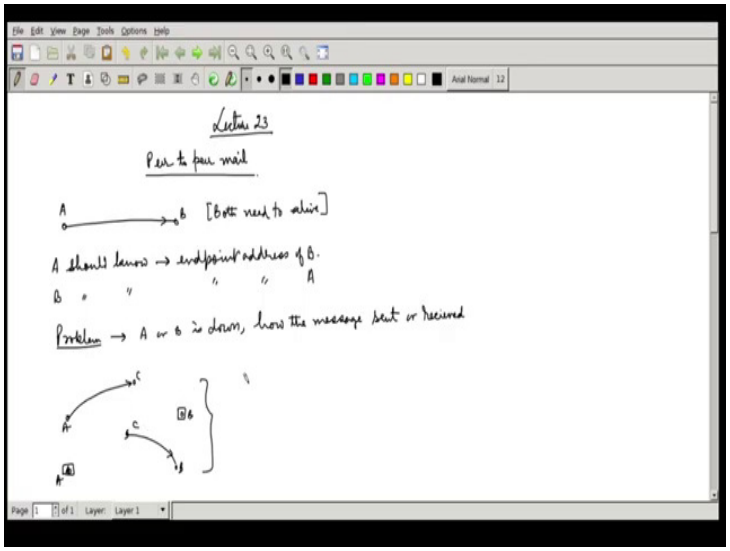
Peer To Peer Networks
Professor Y. N. Singh
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Lecture 23
A Design of P2P Email System

(Refer Slide Time: 00:25)



Welcome to the lecture number 23 in the Peer to Peer MOOC series. In this lecture, we will be focusing on peer to peer mailing system. So, we all know that how the email actually works, normally you are going to sit on your computer you will log on to your terminal. Either you will be connecting either through a web browser to a mail client or you will be running a client on your machine say for example, Thunderbird.

Same thing, you do on your mobile phone. There is a mail client who is installed, for Android phones popularly it is a Gmail client, who is used; it is a program which is running on your machine itself and this talk to a server. So, but idea is that A can write any message and this message can be then transported to B.

And we have a unique identifier being assigned to A and B, so you can actually send a message to anybody. So, this will just propagate through the network and it reach to everywhere. So, if you want to do this, in a peer to peer system the important thing is that A should know what is endpoint address of B. Similarly, B should also know that what is end point address of A? In fact, it is easier if A finds endpoint address of B and when it is going to communicate to B, B will anyway will figure out what is endpoint address of A, because A has communicated. They need to also do a mutual authentication so that B recognize that this

is A, and A recognize who is B, and they can always transfer the message from one to another so, that can always be done.

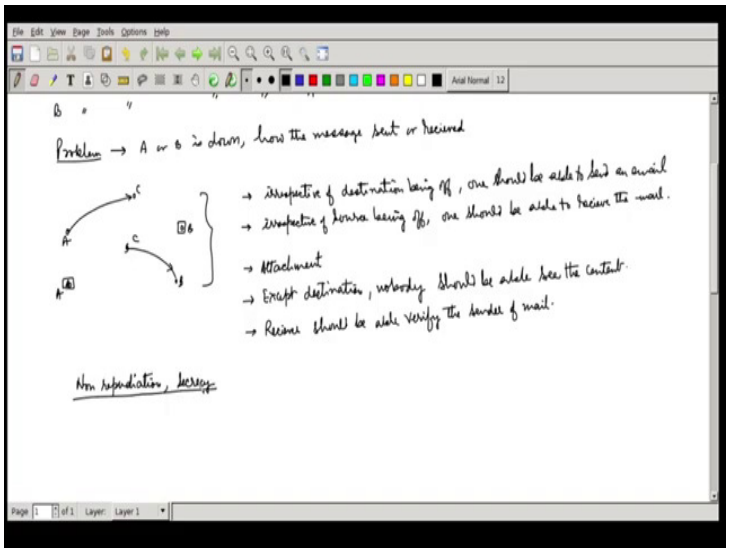
And because we are actually using certificates for A and B, so A can actually digitally sign the message and send it to B, so B knows that it has been sent by A, they anyway have done a mutual authentication so whatever messages have been transmitted, B knows who is sending the message and A knows who is receiving the message, meaning that they will digitally sign in this case, but then they both need to be alive in this case, the important thing that both needs to alive.

Both are there in the network at the same time, they can talk to each other and then they can pass on the message. But normally when I am sending message, I just want my system I want to just hand over the message to somebody even if the other endpoint is off. And whenever that person comes alive, he will get my message and he knows what I communicated. That is ideally the way the mailing system actually works. Now, the question which remains with us is, the problem which we have to face if you want to build up a peer to peer system is, if A or B is down, all the messages will be sent or received.

So if B is down, A should be able to send the message, and when A down, B still should be able to receive the message, it was dispatched. So there may be a time offset so A and B need not be on at the same time. So, maybe what we can do is that when B is off, A can send it to some other peer. So we will call it C, and B is off as of now so I can call it off.

And next time when A is off the message can be retrieved from C by B when B comes alive. This could be a potential way of solving the problem. We can actually use some intermediary which can act as message storage. Even when A and B are off, we should be able to send that is a property that is a desirable thing whenever a mail system could be implemented.

(Refer Slide Time: 04:54)



First thing irrespective of destination being off, one should be able to send the mail. Of course, similarly irrespective of source being off, one should be able to receive the mail. Of course, this was first criteria, secondly, I would like to always send attachments, we actually send attachments through emails, and we can send documents, I should be able to send the attachments, that property also should be there and then I create a peer to peer mailing system.

In this case, when both the A and B are alive, A can send the message, it can transact even all the files, whatever attachments along with the mail to B, and B can interpret that. But the problem happens in this kind of scenario. Here, also we need to provide a mechanism. But whenever I use some intermediary in the mailing system, normally it says we depend on the mail servers, and mail servers we assume are going to be operated by operators.

Though actually they can see the mail what is there inside the content of it. But we normally assume there is no third party except operators who are actually can look at the mail. There is some kind of rudimentary security which is there. But here these intermediaries can be any user's node, any user's machine, so I need security.

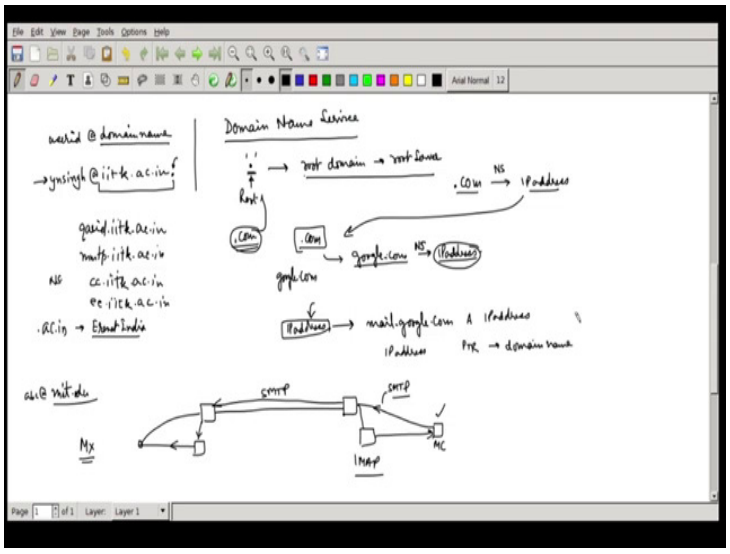
To accept the destination nobody should be able to see what is there, what is the content? We need to provide a mechanism for this. And of course, whenever this kind of thing is there, when I am retrieving the message, I should be sure that who the sender was, the genuineness of the sender, it should not be a fake mail.

The receiver should be able to verify the sender of email, so that is important and basically they are characteristics which are required in any mailing system. We in fact, requiring even the current existing mailing system do not satisfy this criterion unless you encrypt the mails or you actually you are using some kind of a public certificates being stored in your mail clients.

If you install a certificate then it is possible that you can digitally sign the mails. You can receive the mails encrypted. But you cannot send the mail to somebody else unless you have the other person's certificate with you. So this is already there available in most of the mail clients, but now you mostly depend on the operator for this particular purpose.

We call this particular property what is non-repudiation, if somebody sends me an email, that other guy should not be able to say that I have never sent this email, this is a fake email, you should be able to prove it was indeed sent by him, that is a non-repudiation property and the secrecy means nobody should be able to see this, these two properties are essential.

(Refer Slide Time: 9:18)



The current system this can be done to an extent by digital signatures and by using public certificates, the certificate, the public key which is there inside the certificate using that you can do the encryption when you are sending a mail to a receiver. Let us look at how the existing system actually operates. And from there, we will figure out how we have to actually handle things. So currently, emails are normally being always of this form, you will always have user ID at the rate some domain name. A domain name for examples for my main idea

domain name is going to be iitk.ac.in, whatever is my mail id. This is my mail id which is going to be there. This is the way we actually figure out.

Now, important thing, when somebody wants to send a mail to me to which machines the mail should be sent, how that is identified? Now, important thing, in this particular domain, this actually comes from DNS. DNS stands for domain name service; the way it happens is throughout the world. In fact, there is a dot which should come here. If you look at this dot, this is also a domain. And we call it a route so throughout the world there are a few servers which are the route servers.

So, whenever you want to find out what is an IP address for this? This will always go to that route server. Then information about this means actually you have to go to that route server. Route domain, there is a corresponding route server, and these mostly come in hard coded inside your machine. Whenever you do the operating system update on Windows or Linux, these things entries get updated there, they are fixed entries, they run across globally.

Now, what happens is somebody, now we actually create dot com domain, so dot com domain is actually registered in all these root domains, so all these servers will be maintaining entry for dot com. So dot com domain is the responsibility of another server, we call it domain server.

So these are domain server for the route domain, so for dot com server there is going to be a domain server. So these route domains will be maintaining entry for dot com, and they will say for this you have to go to this particular IP address, so there is an IP address for this domain server is going to be retained. So what this particular IP address is for this particular machine, which is acting as a domain server for dot com.

Now, you want to know, start, say google dot com. So what Google does, Google actually has gone to the dot com registrar through this server, which is being owned by a registrar and said, you can create an entry for me. So this guy has just created an entry for Google dot com and so this is going to be the IP address for that.

So basically, what it is saying the name server for this is going to be this IP address, there is a name server entry, which is going to be created here. This is that name server entry, which has been created. So there are many entries which are possible, so I am not going to details of DNS, so this name server entry was Google gets created. So Google actually has this particular machine with this IP address.

So whenever google actually going to change this machine. For example, Google changes his IP address, it has to go to again, this registrar of dot com and make a change here, but anything which is inside the Google. For example, you have some mail.google.com, for this Google need not go to this registrar, it is already having a registrar for Google dot com. Here it will not maintain an entry, which will say mail.google.com and then it can actually give an address IP address for this.

So we call this as A entry actually. And similarly, I can also create this A entry type A, there is also an entry for PTR pointer. So, here IP address is being put and you can actually have what is the corresponding domain name. So, this is used for reverse domain entries for mapping those.

This is how the way the domains actually work. Now, remember, iitk.ac.in is a domain. So, the server for this actually is running in IIT Company. So, we are running a name server, but we are actually this entry will be known to you so what we have done is, IIT computer has gone to a register for dot ac dot in domain, so which in our case is going to be Ernet India.

So this organization maintains a server which is the domain for dot ac, so anybody who wants to get a sub domain in dot ac dot in he has to go here and they will make an entry. They have actually created entry for a name server which is running with IIT Kanpur. IIT Kanpur can actually have various kinds of machines so we have machines like qasid.iitk.ac.in, we have SMTP so we have actually created these entries in our name server in IITK server. You have not gone to the Ernet India, only for iitk.ac.in we got an entry done, any further entries I can do it.

If within this similarly we actually created the subdomain entries for the cc.itk.ac.in which is a computer centre, even entry for ee.iitk.ac.in and we do run a DNS server in electrical engineering also. I can actually further create machine names here, so for this domain, so whenever you want to send a mail to me, you will for example, you are in some other domain you are in say mit.edu domain and you want to send a mail to me so you are abc@mit.edu.

Whenever your mail client sees this, he has to figure out that where is this iitk.ac.in exists, so he will go to a route server because he does not have an entry for, I think it only knows about what are the this DNS for mit.edu only knows about what are the hosts inside MIT.

And it might have cached some entries which it might have queried by assuming the earlier names. So domain has come to iitk.ac.in so it may not be as in the cached entry also. So it

will go to the route server, we will say where is the dot in DNS server, so it will then go to the dot in DNS server, it will ask you where is ac.in server.

So, Ernet India will reply him, will actually the DNS or Ernet India will reach to him, then it will ask from Ernet India DNS, there iitk.ac.in. Our DNS entry will become available to them and then they can actually ask because name has to come here, there is something called MX record, mail exchange record. Last what is the MX entry for this particular domain?

IIT Kanpur domain will now give an IP address of the mail SMTP server which we are running on which we received the mails. Once that entry is available here, this mail client will then or the SMTP server of this particular MIT will then talk to our SMTP server which is identified by this, and will send the pass the message on to him.

But now can somebody else also can spoof can take up that IP address? Somebody can for example, change in iitk DNS, so unless iitk DNS need to maintain both entries. It will maintain the entry from IP address to the domain name and then domain name to the IP address both. So both should be verifiable, for this we call it a reverse zone actually entry mapping. The reverse zone verification you can say it is that done. So then mail can be handed over to you.

Technically, your mail client will talk to your SMTP server. Your SMTP server will then talk to the SMTP server of the destination. And from there, it will be giving to the IMAP server there is another server and my mail client will synchronize with this for receiving the mails, this is my mail client. For sending the mail I will be sending it to my SMTP server here, which in turn will find out whichever is the destination.

It has to give a reply back to you it will give a reply here, which in turn will become available to your IMAP server and you can fetch the things from here. IMAP is Internet Message Access Protocol. This one is SMTP, this is a Simple Mail Transfer Protocol, so SMTP is also used here. But they do verifications of each other by using reverse domain entries that is typically the way that things actually work.

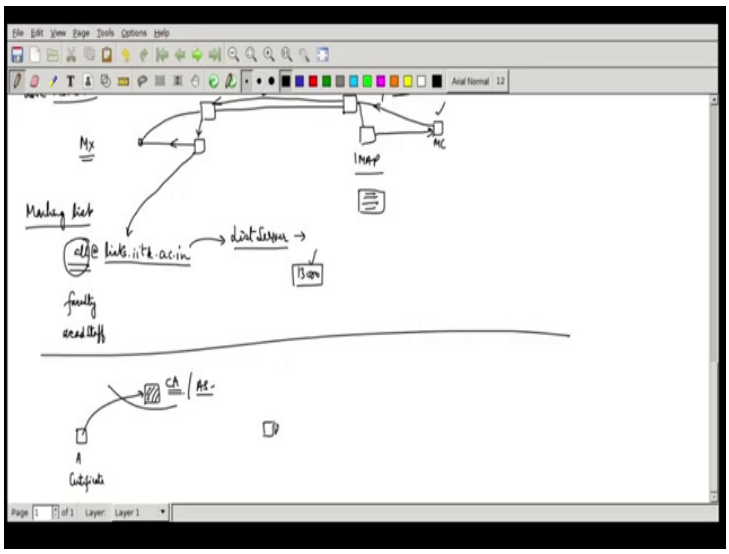
So DNS plays a very important role, it tells which is the entry to whom I need to hand over the mail. In fact, when these mails hand over is done along with this all attachments, everything is actually passed. So normally the attachment limits are there that depends on your own SMTP server. For IIT Kanpur, for example, I can send the mail up till 20 MB, Google provides a different limit. Different mail servers can have different limits. Once the

mail is received me, it will come to my IMAP thing and from there my mail client will actually fetch everything and I can read what is there in the mail.

Now there is no way I can verify it actually has come from here, one way of doing it is if this guy knows my certificate, it will actually encrypt the whole name and then encrypt the key with my public key, and that will become part of the header and this mail will come, so nobody can see in between.

When it reaches to me, I have the private key so I will decrypt the whole mail and I can see. Then Thunderbird actually, we can do this and we can use it, we in fact can even create our own public-private key pair and use that. So non repudiation actually is not built by design into the system, this has to be overlaid on top of it. This is something which we do.

(Refer Slide Time: 20:44)



Now, there is one more thing which we do in mailing systems we call mailing lists. I will be talking about both the implementations in peer to peer fashion. What are the mailing lists? For example, in IIT Kanpur when a director want to send mail to all the people, all the people on the campus, which includes the students, faculty, everybody, so even, he cannot actually remember all the email ids of everybody 13000 email ids, he cannot remember.

So what he does, he sends a mail to all at the rate. Now, I am writing a separate DNS, this actually does exist in iitk, all@iitk.ac.in. So this is a machine which is running separately, and this is going to have a, what we call list server. So this is maintained by computer centre, and they put everybody all 13,000 email IDs are listed in a database here.

They will not only maintain all lists, they also maintain many other lists, they maintain a faculty, they maintain a aced staff, is academic staff short form for that, this almost similarly is students, PG students, all kind of, even faculty has got created their smaller mailing list in this. But the important thing is that when a mail, this is actually acting like a client, this actually gets the mail from director and it is being fetched by a dial up server.

And once the mail comes here it starts processing the mail, it figures out what is this destination and then it will look into, this is basically a separate domain. Technically, this SMTP server passes the mail to this all@ iitk.ac.in and then it will start processing depending on what is there in this user ID part.

And then if it is all, it will then create a copy of mail for every user all 13,000 copies which are made of, this is a cumbersome process. Every time you fire a mail, it takes some time for it to reach everybody. If you want the country wide mailing list, this actually is an exhaustive task to be done. But we do use it and these 30,000 mails or then being sent back by this machine to this place and SMTP server to everybody so there is a one copy goes in everybody's mailbox.

So IMAP actually create a separate mailbox for every user, so every user there is a file, so there are 13,000 accounts or 13,000 copies are going to be done here. There is of course engineer's way of doing it, if there is a universal content is not been digitally signed or encrypted. In that case, we can create only one copy and people can refer to that actually, that also can be done. But this is what the listserv. We also need to create this listserv also in a peer to peer mailing system.

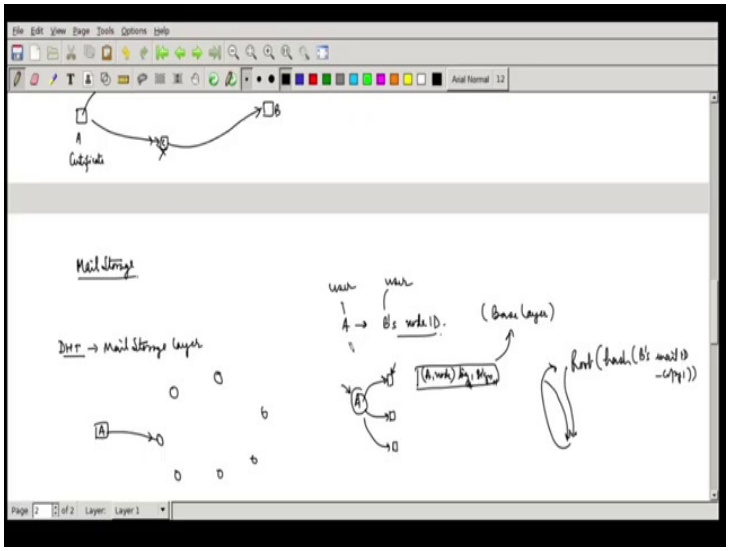
Normally mailing and list both are actually always go hand in hand. Now, before going to the mailing list implementation, let us talk about how the mail actually will be transferred. So now we are going to do away with this particular system, we do not require IMAP or SMTP server, we do not want this. But one thing is sure when first time you are going to actually have a certificate being generated, you have to receive a mail.

And so your client will be talking to a server which will give us certificate, so certificate signed by this guy will be accepted by everybody that is our assumption so you get an identity. You have to just go there and get a certificate maybe once in 2 years if the certificate is valid for 2 years, or if you lose your keys, the private key then of course, you have to go back and get another certificate.

When you go here, you actually tell your email id and everything you tell your public private key pair which we generated, we generate a certificate, the certificate goes here, this guy has to sign and send it back. But this also reads in our certificate or email id. It actually sends an email through this conventional system, you receive that OTP, you put it in, send it back so there is a confirmation then it actually signs the certificate and comes back.

So now your email id is part of a certificate. So after this, this guy is no more required, the Certification Authority or authentication server I call it in this will no more be required after this. Now this user has an email ID, this has the certificate with email ID which is signed by the authentication server. And now it has to talk to some guy, he has to send the mail to somebody. And as I told because they have made both may not be alive at the same time, we need to have a mechanism by which they can take send the mail to somebody who can be then taken up by it; we require something called mail storage.

(Refer Slide Time: 25:45)



Now, the important thing here is, this mail storage and if you look at the DFS we have done DFS earlier the distributed file system, in that case files are stored and they have been replenished or they have been moved around the root, but it is not a transient storage, in this case, A will send a message to C and it will be just held up with C till B receives the message, the member B retrieves the message this message can be removed from C.

It is only a transient storage, it is not a storage of that kind which was required in the file system or where we create a separate storage DHT, so for this we actually thought of

carefully in B4 and B4 design, we have now separated, created a separate DHT layer for mail storage, we call it a mail storage DHT layer.

Some nodes will be participating and they will have some storage. So and of course you can be if you are actually want to send, A want to send a mail so this can be part of this DHT which is running, mail storage DHT or it can be a leaf node. If it is a leaf node, it will be telling because you do not need to put the user ID and node ID will be in the base DHT. You do not require putting that thing here, but for example, if you want to send a mail to B, B is off.

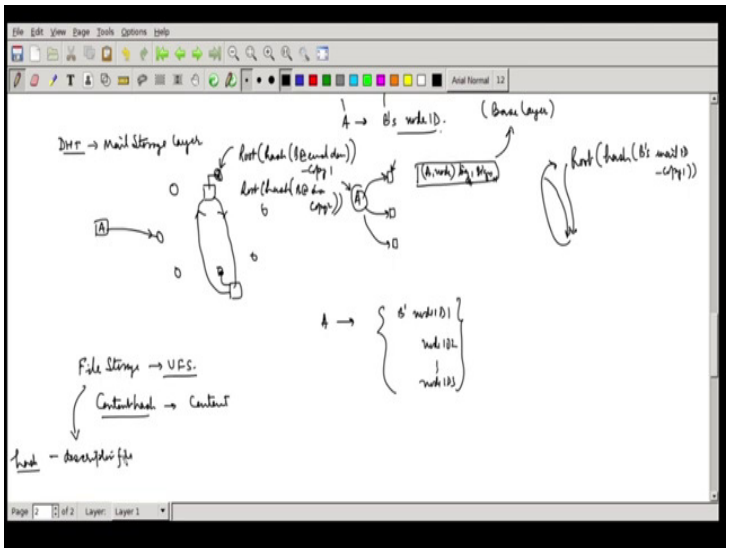
If A will search so A will now going to search for B in these node ID so A will now search for B's node ID in the base DHT in the base layer. Of course, now we are assuming that a user these are users, A is a user remembers it is not a node ID, B is also a user it is not a node ID.

And node A, user A can actually can actually connect on his mobile phone, can connect on the laptop, can connect on another mobile phone, they actually can run multiple node IDs. So every time he is actually using this, he can actually write, submit A to node ID mapping which can be signed by the private key of A and the private key of the node ID, node ID remember is a public key part so there will be two signatures; one is signed by A, one is signed by node and this is what is going to be pushed into the base layer.

This information is also available in the same fashion. What A will do is, A will now find out in fact, A will now first of all go to the base layer, send a query about B. The query will go and it will go to somebody where the one of the nodes which is the route of hash of B's mail ID. And of course, we will be having copy 1 and copy 2, so copy 1 you can search then also the copy 1, reason being why we do it because there are now two route nodes and they will be keep on publishing, they will be monitoring each other.

If this guy is monitoring the copy 2, and this guy is monitoring here, so if this guy dies off, this guy immediately republishes it and copy 1 comes alive again, which in turn start monitoring back to this thing, so this repairing can actually happen. There is also somebody which is going to have these email ids information. So there at this node, this information about whatever mode IDs, so all devices or list of all devices will be available there.

(Refer Slide Time 30:09)



So A can retrieve all the node IDs from there corresponding to B. And once this actually has this, what it will do is, it will now talk to each one of those node IDs one by one, A will talk to B's node ID 1, node ID 2, node ID 3 maybe or as many devices, if anyone of them are alive, it will simply now do a mutual authentication.

The same mutual authentication procedure which we have discussed in the initial lectures is executed between them, once that is done, A will just transfer the whole mail and all attachments to that particular node. Now, there is also an issue because the mail should be synchronized across this, I will come back to that slightly later, so the mail is done. But suppose if none of these nodes are alive, then what is going to happen.

In that case now A has to figure out a place where to dump the mail, and remember A will put the mail there for example, if this is going to the Root (hash (B's email IDcopy1)), there are also going to be copies which are going to be maintained here, In fact, this actually is not required because I am not storing any key value pair.

What it is going to do is, the reason why we do it copy 1 and copy 2 because mail can be copied at two places, so A search for this, A will also search for Root (hash (B's email IDcopy2)). this is a domain name again copy 2, so then the 2 nodes which will become available to him. What A will do is, A will now send a message whatever mail which was to be sent to B will be sent to these two root nodes. So this will be also storing it and this will also be storing it.

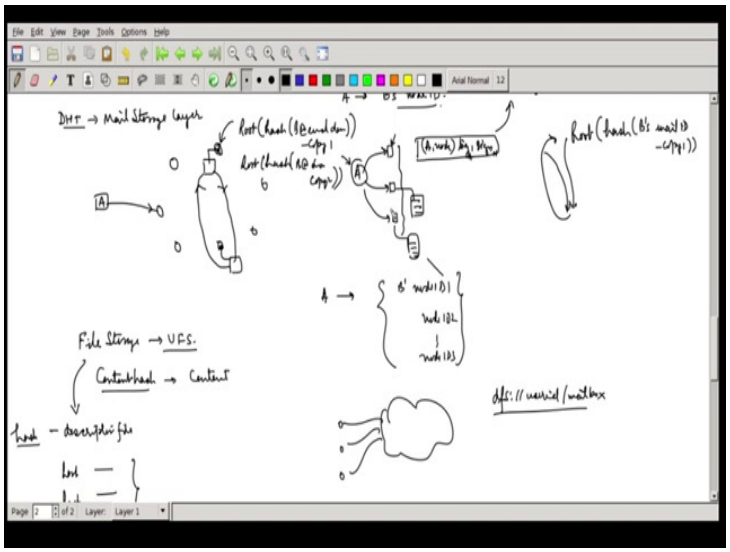
The moment B will come alive; it will pick up the message from one of them. But it will also then inform the other guy which will also purge that particular message B's having the message now, The reason for doing this duplication is if this guy dies off, so they are actually essentially doing this backup of messages to each other.

Messages are always available, it is not that this guy dies off, B comes at the time and B cannot find out the message, there is automatically resilience being created by this route. So we are recurrently using this particular mechanism of copy 1 and copy 2 to provide resilience everywhere in all DHT layers. Including, you can actually submit the attachment here. So that is going to be the best way of doing it, in fact, we can do it in another way.

If I do not have to distribute, I do not have to enter an attachment. It is a kind of, for example, I want to send a movie to you, I do not want to encrypt it, in that case I need not actually send it this way I can actually push that thing into my file storage, there is a file storage, which we have talked about earlier.

There I have talked about the UFS universal file system. So we have the contents hash is going to be the key and the content will be there so it can be for movie, so there is going to be descriptor file, note directly there is going to be a descriptor file. The hash of this is what is going to be used for looking for it, whatever is the hash value of this descriptor file I will actually generate a copy 1 and copy 2 will be appended and hash will be generated, those will be the route node. So this hash to descriptor mapping will be kept at those two nodes, again similar mechanism.

(Refer Slide Time 34:18)



Once you want to find out you go there, get a descriptor file which will contain now the list of form fragments which are there and the hash value of each fragment. You can just add copy 1 and copy 2 and then compute the hash of this and find out in the file storage DHT the route node where the copy 1, copy 2 goes and from there you can actually fetch this key value pairs.

We can fetch everything and create the whole file. This has been discussed earlier, so this is what can be done. So you can simply while sending email can send the descriptor for this and say you have to search it in UFS, so the receiver B can actually search in UFS and fetch the whole file and all the fragments can assemble the file back. You do not have to store the whole file, but anyway even if you have to store a few things but you cannot actually put in kind of movies kind of thing in few GBs, the whole file, they are actually doing it this way is far better, because you are doing fragmentation so it is not loading any one particular thing.

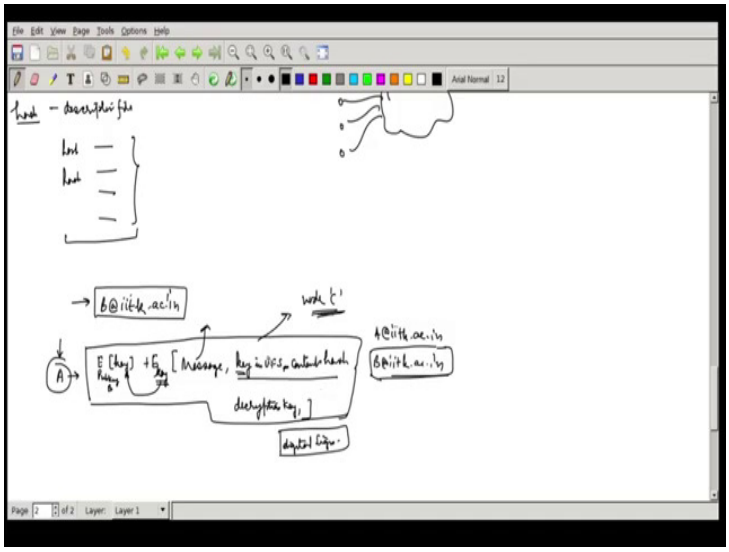
Only problem is if you are actually encrypting this so that the other person can retrieve it, you have to give a timeline on this. After this, this is going to be paired because this is encrypted, nobody accesses, it anyways going to be dead. It will not be a perpetual thing; it will be having a final bit in that case with that timer associated with that. So within that, B will retrieve this whole thing and B will decrypt it. You can encrypt each one of them either with the key can be sent to B from A in encrypted mode, and B can use that key to decrypt all this stuff, so that is another way of doing it.

Once this actually files are being received, the important thing is that the issue which remains is how to synchronize files across these multiple devices. Now for this, what is done is each one of these devices has the user certificate. These devices can actually access the same file system in the storage DHT.

This file system is routed at the user actually, this is not the node dependent, it is a user ID which is the email id slash whatever it is. So I can now create here a mailbox directory, so whenever one node gets it, it will actually have a cached entry for this file system so everybody has that cached entry. We have to first of all update in that cached entry, which in turn will get updated into this DFS. So automatically this is available to every client.

When I want to look at the mail thing, I will actually fetch the same mail from this DFS user cloud system of the storage, but every device you will be seeing the same mail actually coming up. That is how it is actually going to be maintained.

(Refer Slide Time 37:12)



Now, regarding the issue, how the message is going to be encrypted, that actually is another issue. So normally how the mail will be encrypted so that the intermediary in the mail storage DHT no node can decide for it. . What we do is, if we know that I need to send to somebody like B@iitk.ac.in, I want to send a mail ID, I want to send a mail. I actually will go out into the base layer.

Base layer also contains all certificates; all users' certificates will also be stored in base layer. So you can get any user certificate, public certificate from base layer, again there will be copy 1 copy 2. So, I will do copy 1 copy 2 and I will get the certificate from there. So, user ID to

the node ID mapping and user ID to certificate mapping is also stored. I can actually get this user certificate, once I have that I have the public key of this user. So what A will do is, A will now get the whole message. It will be message which will be there; if you create the message it will actually find out if then UFS it has actually done the encrypted file being pushed.

All segments are encrypted so that key will also be there, key which has been used in UFS which will be present here for attachments. If you basically then it will say key it will give the content hash ID content hash by which the search has to be done. Once you get the content hash, the receiver B will be able to search it, it will then get all the things and it will decrypt using this.

Similarly, if this is going to be for that, it can now also do the decryption key for the attachments. In fact, I should say I should actually keep it, I should re-explain it. Seen in the UFS, here actually means the key by which you do the search which is the content hash, this for an unencrypted system. If there is encryption which has been done so there is a decryption key which is basically a symmetric key, the key same key has to be put here.

You can have multiple attachments for each one of them you are going to prepare this actually; this whole message is there with you. You will now encrypt this, you will do encryption with a key, this is another key by which encryption is done, it is encryption key and this encryption key is actually appended. This is the same key as put here and this is going to be encrypted with the public key of B.

A is going to create this particular message. Now, the important thing is this message when actually has been put with node C. Node C will not be able to decipher what is there inside it, it only knows there is a message that is it. The message will also contain only information that this is what we at the rate something is only for this.

This has to be indexed with this so that B can retrieve this message. A will just simply dispatch it, but then B also has to verify that this guy who is the guy who is the sender, so this message will have all that information who is the sender and everything, so it needs to verify, we will sign this whole thing by the private key of A there is going to be digital signatures. In fact, what can happen is even you can also put, who is the sender that information can also be told to node C. Whenever these want to retrieve this, node C can send a message back to A that this message has been retrieved by B so that is the kind of acknowledgment will come to

A, A will know that yes, no message has been retrieved by B, B will be able to see the message actually now.

By verifying with public certificate, A's certificate can be retrieved by B by looking at the sender here and he can actually from their public key of A, it can verify whether this message has indeed not been tampered with. If the message has been correctly received, the message is okay and it can open it up and it can read, so message that is the way the message will actually pass from one to another and nobody will be able to figure out in between that what is the content of that particular message.

Similarly, normally this message has to be signed, if the attachments are in UFS, whatever you are going to put in UFS should not have been tampered with that also needs to be digitally signed.

And normally, you are going to retrieve that attachment and put it back in your mailbox in your UFS system. So you do not need to actually put it into UFS for a perpetual thing or for infinite time. In fact, there is nothing I can find that only perpetual flag, whereby the timer kept on getting reset if you access the file, if you do not access it for a certain period, then it gets pushed.

So that is how the mailing system actually operates. So the next issue is how we are going to have the implementation of a list server. How the list server is going to be implemented. I will talk about in a brief video, which will be the next one in the same series. So as of now we close at this point, and let us look at the next video for figuring out how the list server works.