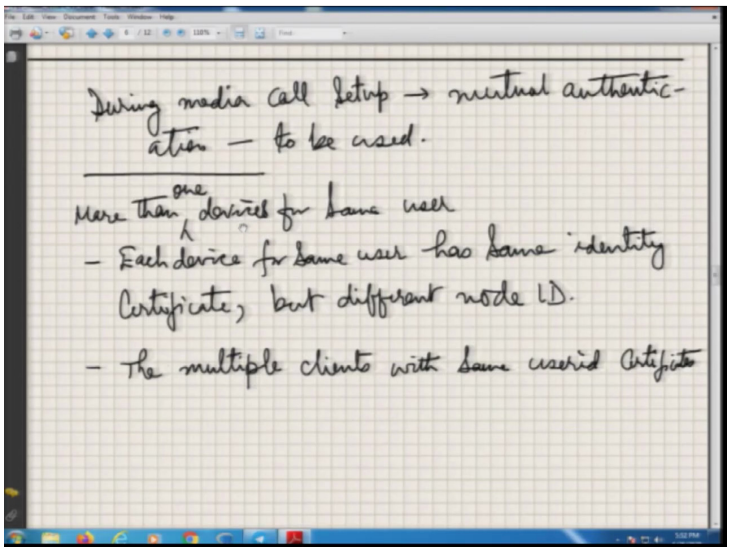
If it is route and node ID, route of this node ID is not the node ID itself that node ID does not exist. So, in that case, the response will not be sent back it will timeout or it will be a failure which will be coming okay.

(Refer Slide Time: 17:51)



So, the node itself should be the route node for a node ID. And whenever the endpoint query is being received by me and I am the route node for myself its node ID is mine, I should now there is no key actually here. I have to simply say that I am going to send my IP address, port number on which the person can communicate back to me. If my node ID and I am the route of my node ID in this, which is coming in this message and I am the route of it but my node ID does not match in that case I have to response with a failure. I will respond with a failure actually.

(Refer Slide Time: 18:26)



And once the endpoint address and media details are available then the source, the Caller will set up a media call with the Callee now and then they will actually talk to each other. So, they will put up their own encryption, direct peer to peer connection will be made. This is how typically actually it will be done.

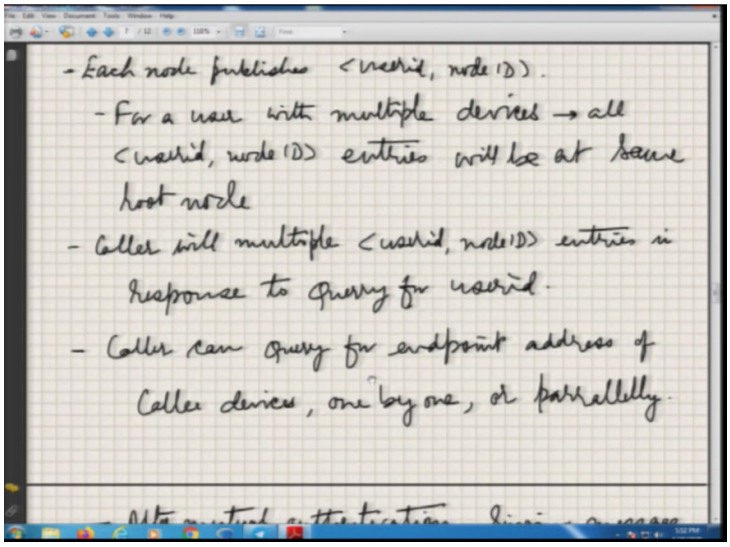
And during media call, you will also do a mutual authentication. So, I want to call some user ABC@iitk.ac.in. So, I will do a mutual authentication by verifying certificate of each other. Now, that is where even if by mistake, somebody else who is a new guy, old guys is dead, new guy picks up the same node ID, and there is an entry clash. Even if I go to the wrong

person, during mutual authentication, a failure will happen and then I will again attempt and by that time entry might have expired and then I will go to the correct entry.

So even if I get for same user ID, for a user ID, I get a node ID but that note ID of somebody else he will not be able to do a mutual authentication. Then I can simply disregard it and wait for some more entry to come, some new entry to come in or I can assume the node actually does not exist. So, mutual authentication is mandatory in this case, so that the wrong connections does not happen.

Now, there is a there is a possibility that more than one devices can be operating for the same user. How to handle that? So, I need to even provide a provision. So, point to point calls I have done so far. So, each device for same user will also have the same user identity, same identity certificate will present but it will have a different node ID. So multiple clients with same user ID certificates will exist but with different node IDs.

(Refer Slide Time: 20:14)



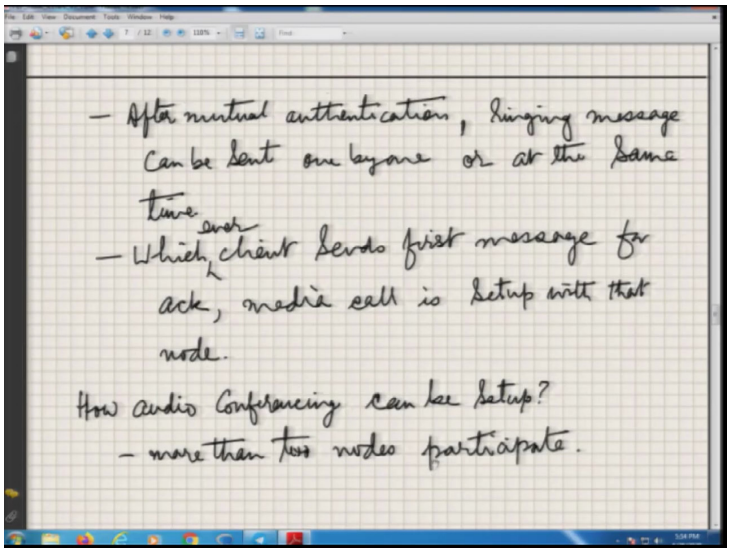
Now each node will publish user ID to node IDs. So, there will be for same user ID, there will be multiple node IDs available in the key value pair. When such thing happens so all user ID to node ID mappings will be sent back to the caller node. They will all be at the same route node. So, route node will send all the entries not one entry but all the entries back to the caller node. And caller with this will actually receive these multiple entries, user ID node ID entries in response to the query. And it can do now handle the things in multiple ways.

So, it can actually query for endpoint address of all the callee devices. So, if I get 10 (ten) devices, I will put an endpoint query for all 10 node IDs and I will get all IP address, port numbers. I can do it one by one. I can do in parallel; I can shot off the endpoint query. So, both ways it can be done.

So, if one by one it is done, I will get the 1st query entry. I will then try to set up a call. So, basically it is equivalent into ringing, after mutual authentication is done. If the call is not picked up, I will terminate, I will find out the next guy. Again, I will now do the endpoint query; again, I will say try to setup a call after mutual authentication. You will keep on doing it.

So, wherever you will pick up the phone so, it is like you ring on the first handset, nobody picks up goes to the next one, nobody picks up goes to the next one, it is like a call forwarding. And, the order can be specified by the user itself when it is publishing the key value pair. And wherever the pickup happens, the call essentially start the media call will start with that particular node.

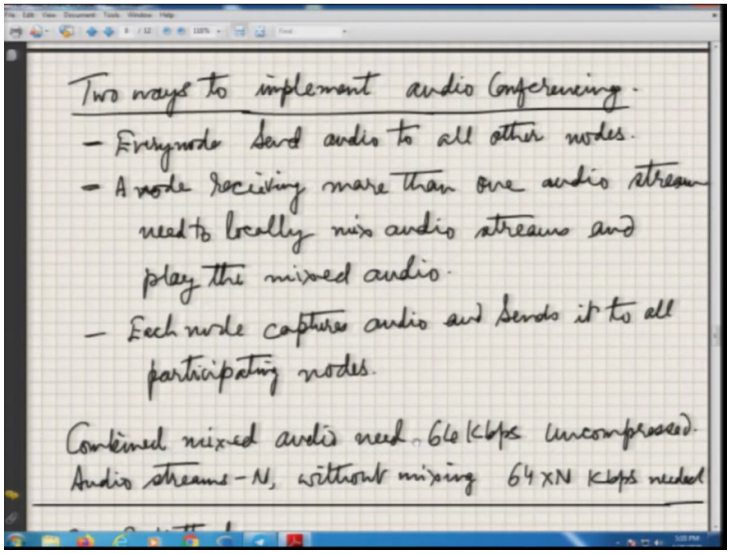
(Refer Slide Time: 21:56)



This is of course, very simple. You can do parallel ringing with this. You can do one by one ringing with this. So, you have 10 phones at 10 different places. When I want to make you a call, all 10 will ring start simultaneously. You will pick up from anywhere; ring for others will stop and I can talk to you. So this is a kind of implementation which can be done in this case also.

Now another bigger challenge is audio conferencing so where multiple people would like to talk to each other. So, it is pretty common nowadays. So most of the telecom operators do allow on your mobile phone using multi-party audio calls. So how to implement that? A similar concept can actually be used to even for video calls here.

(Refer Slide Time: 22:43)

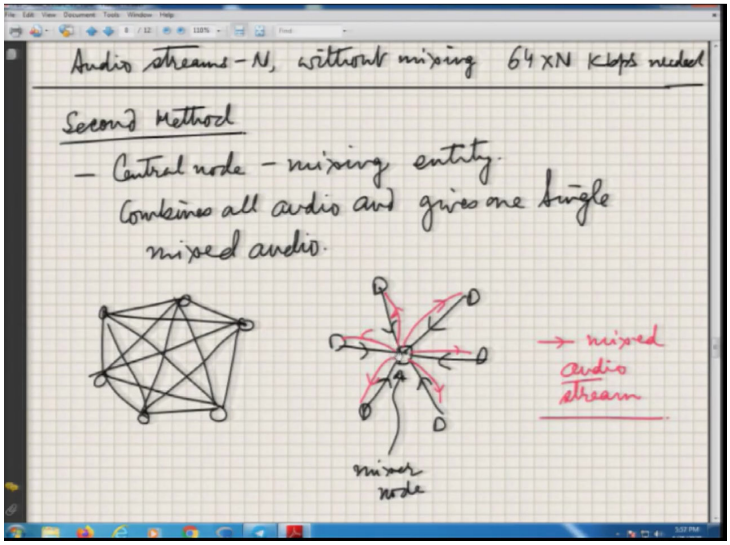


So, let us see how this there are two ways you can actually do it. So, everyone will send his audio to every other node. So, if there are say 5 guys who are participating in the audio-conferencing thing. How to set it up is 1 different issue but once you set it up, everybody sends the audio, a copy of it to all the other 4 (four) guys. So, everybody does it.

So, I have received 4 (four) audios from others and I will combine them in my device and will then play it back on a speaker. So that is one way of handling it. Basically it is creating a full mesh connectivity. And each node will also now capture the audio and will send it to everybody else, this is a thing. But the problem is the audio need to be mixed. So, everybody needs to do a mixing thing. It requires more bandwidth.

If there are say 100 guys coming in so 100 bandwidth streams will be coming and each band audio stream will be of 64 kbps by default. That is a standard with 8-bits per sample and 125 microseconds for between every 2 samples, consecutive samples. It will turn out to be 64 kbps. So, you require 64 X N kbps for N user audio-conferencing system. This is not very efficient. It is better that we do different thing.

(Refer Slide Time: 24:00)

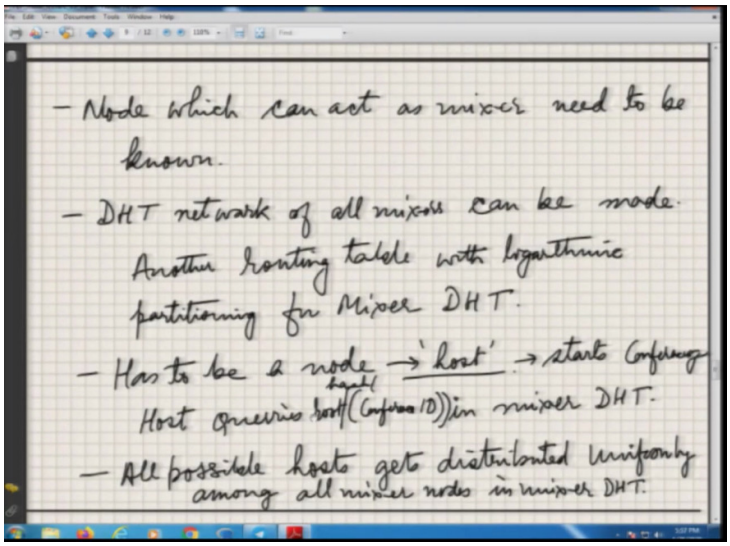


So, this is the first one where I am telling that everybody is communicating to everybody else. You can use a 2nd method. I can define a mixer unit. Now, remember the mixer is not an ordinary node; it is a special node. So, everybody sends the audio to the mixer. So, this guy combines the audio and sends the mixed audio back to the all the users. So red one is the mixed audio.

And normally it will do; what it will do is whatever audio is coming from this guy will be subtracted, a part of it. Only 0.1 or 10 percent of it is going to be fed back. This for a feedback purpose but the audio from all other nodes will be sent without any attenuation to this. So, this has to be done for every user.

So, there is a computational overhead at the central mixer node. But this is efficient. See every node is only using still 64 kbps in both directions. It is not 64 into N. 64 into (N-1) actually for that matter for N-way video-conferencing, audio-conferencing.

(Refer Slide Time: 24:57)



Now nodes which can act as a mixer need to be known to everybody. So, how do I know that for this audio call I have to send it to this mixer? So, what we can do is we can actually some

nodes were willing to become mixers. I can create a DHT network of them and they will maintain now, every node will maintain two routing table if he is mixer as well as ordinary node.

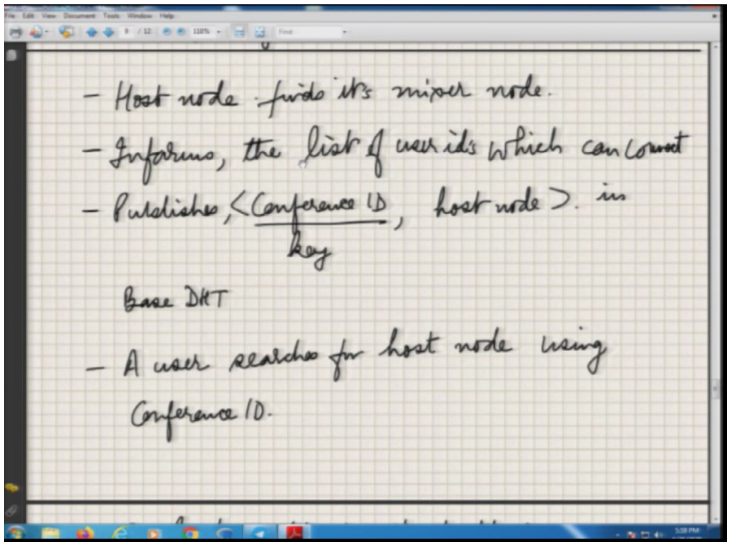
If it ordinary node, he will only maintain, he will get the routing table of the DHT, the mixers but he himself will not be part of that thing. So, his entry will never go into DHT mixers. But he will be essentially getting from some mixer, the entries for a mixer DHT. There will be 2 two DHTs which will be operating.

And mixer DHT will also have a logarithmic partition so that the everybody is always part of the network. So, a node has to become a host first before audio-conferencing can be set up. So, host will always set up. It will start the conference and host will query the route of the hash value of the conference ID, a conference ID has to be generated. It has to have for example, the email ID of the host user and then some sequence number date so it will be kind of a unique conference ID. Compute the hash of this and compute the route in the mixer DHT.

So, the node which will be responsible which will be route node in mixer DHT network will become the mixer for this conference ID. So, everybody who is participating in this conference has to communicate to it. So, all the conferences which will be created will get distributed across all the mixers which are forming a DHT. So, not single guy is going to be loaded up. Even if a mixer dies off, again a load re-distribution will happen and somebody else will can act as a mixer for your audio call.

So basically, all hosts will get uniformly, technically all audio-conferencing instances will get distributed across mixers uniformly. So, load will get distributed and of course it is a fail-safe. So, if some something fails, it is the resilient system. Somebody else will take over.

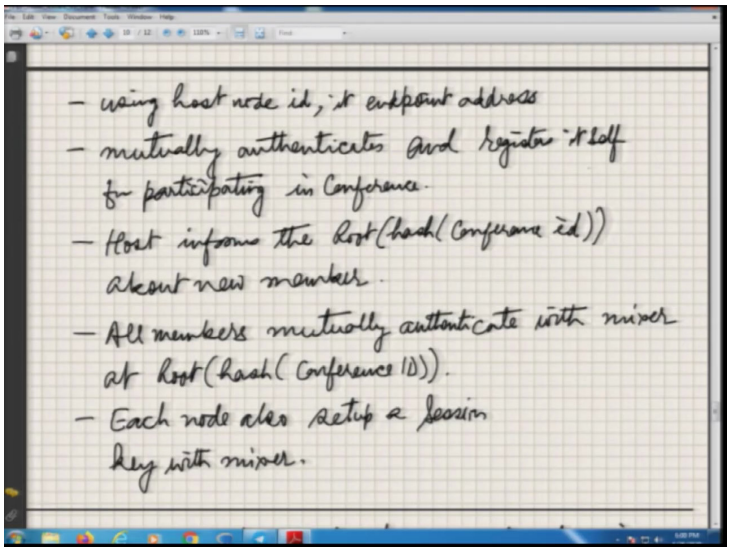
(Refer Slide Time: 27:00)



So, hosts will find out its mixer node by doing the route of hash of conference ID. It will inform the list of user IDs which want to connect, which can connect to this mixer node. So, I should know that who are going to participate with me in the audio call and I will send a list of all those user IDs they can communicate from any device. So, there will be mutual authentication which will be happening and of course a secure channel also will be created.

And it will publish this conference ID, which will be acting as a key and the host node information in the base DHT. So, people will search with this conference ID, get the host node. They will come and talk to the host node.

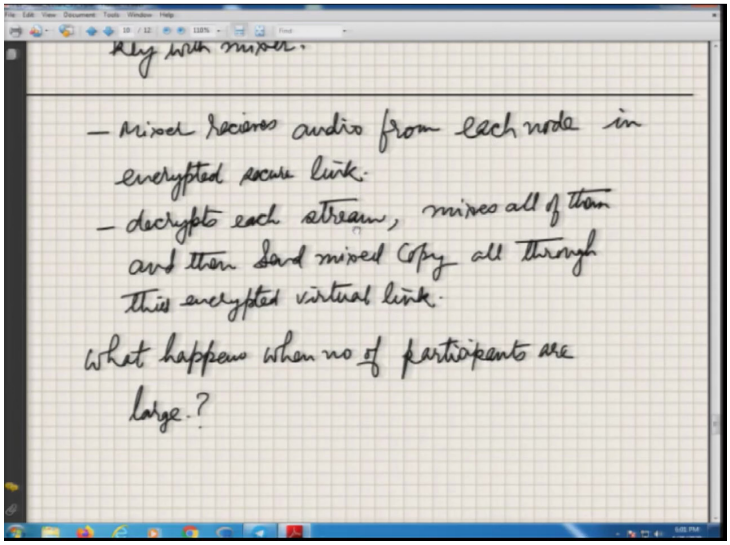
(Refer Slide Time: 27:44)



And, and once they know the host node ID, they will find out the endpoint address, they will do mutual authentication, they will register themselves for the participating in the conference. The host will inform this to this particular route node in the DHT of the mixers about this new member. So, all members now will do, will be actually informed who is a mixer node where they can; they actually can find out mixer node using route hash conference ID also.

But a mixer will not accept them unless is being informed about that who is permitted by the host. So, all members will do the mutual authentication with mixer. It will look into the valid users who can be pushed into the network. And each node will because they are doing mutual authentication, they will also set up a session key. So, from each user to the mixer, the audio will be transported in a secure channel. But it is being taken out. So, mixer unit actually can listen to your communication. It is not end to end encryption as of now.

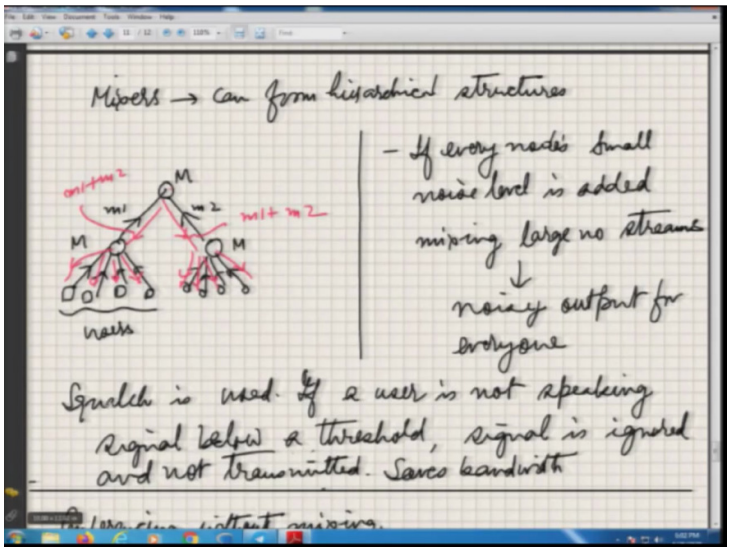
(Refer Slide Time: 28:48)



So, mixer will receive the audio from each node in encrypted secure link, decrypt each stream, mixes all of them and then send the mixed copy of all these through the encrypted virtual link to all the nodes back. Now what is going to happen when the number of participants are extremely large?

A mixer node may not be able to handle. Maybe 5-6 people is fine but if 1000 people want to listen. Of course, one important thing you should know note down that all 1000 people will not be talking otherwise, it will become noise. Normally, it is 1 or 2 people or 3 people will be talking at any point of time; otherwise it is a noise, people will simply turn off their speakers. Or we should simply shut the communication down at that point of time unless only one or two people start talking again. It does not make sense to transmit noise.

(Refer Slide Time: 29:37)



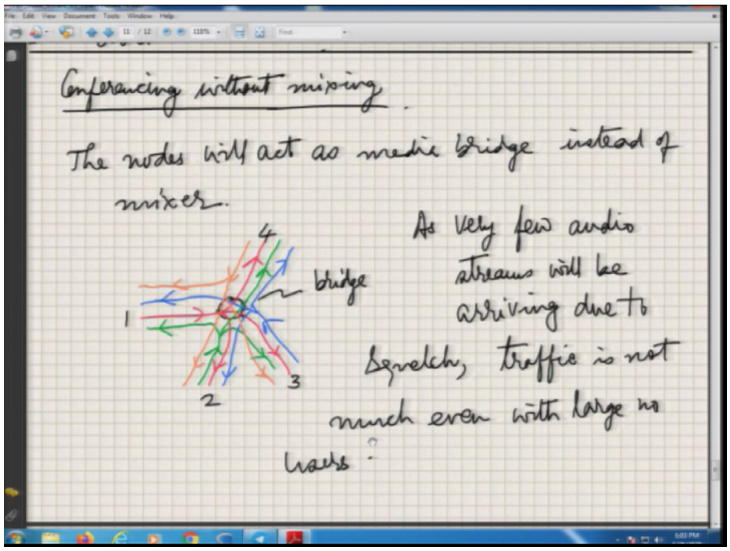
So, we can even actually go for a hierarchical mixer. So, there is M1, M2 are the links to another mixer. These are 3 mixers which are there. So, some groups are connected here; some groups are connected here. So, all audios are being coming here which are being mixed here

and sent to the master mixer. Same will happen from here. It will do the mixing of these two mixed audios which are coming and this is being further redistributed back. This is done kind of forming a tree.

Now, the problem here is that every node, even if you are silent some noise will be get added and if you start mixing like this, the noise level will become very high. So, noisy output will be there everywhere. So, this actually is not used in real life. We use something called a squelch.

So, if there is a sound level is below a threshold, you simply shut off everything. You do not transmit anything actually. So, only guys who are actually talking, their audio will be going. So, you would not be seeing the noise actually. So, squelch is one of the important things which is implemented.

(Refer Slide Time: 30:38)

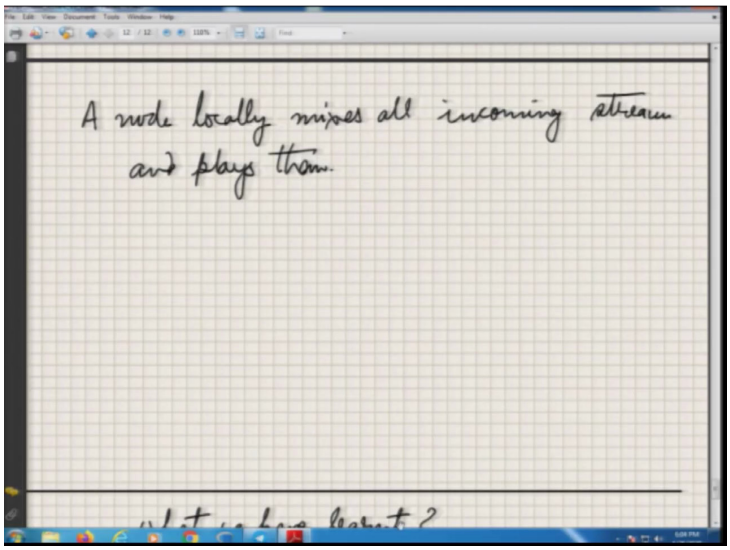


And now you want to do it without mixing. Certainly, it is possible. We use it through something called media bridges. So, in fact, a lot of popular video conferencing systems nowadays, actually use this technique. In this case, the bridge is communicated to you by either peer to peer DHT system or through a centralized server. So, most of the implementations are using centralized server currently to tell you to which brings you have to connect and you can create a tree of bridges by where the mixing can happen.

So, the bridges are very simple. There is no mixing done at the bridges actually. So, your audio comes in and audio is simply is being routed to all the other nodes. Now, important thing is that since not everybody is talking when the audio comes into the bridge, it may decide not to transmit to the nodes if there is nothing coming in.

So, you have a communication outgoing link to a bridge but bridge will not be forwarding your voice because you are not talking. So, at any point of time what you will be receiving only 3-4 audio streams which you can mix locally. So, bridging is far simpler to handle and even with 1000 of people, only 3-4 streams will be received by everybody else. So, it is basically kind of it is gating function. There is no mixing of audio being done. So, squelching is essentially handled by video bridge. So, this also improves the quality. Okay, this is a kind of a very elegant system being used very popularly.

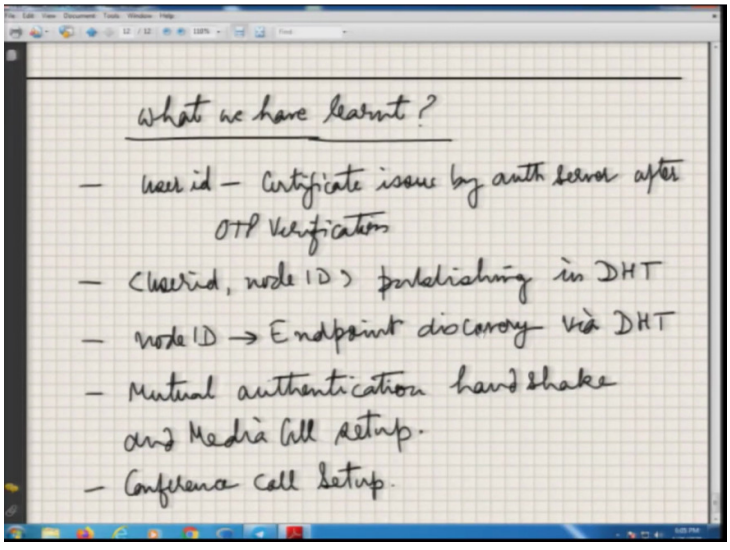
(Refer Slide Time: 31:58)



And nodes will locally mix all them incoming stream and will play them. So that is how we actually do the Voice over IP over P2P. The same thing can be extended to audio thing, but audio mixing is slightly complicated because screen size remains the same but multiple audios are reduced in size and they are fitted into a smaller frame.

So video quality goes down but the because the frame size remains the same, your bandwidth remains requirement remains the same, but more videos can be pushed in by reducing the size. So, it will require more signal processing.

(Refer Slide Time: 32:33)



So, what we have learned in today's lecture is that a user ID will get a certificate issued by Auth server after the OTP verification. User ID will now be published along with the node ID

not the endpoint address in the DHT and node ID to endpoint address discovery will be done through a DHT but there is no key value pairs to be stored by the route node in this case.

So, basically you just search for IP address of a node ID. It will goes all the way to that node and he will inform what is his IP address, port number on which you can communicate. This takes care of dynamically configurable dynamic reconfiguration of IP addresses which happened in the nodes very frequently when we are using DHCP for IP location.

And of course, we do mutual authentication-based handshakes. So, that ensures that you do not get connected to the wrong person. And then only you do a media call-setup. And we also have discussed in the lecture conference call-setup. So, we will actually take off from here further in the next lecture.