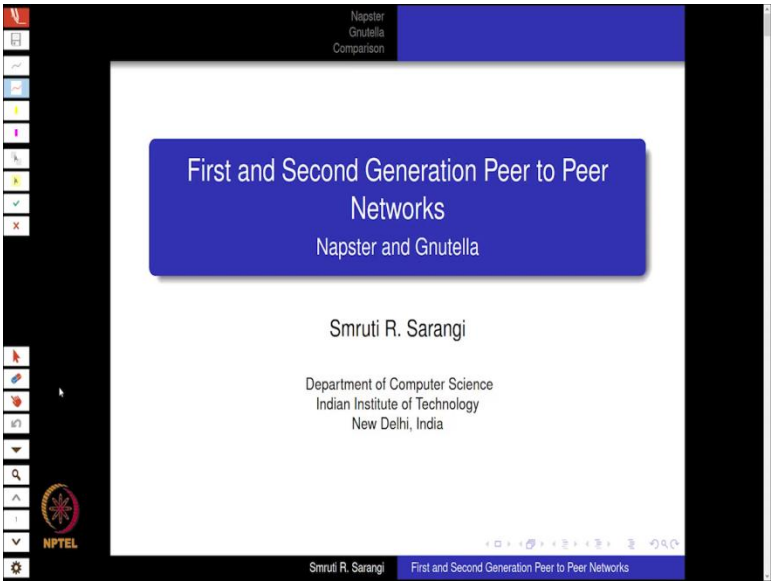
Advanced Distributed Systems
Professor Smruti R. Sarangi
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture 2
1st and 2nd generation peer to peer networks: Napster and Gnutella

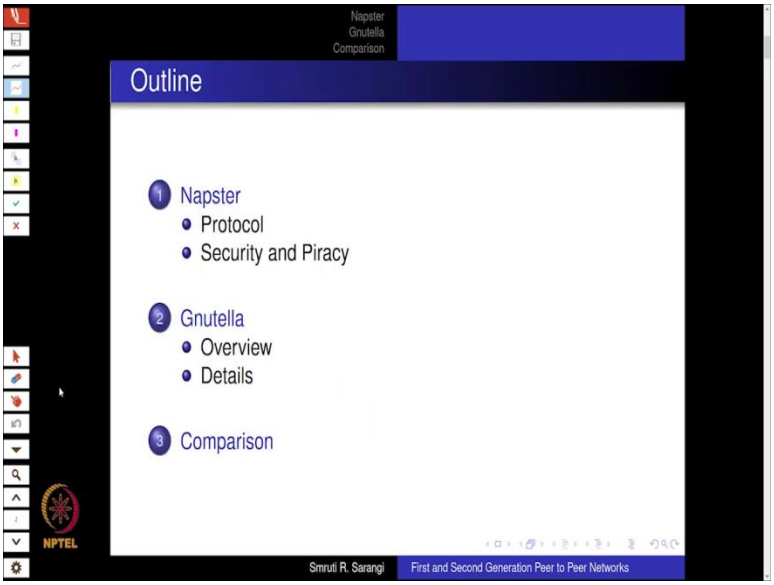
(Refer Slide Time: 00:17)



Welcome to the lecture on first and second generation peer to peer networks. So in this lecture, we will discuss two very important networks that, unfortunately have acquired a little bit of a bad name primarily because they were used to share unlicensed, and in some cases illegal content. But they have helped to a large measure determine the direction of work in this area. And furthermore, most advanced distributed systems today, or they arise to search peer to peer technology. So we will discuss Napster and Gnutella.

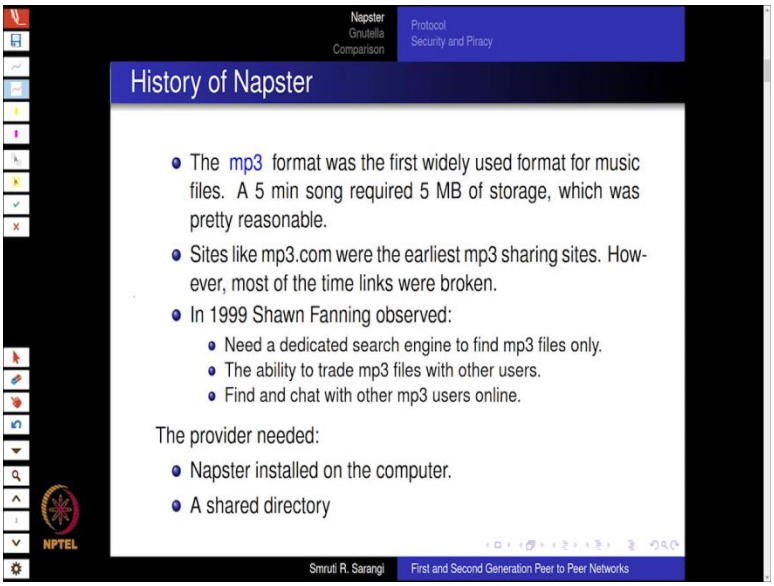
So Napster was considered the first generation peer to peer network. So what is the peer to peer network? Here we have a set of machines. No machine is more privileged than the other in general, and they cooperate among each other to provide a large number of files the case of Napster music files to a worldwide community of users. And a second generation kind of Gnutella is an improvement over Napster. We will discuss how?

(Refer Slide Time: 01:29)



So the overview of this we will discuss the protocol security and privacy issues with Napster. Then we will provide an overview of Gnutella. So we will discuss the details. And finally, we will do a little bit of comparison of Gnutella and Napster.

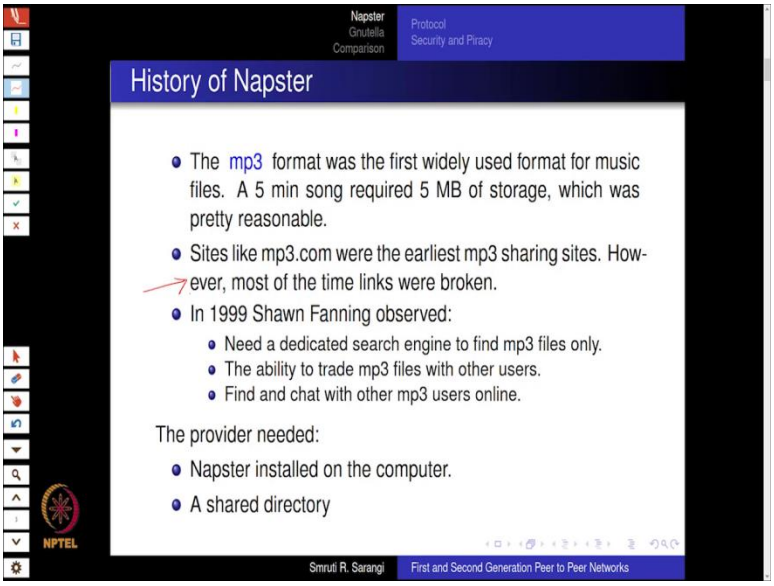
(Refer Slide Time: 01:50)



So what is the history, so this history is roughly towards the end of the 90s. The mp3 format was discovered mp3 is an audio format. So what people those days used to do is that they would record these wonderful five minute songs is five minute mp3. Idle, roughly an MB a minute. So five minutes was five megabytes, which was something just about the internet those days could handle.

And the main advantage by the way of mp3 was that as compared to the previous format, which were wav files, mp3 format was somewhat much more compressed without any great loss in quality.

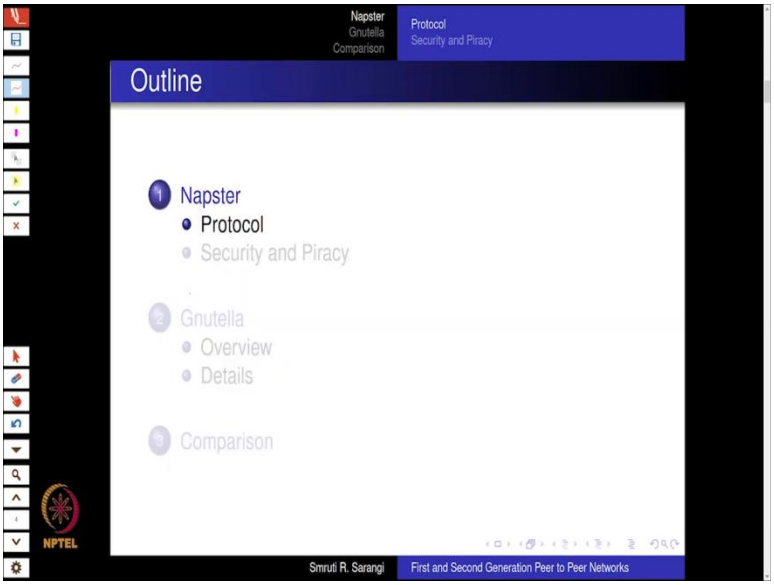
(Refer Slide Time: 02:37)



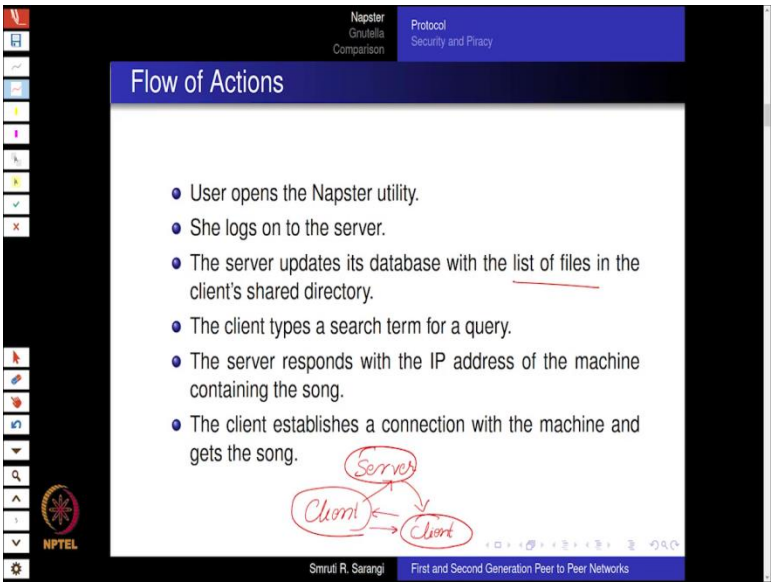
So that kind of started a boom of mp3 sharing. And a large part of that sharing was an unintended consequence of technology, like Napster were a large part of this was illegal. And Shawn Fanning, who invented Napster created a dedicated search engine for mp3 files only. The ability to find trade mp3 files, chat with other users and an entire community meant to exchange and distribute mp3 files, audio files, music files, to whoever required it.

So the provider needed Napster installed on the computer. And there was a large number of shared directory servers over the net, we will discuss that. But primarily all that was required was a Napster client on a computer, nothing else was required.

(Refer Slide Time: 03:37)



(Refer Slide Time: 03:41)

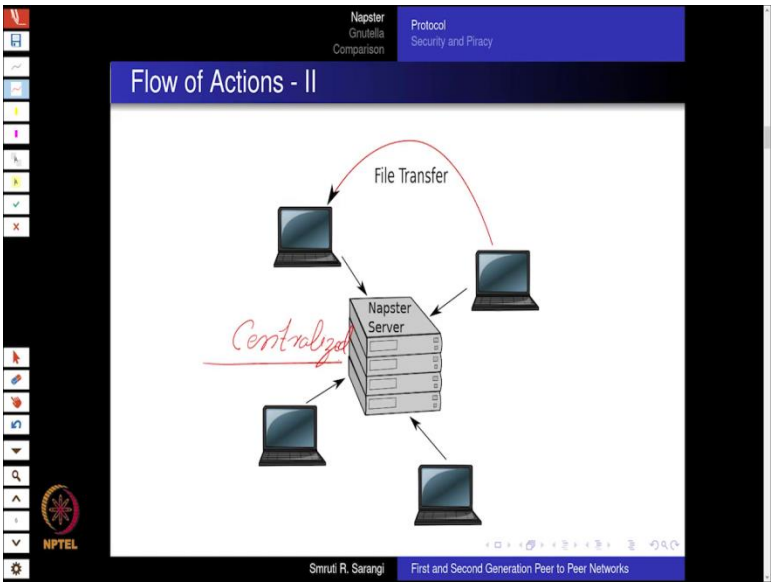


The basic protocol now, so what the user used to do so this is a first generation peer to peer network around 1999. So what the user did or let us put it this way, all that the user had to do was open the Napster utility, log on to the server. The server would update its database with a list of files in the client shared directory. So every client had to share a directory with the mp3 file so the client owns.

So in a certain sense, these files are made available to the community. Then let us say the client wanted a certain file, the client would search and write a query string and search for that kind of file. The server would respond with the IP address of the machine that contains the song. The client

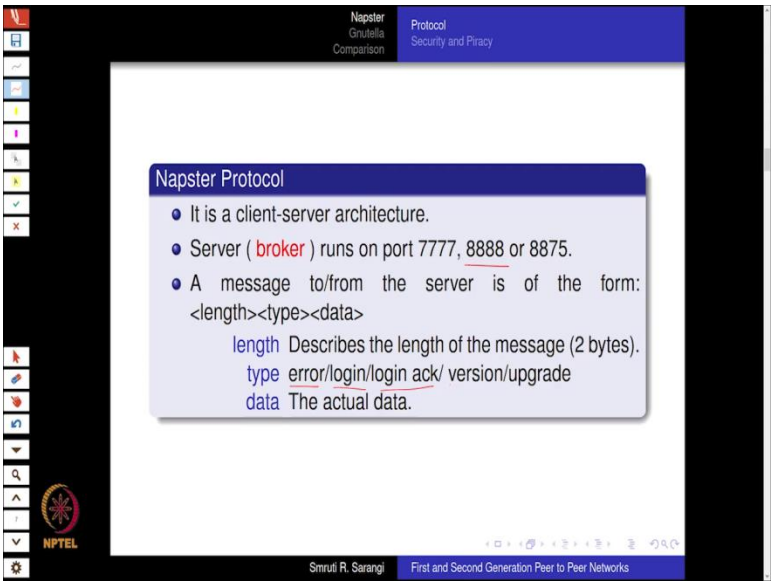
would establish a connection with the machine with that IP address and get the song. So all that the server would do in this case, is that it would act as a broker. So once the client logged in the server would get a list of songs that the client has. And furthermore, if there is a search query, the server would essentially connected one more client and then a direct connection could be established between them and could be transferred.

(Refer Slide Time: 05:02)



So what was the architecture again? Well, the architecture those days, given that it is a first generation peer to peer network, this was in centralized architecture, with one central Napster server in the middle with a lot of client machines around it.

(Refer Slide Time: 05:25)

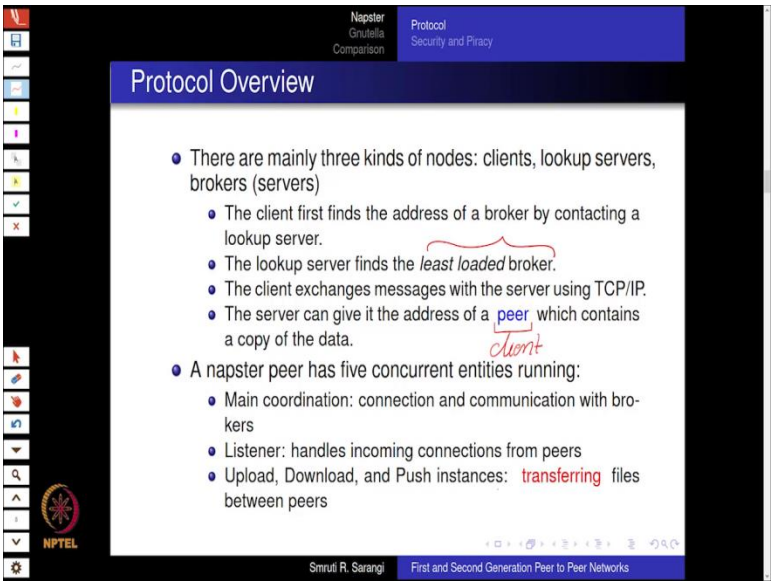


So it was a quintessential example of a client server architecture where we have one centralized server. The server runs or used to run on these three ports. 7777, 8888, 948 and 8875. So when universities later on started banning Napster, primarily because songs without legitimate legal licenses were being shared. What they basically did is that they closed down these ports.

And to a large extent, at least, most Napster clients had a major issue. So the message format between the client and the server was very simple. It was length, type and data. The length describes the length of the message, it was limited to two bytes, sixteen Bits, the type well, error, a login message to login with the password, and login ACK and acknowledgement message. And also what was the version of Napster that has been used, and also whether the version of the Napster client need needed to be upgraded or not. All of this was included in the type of the message. And the data in this case was the actual data the actual data that was being transmitted.

So this is what, in a nutshell, Napster message actually contain, which is very simple just length, type and data. So nothing. So if you look at Napster right now, in 2021, when I am recording this video, Napster would appear to be a very simple protocol.

(Refer Slide Time: 07:09)

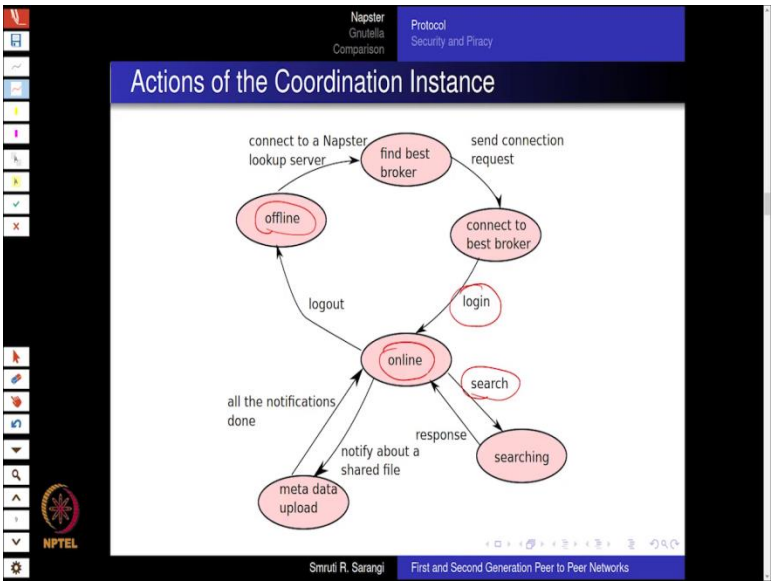


So there are mainly three kinds of nodes, clients look up servers and brokers, so brokers will also called servers. So the client first finds the address of a broker by contacting a lookup server. So a broker is basically the server that does most of the job, but the lookup server is you can think about is a special kind of a server, which just gives the address of the least loaded broker in terms of the amount of work it is doing. The least loaded broker to the client, the client. The will establish a regular TCP IP connection.

So at this point of time, if the reader or the viewer of this video, reader of the paper and viewer the video, are not sure about some networking concepts, such as TCP IP, proxy servers, and so on. I would request them to kindly look it up. So the client will establish a regular TCP IP channel with the server. The server can give it the address of a peer. Peer is in this case, a special term because the click peer is essentially one more client, which contains a copy of the data and in this case is a song.

So Napster peer will have five concurrent entities and entities essentially software thread. So it will have a coordination entity. So the job of this piece of software is to connect and communicate with brokers, a listener, which will handle incoming connections from peers. So if there is a peer, which is trying to connect to it, it will handle incoming connections, upload, download and push instances. So we will have ample opportunity to take a look at what these software components are. So these will essentially help in transferring files between peers.

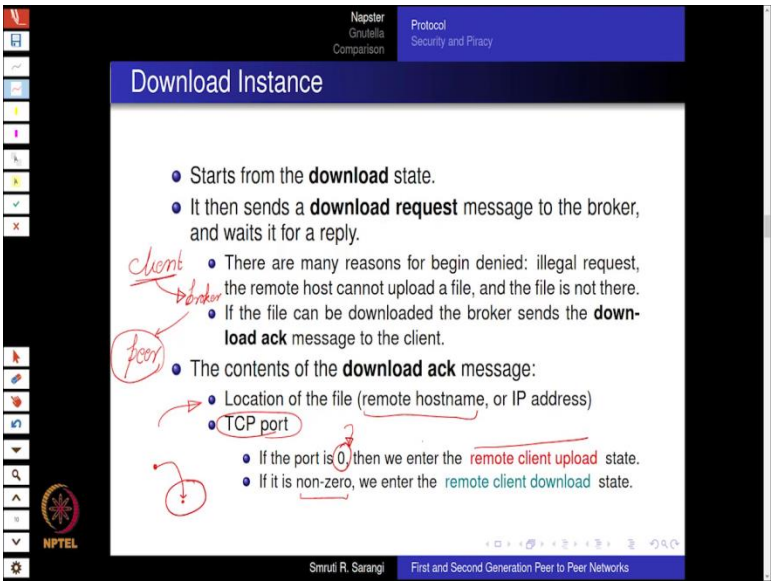
(Refer Slide Time: 09:11)



So if I were to look at the flowchart of Napster, this is what it would look like. So let us look at the online state. let me maybe start with the offline state. So after offline, when the user wants to log in, the user will connect to a Napster lookup server, that will find the best broker then its connection request will be sent to the best broker. The login information will be exchanged and if the password matches the client is online.

Subsequently, what it will do is that it will let the broker know, that it wishes to share a certain number of files, that are there in a shared directory. The Meta data will be uploaded and once all the notifications are done, you can say that the client is ready to share. After that, the client can begin searching. So let us say you want to play a certain song. So you write the keywords of the song, the process of searching again will be initiated. And a response will be sent back to the client, such that the client can then establish a direct connection with the remote peer and get a copy of the song.

(Refer Slide Time: 10:30)



So the download instance how does that work? Well, it starts from the downloads state, it then sends a download request message to the broker and waits for a reply. There are many reasons for the request being denied. It can be an illegal request, which is the remote host cannot upload a file or the file is not there, is a file can be downloaded, the broker sends the download ack message to the client, which means the client knows that look, yes, there is a remote peer that has a copy of the file.

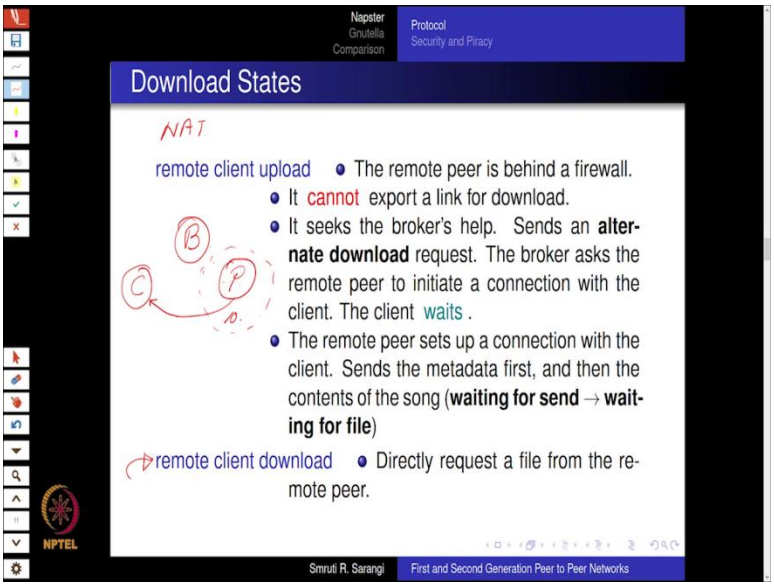
And the file can be downloaded from the remote peers. The contents of the download ack message like this, which the broker sends back to the original client. So let us say, let us refer to these terms like this, we are the client, then we have the broker whose job is to search. And then we have another client but let us call this client remote peers.

So once a download ack message received by the original client, it knows that it needs to establish a connection with the remote peer. So the information that will be sent back to it will be the location or the file that includes the remote host name or IP address and the TCP port. So let us first start with if the port is nonzero, then you will enter the remote Client Download state and there is no problem message will be sent and then we can download the file from the remote client. However, there is a problem. So let us consider a university setting. And the remote client is kind of behind a proxy server as it is the case in most universities which have their private networks and behind a firewall and a proxy and so on. So, in that case, it is not really possible for any machine outside the organization to initiate a direct connection to a machine inside the organization. So this will

not make sense. If this will not make sense, then Napster did provide a wonderful mechanism because primarily this was meant for college students to share files.

So it return the special value of zero for the port. And we entered the client upload state, which I will say so this was a nice trick you can think of this this was one of the dirty tricks that gave Napster fame as well as infamy. So Napster, in that sense is extremely controversial, because it allowed college students behind these closed networks to actually establish connections and exchange data. So a key part of this was that if let us say the remote peer is, let us say within a university, that does not allow a direct external connection initiated from outside, then a port of zero is written and we enter a different state called the remote client upload state, we will see what it is in the next slide.

(Refer Slide Time: 13:38)



The remote client upload state, slightly complicated, which as I said, the remote peer is behind a firewall, it does cannot export a link for download. So then what happens is? so we have the client here, we have the remote peer here and we are the broker here. they will seek the brokers help, it will send an alternate download request. So instead, what the broker will do? it will ask the remote peer to initiate a connection with the client rather than the client initiating a connection with the remote peers. So the client will wait. And since it knows that the remote peer is behind, of all right, and network wall, it will ask the remote peer that look, this is the address of the client. And what

you do is that you send a message to the client. And then once there is a connection between both of you, then you can transfer the file to the client.

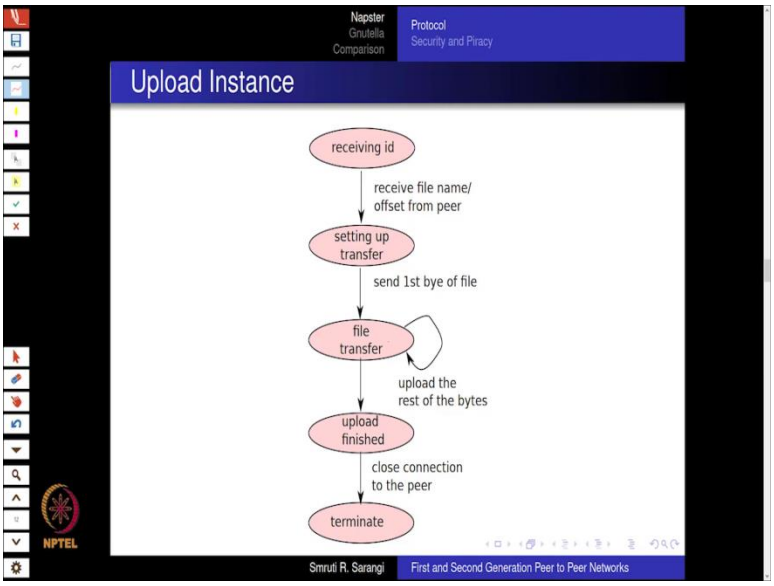
So if somebody has not really if somebody really does not know what is a NAT and network address translation, and what it means to actually be in a closed network like a university? I would suggest that you kindly look up this piece of metal working. And so this is easily understood for somebody who knows the concept. Otherwise, the greatness of this will not become, it will not be aptly clear.

So if you are not able to understand kindly look it up, what you need to look up is called NAT, right address writing it down NAT network address translation. The key idea of NAT is that if the remote peer is behind, if the remote peer is within a secure internal network, it is not possible for the client to contact it. However, if the client is publicly visible, then the remote peer can establish a connection with the client, set up a connection with the client, it can then send the metadata and the contents of the song.

So then the client will anyway get the contents of the song. But it will happen via this mechanism. Of course, the simpler is if the remote peer is directly visible, then the client will directly send a request to the remote peer and get the song back. But because many university setting the remote peer as such is not accessible. The broker will say tell the remote peer that looks since you are behind, since you kind of have a private IP address, which starts with 10, which means you are not externally visible.

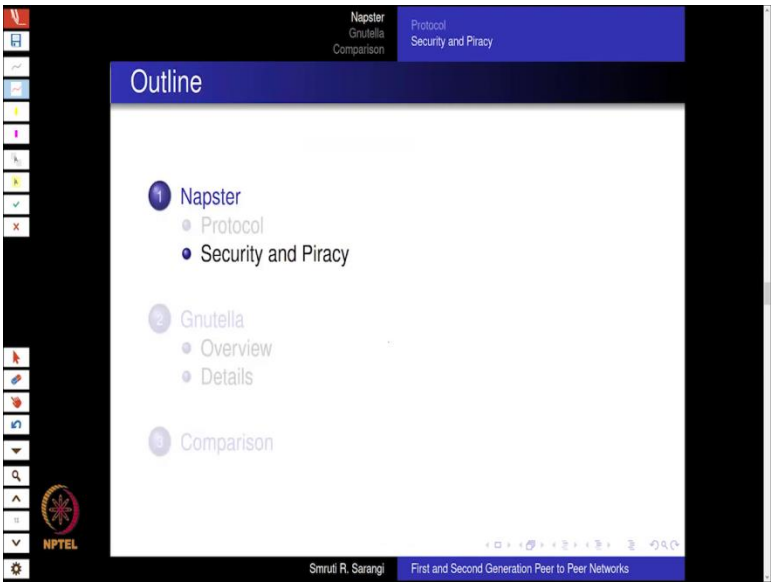
So you need to know what a ten dot.IP address means? So since you are not externally visible, you initiate a connection with the client and get the transfer done. So I hope that this sticky notion of remote client upload and remote client download is clear. If not look at NAT. But assuming what the concept is? The concept now basically is that even for users where one was kind of behind a firewall in a private network, there was still a way to do it.

(Refer Slide Time: 16:56)

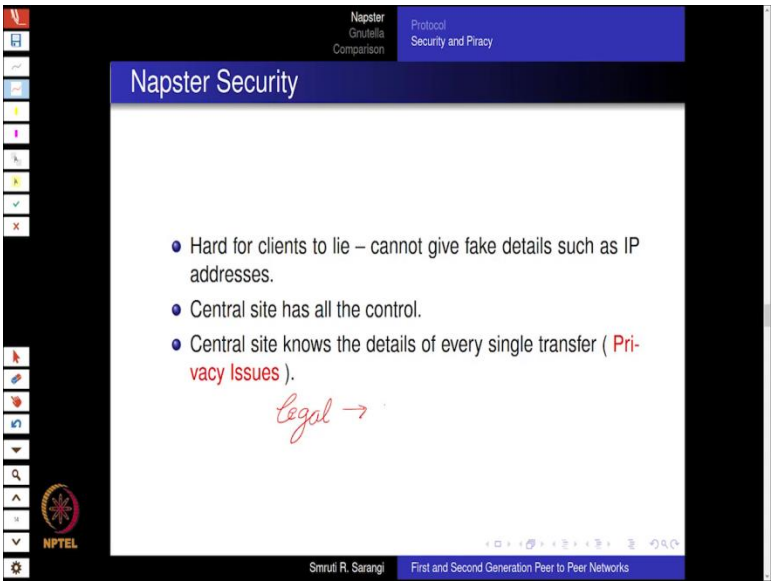


So now let us come to the upload instance. So when I am uploading the file, which means for them transmitting the file, I received, the ID received the file name, I set up the transfer, so I send all the bytes of the file, the upload is over and the connection is closed. So there is no great rocket science in this.

(Refer Slide Time: 17:19)

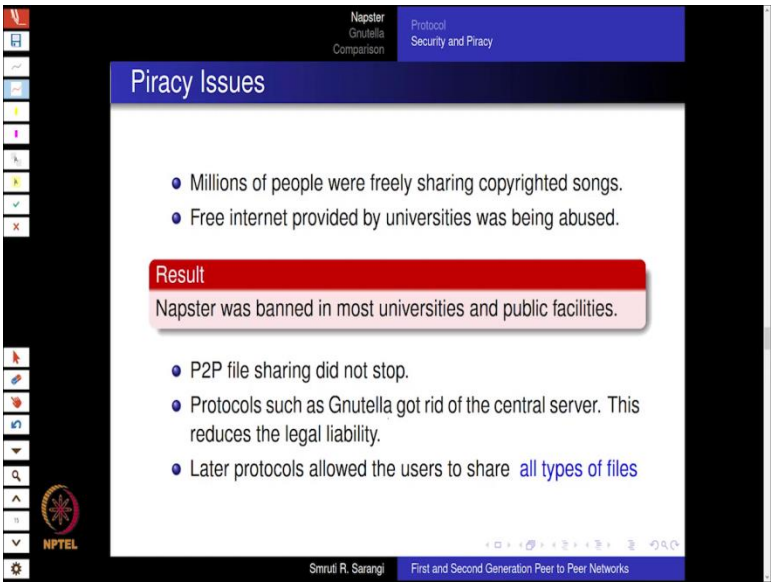


(Refer Slide Time: 17:20)



Security and piracy. So this is why Napster ultimately got a bad name? So the first is that it is hard for clients to lie. Okay, they cannot give a fake details such as an IP address, otherwise, who do you transfer it to? So the central site, in this case, the central site is the broker has all the control. It is aware of every single transfer. So later on, if let us say there is any legal issue, then you can pretty much catch the central site. And then that is trouble.

(Refer Slide Time: 17:47)



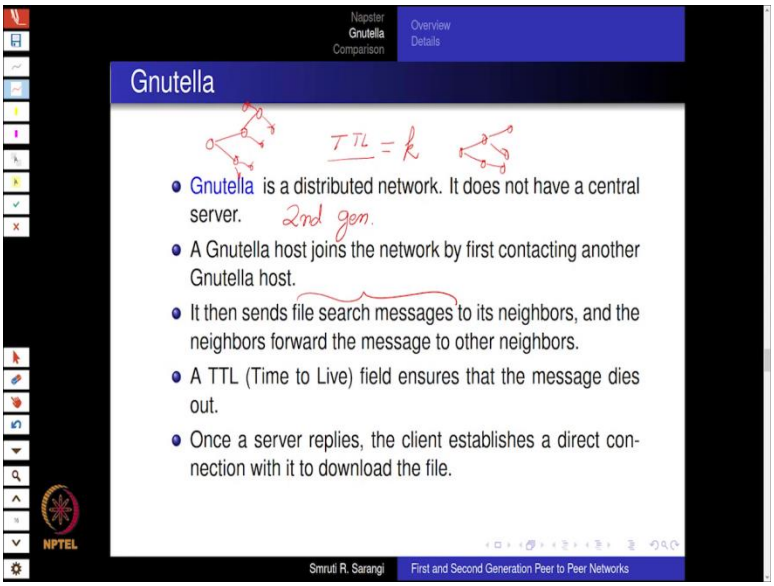
So, the problem that happened is that millions of people were freely sharing these copyrighted songs. And furthermore, universities were providing free internet facilities. And these facilities, frankly speaking, were being abused. So by the early 2000, most universities actually banned the

use of Napster. And pretty heavy penalties were put on students and other people who were using Napster to share files for which they did not have legal licenses. And also, this did cause a huge amount of revenue loss for the music industry. So the music industry did run after many people pretty hard. But peer to peer file sharing did not stop.

So the problem with Napster was the issue of the central server, which is aware of everything. So pretty much you get control of the central server, you find every single transfer and then people are suddenly legally liable. So Gnutella got rid of the idea, the central server. And also later protocols by Gnutella and so on, did not limit the sharing just to music files.

They brought in all kinds of files. By the time Gnutella came, which was roughly like, after 2000. All kinds of files could be shared with reduced legal liability. But still it did not make a transfer of a file for which the user did not have a valid license legal. Nevertheless, it was an improvement from a computer science sense at least.

(Refer Slide Time: 19:30)



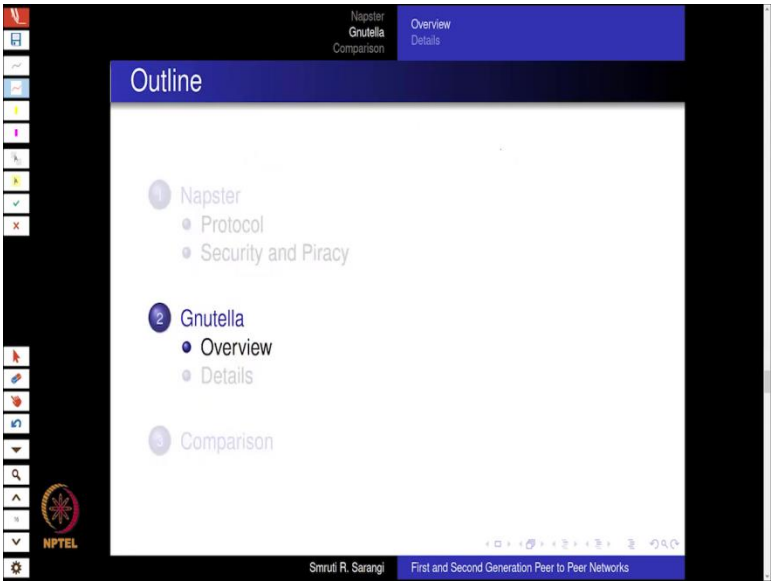
Now let us come to Gnutella. So Gnutella is another distributed network. We are calling it a second generation peer to peer network. It did remedy some of the flaws of Napster in the sense that it did not have a central server. So then in this case, a Gnutella host. So what we are calling a client or a peer there we will call the host over here. It will join the network first by contacting another Gnutella host. So you are assuming that it knows of some other Gnutella host you can contact it enjoy. So what it will do is that once it has joined, it will send a file Search Messages to its

neighbors and the neighbors will forward the messages to other neighbors. So in a sense, it will dynamically discover the network. So if you consider a graph like this, first it will send a message to its neighbors then to they will forward it to their neighbors, they will forward it to their neighbors and so on.

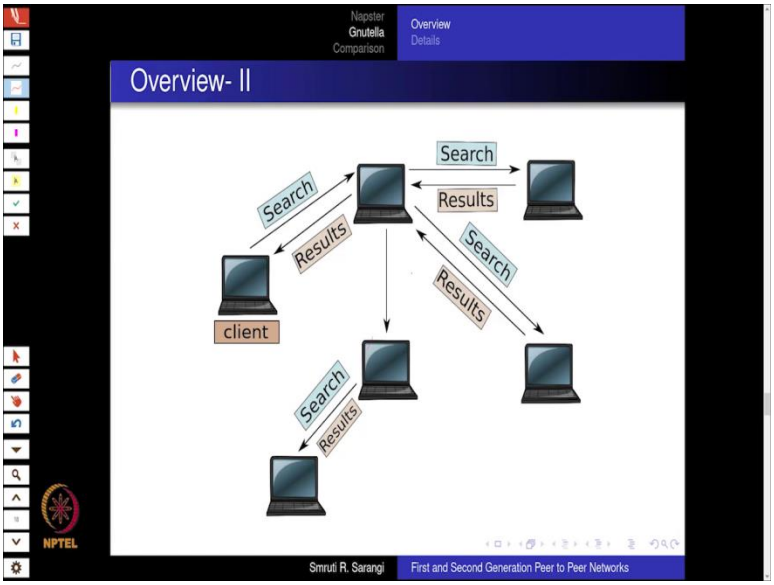
So it will know that in its neighborhood the names of all the files that are there. And of course, every message you don't want the entire internet to be swamped with messages. So every message has a message and a TTL field a time to live field that with every hop, the TTL was decremented. So ultimately, the message is died out. So if let us say that the TTL was equal to k, then this means that within a radius of K hops, you are going to have service of this, all of them all the leaf nodes are a radius of K hops over here, the message used to go to all of them, and the entire list of files used to come.

And this is how a node used to initialize itself? And once the server in this case used to reply, the client established a direct connection with it to download the file. And a server in this case is a remote period which to which a request used to go from the client saying that can I get this file from you? and then the remote be it used to say yes, if it had the file.

(Refer Slide Time: 21:52)

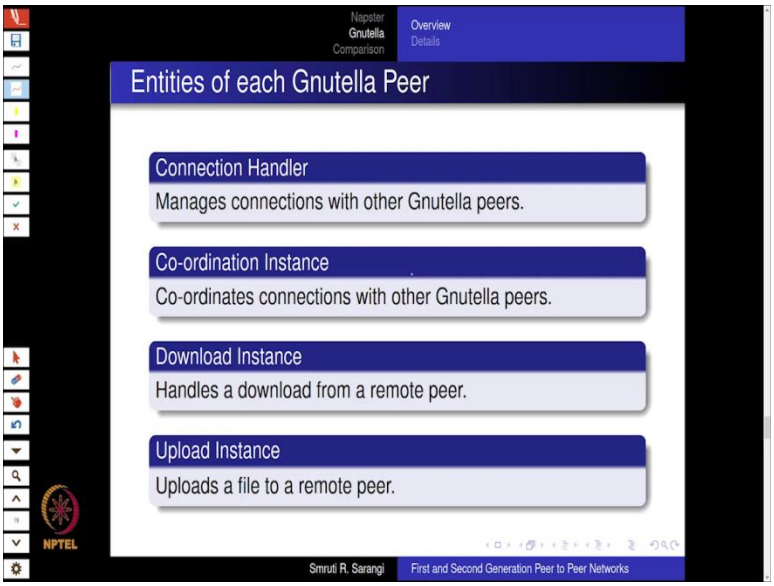


(Refer Slide Time: 21:53)



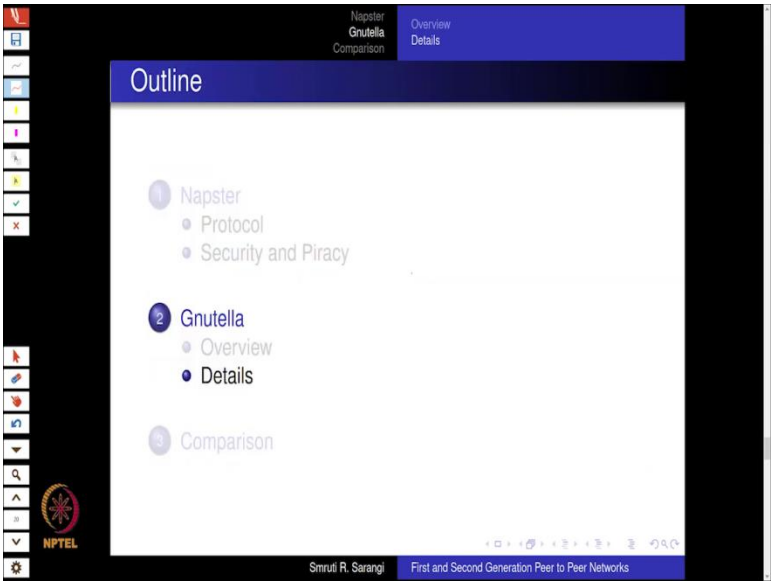
A quick overview. So the overview is like this, that Search Messages are basically if this is the client, then search messages are sent everywhere within the radius of K hops. And if anybody has the results, the results are sent back.

(Refer Slide Time: 22:15)

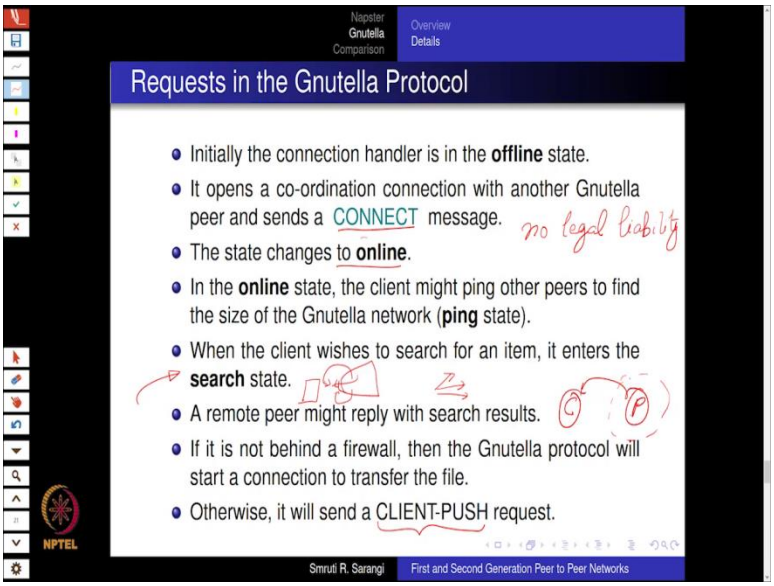


So again, every Gnutella peer would have four of these entities which would be a connection handler to manage connections with other Gnutella peers or coordination instance, which would coordinate connections with other Gnutella peers a download and upload instance as we have seen.

(Refer Slide Time: 22:34)



(Refer Slide Time: 22:35)



Now come to the details. Initially, the connection handler is in the offline state, it opens a coordination connection with another Gnutella appears. And since a connect message, so this would change the state to online. So let us assume that it somehow knows the idea of a Gnutella peer or there were some open source openly visible servers, which could at least connect you to Gnutella peers. So mind you, there was no legal liability there, because all that it was doing is that it was connecting you to another Gnutella peer. Nothing more than that it was not coordinating any transfer. So technically speaking, or legally speaking, connecting to Gnutella was not illegal.

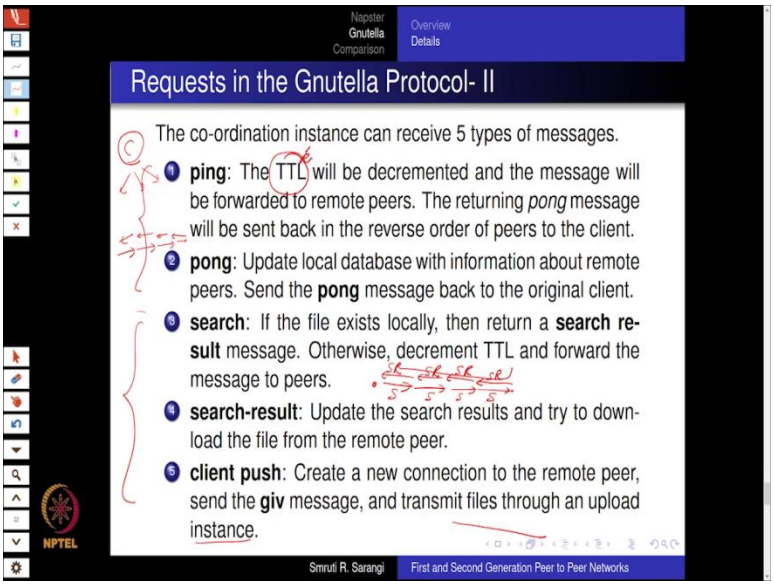
But transferring a file without licenses was illegal. So transferring illegal content was illegal. So that is the reason these lookups servers never participated in any information transfer as such.

In the online state, the client might ping other peers to find the size of the Gnutella network. Now, when it wished to search for something, it entered the search state. So what can happen is that initially, there is some amount of network discovery and when you discover the network within K hops, each one of them sends the list of files that it has or the list of files and things others have to it, they create some sort of a localized database use for searching that otherwise what you do is that you broadcast a search message, and of course every message in a time to live field which meant it ultimately died out.

And this was sent to a large part of the network. And once it was done, if any peer knew who has it, or if they themselves had it, they would return with the search results. And if the particular machine was not behind a firewall, and a Gnutella protocol would start a connection to transfer the file. Otherwise what you have seen in the previous case even with Napster where we had a remote client upload state, we had something very similar here because again this was made for college students, who unfortunately are known to transfer files illegally without licenses.

So Gnutella called in a similar mechanism, the client push request fair again, the remote peer, which in Gnutella terminology? we also refer to as a server even though Gnutella does not have a central server, but I am just going with the terminology that the Gnutella people use. So we, the remote peer in this case, would initiate a transfer in the client push model and send the file that is if the remote peer is behind some sort of a firewall or, or it is within some internal network.

(Refer Slide Time: 25:34)



So now let us come to the coordination instance, the coordination instance can broadly receive five types of messages. So the coordination instance is broadly looking at five kinds of messages. So the five kinds of messages that we are looking at here, the first two are pretty much net network discovery messages, the ping message and the pong message, ping pong, so a ping message, it is more like a new client joins.

And then it sends these ping messages to all over its neighborhood with a time to live field. If this is key, every time that we traverse one hop, the TTL field is decremented. And the message is forwarded to remote peers, then the returning. So when it reaches the end, what comes back is a pong message. So when the TTL becomes zero, what comes back to the original client is a Pong message, which is sent back in the reverse order of peers to the client.

So how does it work ping, ping, ping, ping, and again, back pong, pong, pong, and pong. So when the pong message comes back, we update the local database with information about all the remote peers. So remote peers can do the same to between themselves, read along the path of the pong message, and the pong and just keep on forwarding the pong messages towards the original client.

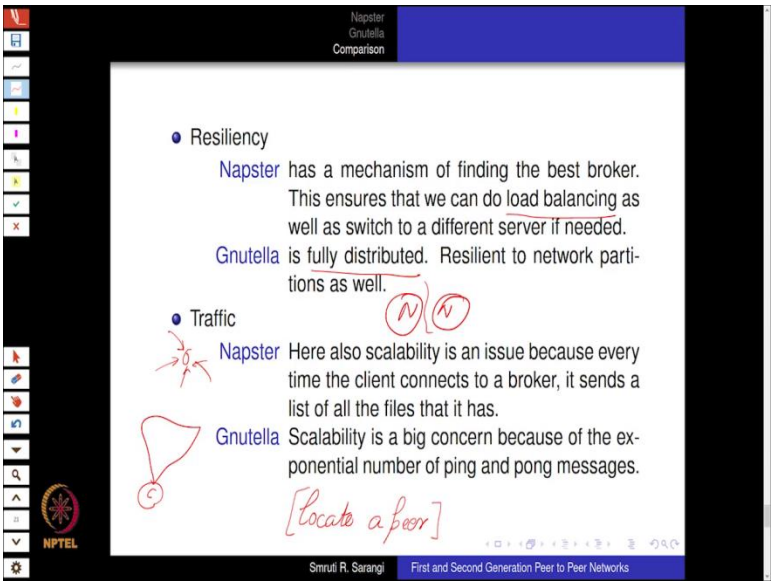
So this helps clients within a small neighborhood at least share messages between each other, and share information between each other regarding which song is there. So they maintain a localized database, because there is no central server. So then let us say that we know that a given song is there at some IP address. Then we can establish a connection directly. Now, let us assume that even after this network discovery, we want a song, which we don't know where it is? So what we

do is? if it does not even exist locally, you take it otherwise, we send a search message. So basically, if it exists locally, then of course, you return back a search result message. Otherwise, the search message is sent where the TTL field is decremented.

So the TTL field is always there to stop flooding in the network. And the message is forwarded to the peers. So what do we do? if we have one client, we just keep on sending search, search, search, search and so on. And the moment some information is found out, similar to ping and pong, what comes back is search results search result and soon. So the search results come back in the reverse order with the address of the remote peer, who has it? So we also refer to the remote peer as a server occasionally in Gnutella. But that is not all that common even though we have done so once in the past.

So here again, we have the same idea that if a connection can be established with the remote peer, it is fine. Otherwise, the remote peer realizes from the request that the original client cannot establish a connection with it. What it does is that it creates a new connection And transmits files through an upload instance. So this is similar to the client push is similar to the remote upload that we had in Napster, when the remote peer was behind a universities internal firewall. So in that case, what would happen is that the remote peer would establish a connection with a client and supply a copy of the file to it.

(Refer Slide Time: 29:34)



Resiliency wise, if I were to compare Napster has a mechanism of finding the best broker doing some centralized load balancing in the sense that it ensures no broker is overloaded. So this is all the advantages you get from a centralized network where you have more control. And also you have a centralized directory for lookup. It is not really a search based system. But here again, there is a massive amount of legal liability. Gnutella has reduced that to a large extent by being fully distributed. So the only job of lookups servers and Gnutella is to actually connect them to a host. And henceforth there is network discovery as well as searches within the neighborhood.

It is furthermore resilient to network partitions as well, because assume that the network gets partitioned, the sub network here does searches among itself and the sub network here does searches among itself. So Gnutella in that sense is extremely flexible in the way that it operates. So in terms of traffic, scalability is an issue because whenever you have a centralized server, it does see lots and lots of requests, the requests kind of swamp it. And because of that, what happens is that becomes a performance bottleneck.

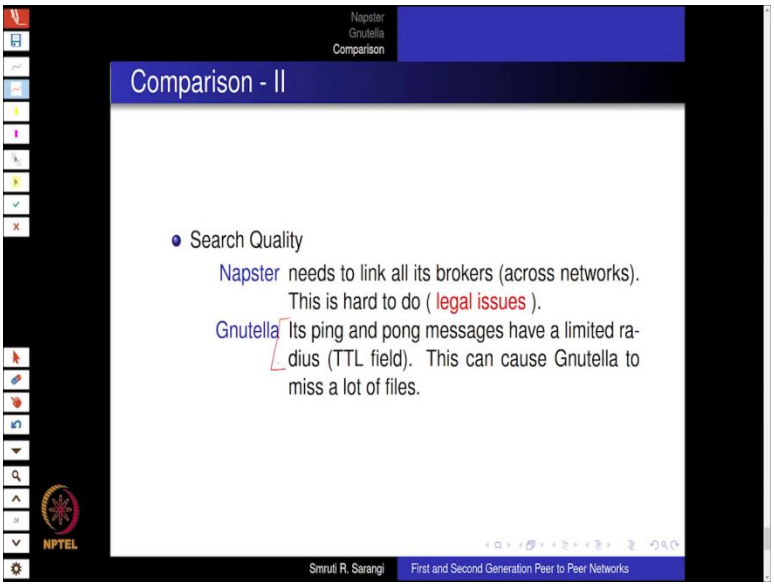
Gnutella also scalability is a big concern and bit in a different manner. So what happens is we send a lot of ping, pong search and search result messages. And they are all kind of broadcast within a radius of K starting from the client. And if you have many search messages that are being broadcast, then a local network can be full of such, search messages. Which was not happening in Napster, we just sent a single message to a centralized server and we got a request. But in Gnutella,

this issue will happen. Where we will say essentially be flooded by these discovery messages, just to find out who has what? So they have different kinds of issues.

Napster law, being less resilient, Gnutella being more resilient, Napster not being scalable, because you will have a lot of requests that the central server. Gnutella and not being scalable, because for discovering who has what, and also for the searches, a lot of search messages need to be broadcast throughout the network. So they have their own pluses and minuses. But clearly Gnutella, the legal liability is lower.

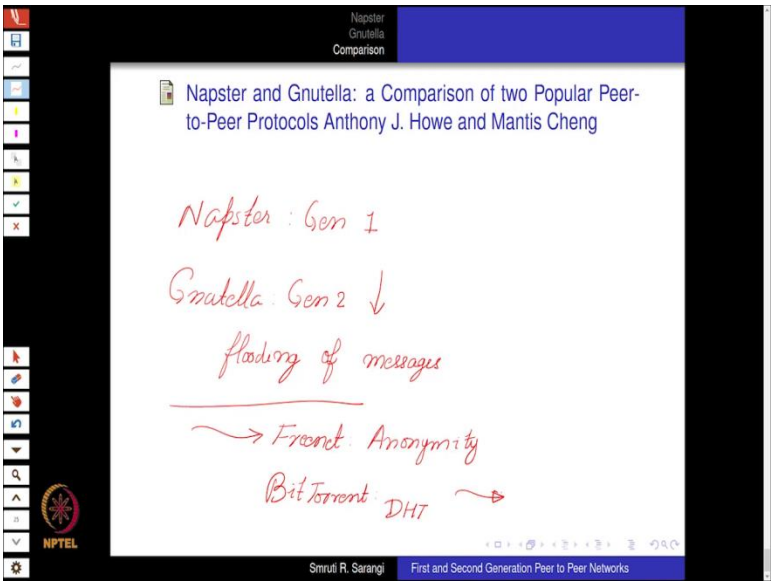
Hence, we are calling it a peer to peer network because in this case, the Gnutella server who the job of the server is only to locate a peer. If there is a new node that joining is only job is to locate a peer. Nobody maintains any usernames or passwords or anything. So this centralization is not there. It is truly a distributed system. And in the sense that Gnutella does not really say that you transfer illegal stuff, just in case two peers are doing their mutually responsible, nobody else.

(Refer Slide Time: 32:46)



In terms of search quality. Well, Napster needs to link all of its brokers across network, it needs to maintain a centralized database. This is hard to do, but you will get a better result if you can do it. And of course, there are legal issues. The ping and pong search and search result messages have a limited radius, this can cause Gnutella to miss a lot of files.

(Refer Slide Time: 33:13)



So given the fact that we have discussed Napster and Gnutella now, when Napster being let us call it gen one, Gnutella being gen two, we have seen the legal liability reduce. But this flooding of messages is clearly the issue with Gnutella. And furthermore, if let us say two peers are exchanging some data, they are clearly aware of each other's IP address. And all the nodes along the route are also aware of who is searching for what and who gave what search result? So in a sense, anonymously is not fully guaranteed.

So we will now see a few more technologies, where we can have free net, which was precursor to dark web and POD networks, where anonymous is pretty much guaranteed. So nobody knows what I am searching for and what I want. This is far more anonymous and secure. Again they are been used to do wrong things and bad things, but from a computer science, knowledge point of view you should know what is, and then will come into Bit Torrents.

So Bit Torrent, of course I have seen lots and lots of commercial application in a sense most of our model bit systems such as Amazon, Facebook, Google, etc. They keep technology the use is similar to what Bit Torrent uses and here there is flooding out messages. It uses a beautiful thing called a distributor hash table. So we will discuss that, this is pretty much the way that we are going. This is pretty much the direction in which we are going. So is this where the current lecture ends.