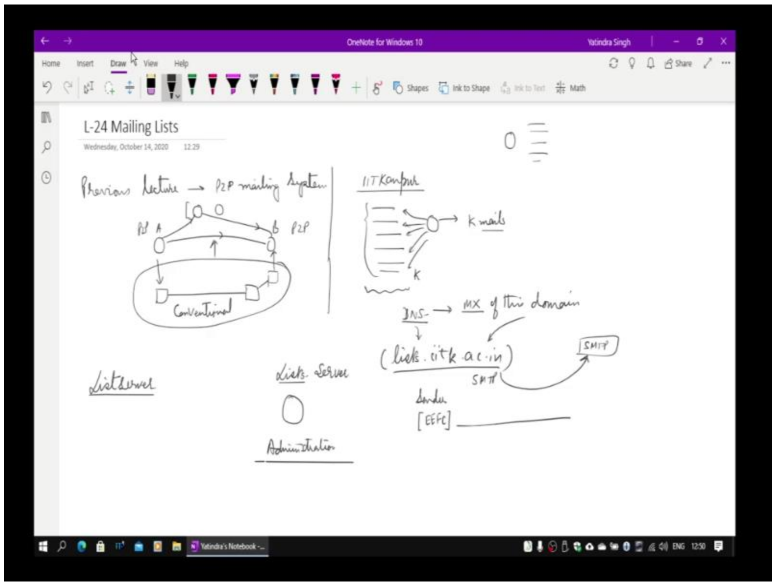
**Peer to Peer Networks**
**Professor Y. N. Singh**
**Department of Electrical Engineering**
**Indian Institute of Technology, Kanpur**
**Lecture – 24**
# P2P Mailing List Services: A Basic Design

(Refer Slide Time: 0:16)



This is lecture number 24; we will be talking about mailing lists implementation in this particular lecture. In the previous lecture, I had talked about in the previous lecture, the peer to peer mailing system. So, the mailing system was always one user A to another user B, so we did it instead of routing it to some SMTP server. Then, going to another SMTP server and then going to an I-map server, and then B accessing it. We try to set it up peer to peer directly, and for this, we use some other nodes, which were there in the mailing storage DHT network.

So, we use them as intermediate storage if A and B are not alive. So, we can pick up, so A deposits and B picks up; and it is done so that A digitally signs it, but it encrypts for the B, only B can decrypt it. B verifies A has sent this particular message and decrypts; both are maintained non-repudiation and secrecy. Here, the non-repudiation and secrecy will depend on whether B else accepts A to connect and send the message. And these SMTP servers are doing verification of each other via reverse zone mapping.

Reverse zone verification is dependent on DNS resolution for both IP address to DNS name and DNS to IP address. So, both verifications are done, and the same organization owns these servers; they rely on each other. B can retrieve the message by authenticating this, so this B authentication to this and A is authenticated. B is that authentication is not aware to him, so it is only just passing on to somebody responsible for this domain. So, we replace this whole thing, so we just replaced this whole mechanism, and we want to peer. So, that is what we did in the previous lecture.

Those are few who have been using the mailing list, so we sometimes, for example, in IIT Kanpur. So, I am in electrical Engineering then I want to send a mail to all electrical engineering faculty. One possible method is to maintain my list, where I will list all faculty members' email-IDs. And my client will now read each one's email-id and will create those many copies. And these copies will be dispatched either via the conventional mailing system; this is a conventional one or via peer to peer system.

Then peer to peer system will nearly send if they are K faculty members here; there will be K mails and dispatched by him. This is usually one possible way of doing it; there is another way we maintain something called a list server. So, this is a special sever array machine, so this is a list. The advantage of this is that I need not, in this case, in the scenario where I am maintaining a list of my own. So, every faculty member has to maintain his list of his own. So, a new faculty member joins, I have to add my list here; everybody else in the faculty member has to do it.

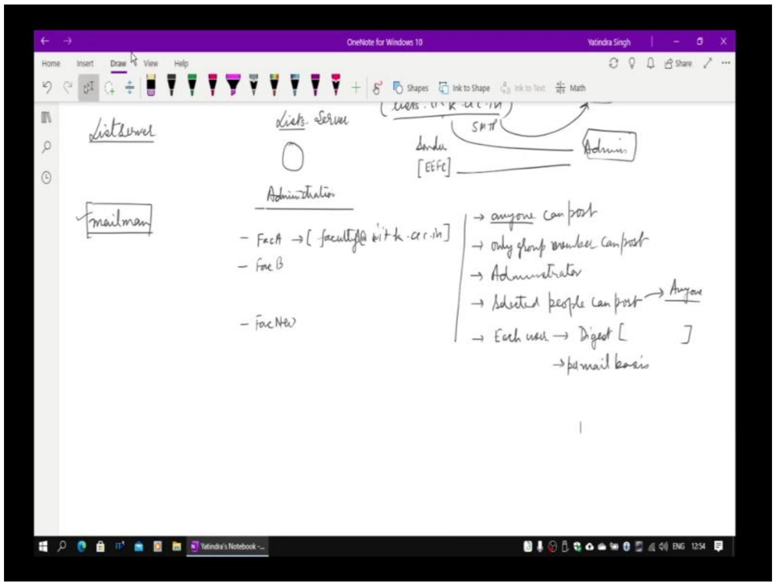
So, if somebody fails it and he would like to send an email to all faculty, he will not be able to send it, somebody will be missed out. So, people came up with the concept of a list server. So, in this case, a list is going to, a list is a separate server, we call it list server. In IIT Kanpur, we run it with the domain or list.iitk.ac.in; these are domains. So, if you ask DNS, DNS will always resolve this as the mail server. And of course, the MX record is also maintained for this particular domain, so the MX record for this domain is also this same machine itself. And this machine is being, is running like an SMTP server.

This machine can also communicate to IIT Kanpur's SMTP server, so this machine is authenticated authorized to send mail to anybody. It will show that both are sending mail, but it is going to come with some mechanism. It will indicate that this was a mail sent to lists, and there is a list name.

Generally, we will have a list name, where it will say EEFC; and then the title will be there. The topic on which it has been done and then the sender will always be the original sender, who has sent the mail to him. This will essentially use that as a sender, but we can look into that it originated from this particular mailing list server.

Now the administration, so there is only one place where the whole list is maintained, will maintain a data structure. So, we have to remember that we will also do the same thing; when implementing it in peer to peer mechanism.

(Refer Slide Time: 6:16)



So, the administrator will now maintain a list to maintain faculty A, faculty B, etc. Whenever some new guy joins in, it will put some faculty, and new will be added to this list. This is nothing but an email id of that person, so this will be essential but written as faculty A@iitk.ac.in.

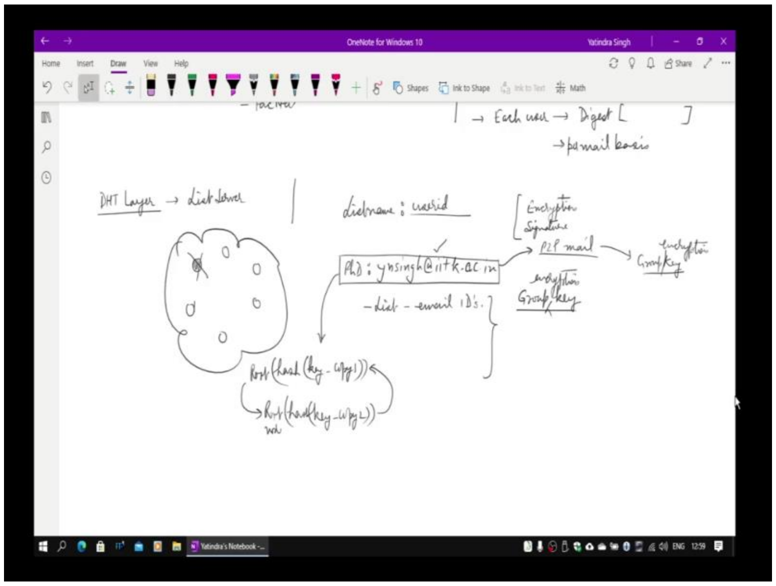
We will also have those whom all can send the mail to the list, so you will say whether this guy is permitted to send or not. So, there will be settings for each list, whether anyone can post on the list. So, even if he is not a member here, he can still post; we can use an open-source called mailman. This is an open-source list server, which is available; which can be used to configure all these emails. Anyone can post; only group members can post; we will also have an option if the only administrator can post.

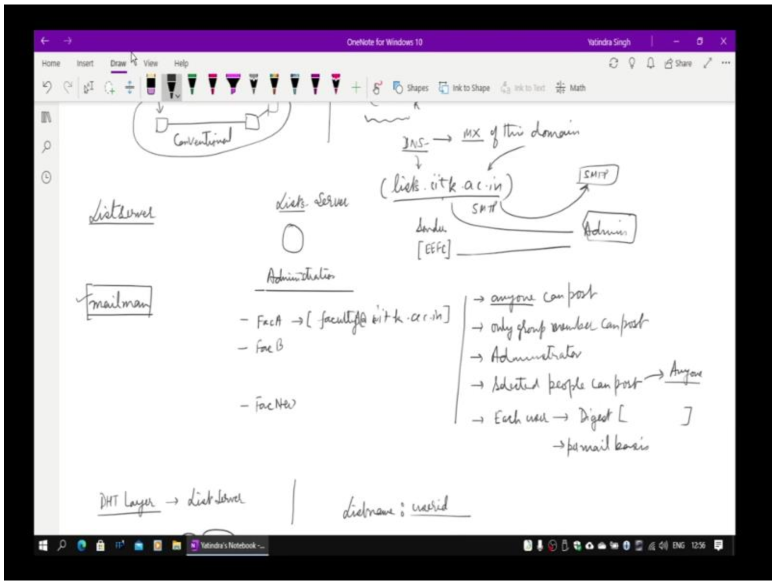
The administrator can always post actually, or there will be only selected people. It can specially select special people, who can selected people can post. So, whenever a mail comes, all this logic has to be tested and then only mail will be permitted to either be forwarded to every member; or not to be forwarded to every member, and this check has to be done here. So, once this check fails, the mail will not be forwarded; so, all this mailing will be done. Anyone means that person need not be even member of this, member of this particular group.

So, group members can be there and then selected even from outside, from the group of anyone can also be there. We create a filter, a restriction here; we can also have it for each user. Each user can have a setting that will be a digest mode; all the mails dispatched to this particular group in a day will be combined. A single mail will be sent to the user or user mail basis, per mail basis, so both ways it can be done. So, we have many options of this kind, which have evolved and added to the system.

We want the same thing to be done here, but I do not have a single server; that is the problem. And then I also do not want the list to be modifiable by anybody else except the owner. In this case, the owner, so the owner, in this case, owns this particular server, who is having the admin rights on this server. So, the admin here is what controls the list. So, in our case, in peer to peer system, and I will do it; we will do it in a slightly different way.

(Refer Slide Time: 9:47)





So, we have handled it to create a separate DHT layer, another DHT layer, and we call it a list server. So, any node can act as a list server. That is the essential thing, so they all will form a separate DHT network to be in a. So, now basically what is happening is anybody can create a list will be uniquely identified. So, there will be a list name which has to be there, and they have a colon as a separator; and then the user id. So, I can create my Ph.D. students and then I can say this is my Ph.D. students; this is I am creating, so I am the owner. So, these are the unique identifier for the list.

So, when I create this list, I need to create a data structure regarding the configuration; so, I will now put up a list of email ids. Another important thing is that now I am using some third party acting as a list server. Only these people should be able to read the mail which is sent to this group. So, even if he does, this guy does some problem and once copies mail to somebody else; he should not read it. So, we also now start using a group key.

So, in the earlier system we have here when we are using the list server, we are dependent on the administrator because controlling we have faith in him. So, in our case, there is a computer centre head who takes care of this; or an engineer on his behalf takes care of it. And so we have, but if we do something incorrectly, what they do miss configuration; you cannot stop. Because I do not own the list server, somebody is acting voluntarily as a list server. So, we cannot allow anybody else to read except the group member, so we need to maintain a group key.

So, for everybody who becomes a member of this list, I need to send a group key to him; so, this is done through a peer to peer, point to point mail system. So, I will send a mail message telling me that I have created this particular thing, the group key. Now, remember this particular messaging peer to peer mail is encrypted, and the source can figure out, the source can be encrypted; so that only the sender can get it. And of course, the sender will always, receiver, the sorry receiver can get it, and the receiver can figure out who the sender is. So, there is non-repudiation, there are signatures, which are there are digital signatures.

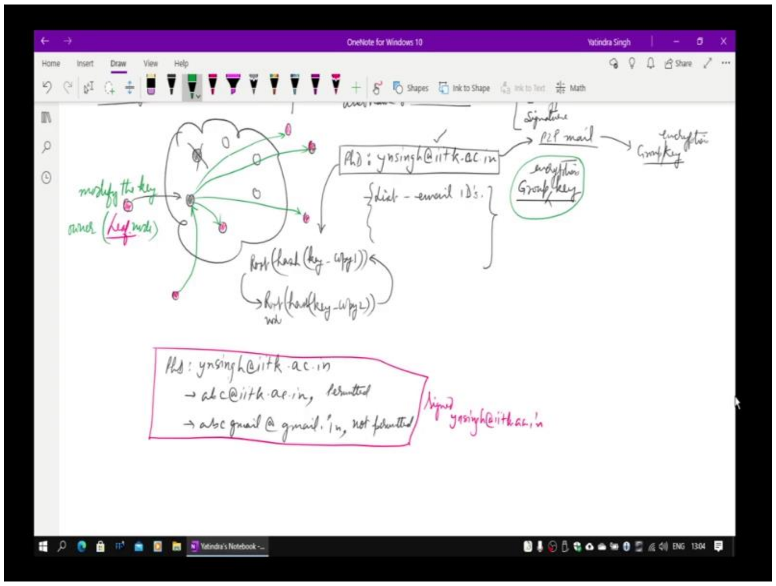
There is tempering in between that we can figure out. So, this nobody else can read except the receiver, and nobody else can send except the sender; so, the group key can be communicated to every; this has to be done if you have 10000 guys; the moment you add somebody, you have to send the message to him, this your group key. It would be best if you also had a mechanism somebody forgets the group key; a mail comes, this guy then can send them a message back and wait for a group key to arrive.

So, every user needs to maintain all the groups she has created and all the encryption keys. It can also change the key so that everybody will have the new key. So, what will happen is this will act as now my search key; so this key is for the encryption; there is a group encryption key I should call it. This will be my list, list name; because I operate because every user can reuse Ph.D., so Ph.D. with the different, with the different user is a different list. I can now actually create from here; this is my key, search key.

So, whatever is key, can copy 1, remember this copy 1, copy 2 we have been using for resilience. For every key whose hash I am, before I compute the hash, I will add a copy1 and compute it. And then find out who is the root node for this one in the DHT. And this root node will also now create find out using the same thing copy 2 hash, and there is a second root node. So, this guy will be now doing republishing periodically here; this guy will republishing here.

So, even if a node dies off, entries are not dead; so root 2 will figure out, and we will republish. Again, copy 1 will get recreated; they will essentially keep on provide feedback to each other to maintain stable and reliable entries inside the DHT; so, we are just using the same thing here, which we have used earlier.

(Refer Slide Time: 15:09)



So, now, in this case, if this guy now publishes a list like this Ph.D., I am sitting here on this device. Remember this is a user-dependent; this is not node id dependent, so that a user certificate will be used in this case. So, I will create this list; I will now create an entry for this.

I will now add all my email ids and everything; I will create a data structure; so, there will now be a data structure with the list name. Within this list name, I will have all the email ids of my students, so abc@iitk.ac.in. Some students want two email ids to be put in, so I can go abc, and he has also got an account at Gmail. And of course, I will also can but the important thing that here can transmit, who cannot transmit these permissions. Of course, I can put in, but it all depends on this list server; whether he controls per the permission, I can say he is permitted to post.

I can say this is not permitted to post. This is an openly available list, but this only available to this guy, nobody else. So, I have published my list here, so many group members are sitting in, so group members can also participate in DHT. So, for example, there is a group member, this also a group member, this also a group member, this a group member and this one is also a group member. And of course, I am also a group member; because I am the list owner. So, this guy wants to send a message about how this will be happening? So, it knows, it does not know the complete list, list I am managing.

So, whenever I am doing the updation of the list, this list has to be updated in this DHT; so, I am acting here as a leaf node. So, I can also be participating in this DHT that is possible; but where my list which will, which root node will become responsible is not in my control. That is purely random. That depends on the cryptographic hash.

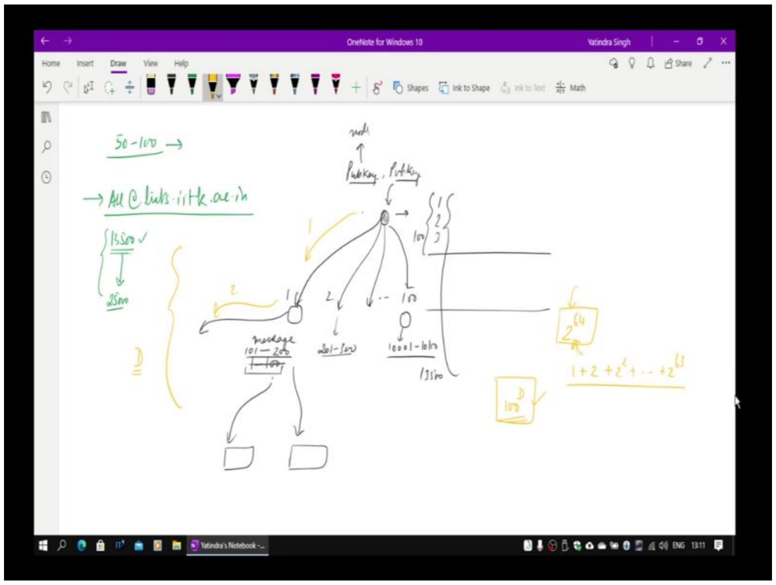
I will publish my list here; I will digitally sign this list or this data structure. So, this has to be signed by my user id's private key; and with my certificate, you will be able to verify. Remember any user certificate you can retrieve from the base layer, base DHT layer that is possible. So, just let me keep this figure here.

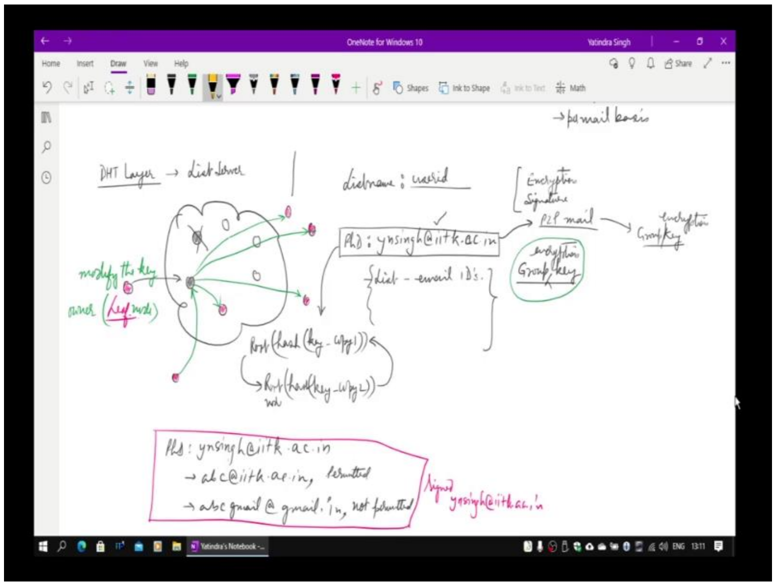
So, in this case, now this guy wants to send the message of what will happen. This guy now sends, will find out who is the root node for this; he will create whatever is this list name, copy 1, compute the hash. Then, find out the root, go to the root, so even if this node dies off, somebody else will become the root node. So, copy2 will ensure that it goes to the new root node for copy1. My request will always be reached there, so my request is here; it will look into the list now. It will look into the list; it is his job.

Remember this message is encrypted with a group key; so this key is being used for encryption. So, nobody else can read except the group members, who have individually got the keys. This guy will now look into the list one-by-one, and it can now start dispatching the messages to everybody. So, far these numbers are small; it is perfect; it sends the copy to each of them. So, everybody will figure out this has been digitally messaged, has been digitally signed by it; encrypted by this key, send to this particular group. So, using that group, they will be able to decide on it.

Remember the change if you send it to point-to-point, you say, you can use the public key; but, group key has to be distributed to everybody. That is the only way it is done, and the owner can modify the group key. This is the owner who will, the owner acting as a leaf node, and can modify the key. So, this is the way now the list is being made.

(Refer Slide Time: 19:45)





Now, the problem which will face here is. So, far the number is small, say 50 or say 100 at best; this mechanism will be okay. But, suppose if in IIT Kanpur, for example, I have to send a mail to all@list.iitk.ac.in; so everybody who is having a mail account in this domain iitk.ac.in a domain or whoever the member of this list is, they all will get this mail.

Now, this number is roughly about 13500 as of now. And as my number of students and faculty will grow, this may become even very large. This may become 25000 or maybe a lakh; if this is a huge institute. Now, how to sort out this particular matter? So, one way is that yes, let the server be get loaded; and based on that loading, this same guy will keep on sending a thousand copies to everybody.

So, that could be one possibility; but we can do something better than this. What we can do is this list is all available with the list server. The list server has an available list, and it is an ordered list is say 1, 2, 3, and so on, may say 13,500. So, what I can do is I can now partition it, so the list server, instead of sending it to everybody; will say that send it to only say 100 guys, not to all 10000 and it will send a message, it will send the message as it is intact; it will send the message but message will now. You would also have some to say that you pick up 1 to 100 of the first 100.

So, maybe what happened is the first hundred guys are being picked up, and these are those people so that I can do that. So, this guy can be told that you pick up from 101 to 200. This can be said that you pick up from 201 to 300, so everybody has been given an index. This guy will correspondingly pick up 10001 to 10100, which will load a large number. And this further actually can be this; the whole is space has been partitioned. This node cannot handle it; it can further tell two nodes we can pick up from this range. And tell them that you pick up from this range and this range.

So, in some range, it can directly communicate so that it can be split into three. So, now each of these nodes will become more responsible; they can fetch the list from here. The list then gets signed by the list server, so this has to be signed because the user, the list owner, does the signature for the whole list and data record. But, now this owner cannot sign it because the private key is not with the list server; so, the list server can only sign on its own.

It will be using node id; there are a public key and private key for the node id as a member. So, this one is using as a node as the public key part; so, a private key can sign it and send the lists here. This can then start dispatching the mails to everybody; this guy cannot even if it is a wrong email id, somehow it is a rogue. The receivers will not be able to decode because everybody needs to have a key, and the key is being distributed over time, which needs to be maintained. And before expiry again, the key has to be replenished by the mailing list's original creator.

This is how we can maintain handy mailing lists for the peer to peer system. This is an addition to peer-to-peer mail; this also needs to be done; this will also be a requirement to maintain a huge mailing list. So, their size can become huge, but the important thing is that I can handle even huge numbers very effectively because of this binary tree formation and distribution. The depth, so this is, for example, this depth is so this is this depth is 1 and this is the 2. So, the depth required is pretty small; this depth is represented by D.

So, every time if I am doing 100 for the thing being done at each depth; so if the depth is D so 100, this is power D. D will be the total number of nodes which will be, which will be catered to by the mailing system; and these are a pretty large number which can be handled.

So, $2^{64}$ is such a large number that every particle on the earth can be counted into this; this is such a huge number. Every time you keep on doubling and do it 64 times, you start with 1, 2, $2^2$ and so on plus $2^{63}$. So, this will give you $2^{64-1}$ total value.

You can do with a progression sum, and with that formula, you can prove this comes of this particular order; and this is extremely large. We can handle the mailing list in this fashion but remember we need to maintain a group key with one additional heading. This also provides security, and it is better than the normal list server; because you have to have faith in the computer centre. Now, you do not need faith in the computer centre. For smaller ones, you can maintain your local list and that your client distributed.

If you are not there, you can create a separate DHT for the list servers; and essentially compute them. Use this list server DHT to essentially do all the work for you, so how the list server is managed.