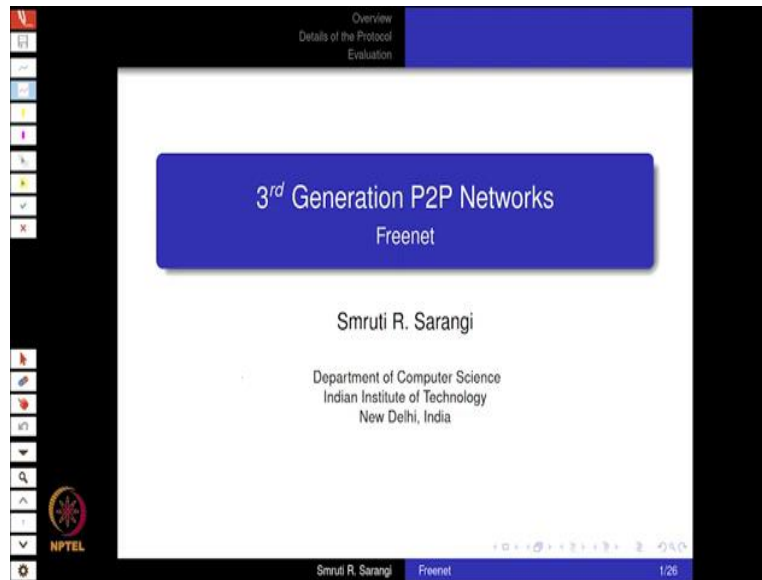


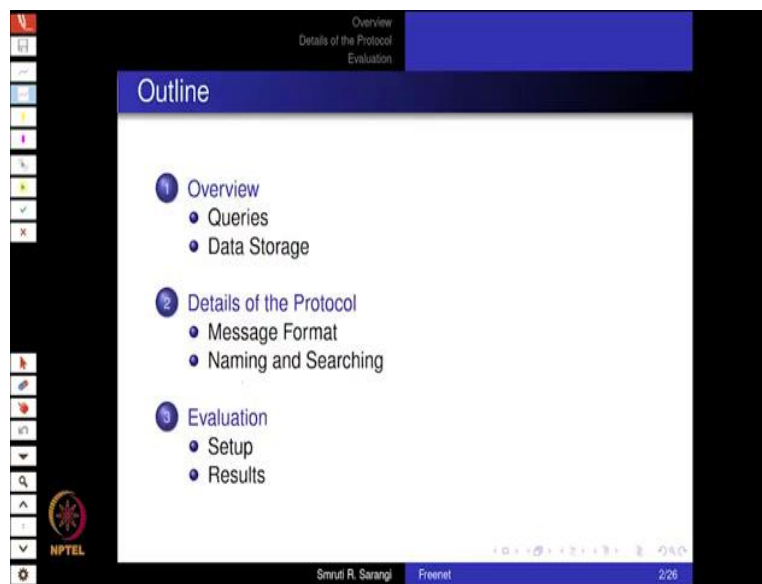
Advanced Distributed Systems
Professor Smruti R Sarangi
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
Lecture 04
Freenet: Distributed, anonymous storage

(Refer Slide Time: 0:25)



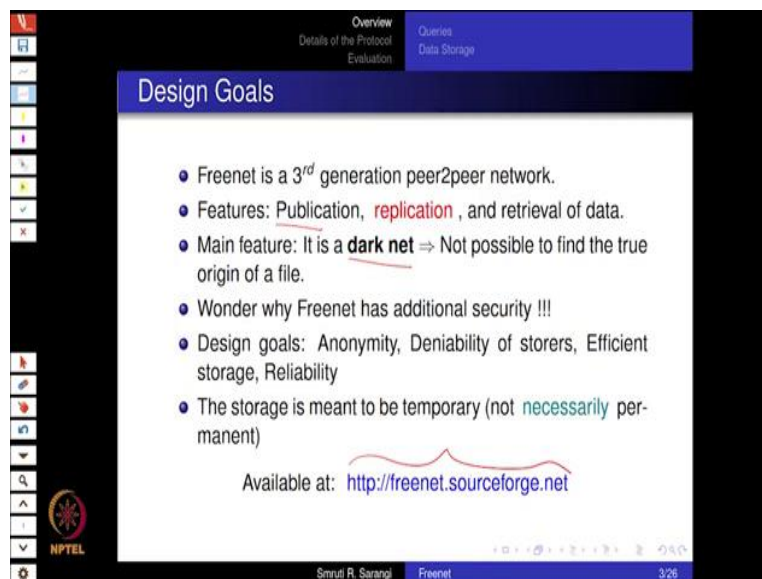
Welcome to the discussion on 3rd generation peer2peer networks. So we will discuss Freenet. So Freenet is also a precursor to the dark web. So we will look at this purely from an academic point of view, but there is, of course, a huge potential for such technologies to be misused and they are also being misused. But our discussion will only look at the academics part of it and we clearly do not support or endorse or promote or encourage any form of misuse. The discussion for this lecture is completely of an academic nature.

(Refer Slide Time: 0:56)



So we will look at queries and the data storage protocol in Freenet, details of the protocol and look at few of the evaluation results published in few of the original papers.

(Refer Slide Time: 1:12)



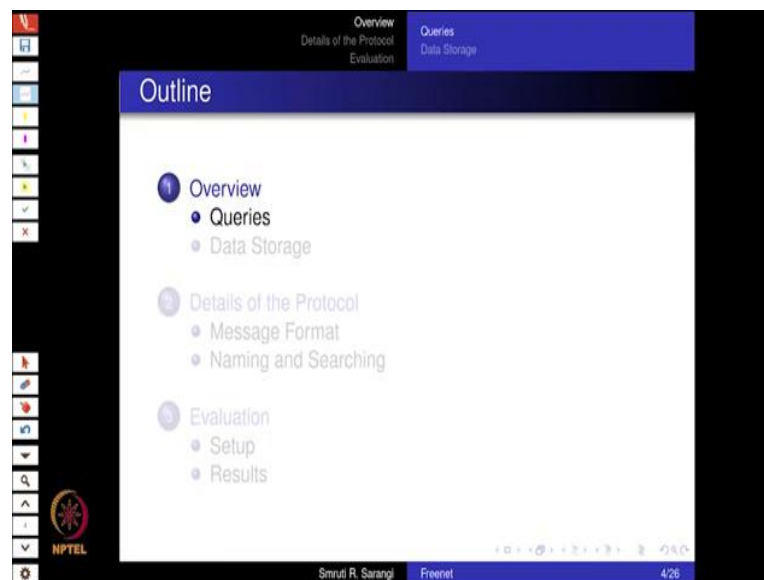
So Freenet is a 3rd generation peer2peer network. The main features are that we publish data, so different nodes can publish data, whatever it is, so unlike Nutella and Napster, Freenet is meant to share everything. There is replication in the sense that unlike other mechanisms, where the data is not replicated, you just say that where a piece of data is stored, so in this case the data is replicated across the nodes and data is retrieved.

So the main feature of Freenet is that it is a dark net, which means it is not possible to find the true origin of a file. So people will actually not know who is requesting for the file to a large extent and people will also not know who is the original provider of the file; so hence, it keeps both the provider as well as the requester of the file, it keeps both of them anonymous. This is very powerful. So there can be a lot of negative uses of this.

But it is important for us to understand this of how exactly it works. So the design goals are clearly in Freenet anonymity, the deniability of stores, so let us say one machine can say that I was just distributing the file, I was not creating it, I was not storing it, efficient storage and reliability, of course. So in this case the storage everywhere is meant to be temporary, it is not made to be permanent.

Because you need to understand that this protocol was completely made from a legal point of view where it can be denied that a certain machine or a certain user has actually created a file. So deniability was a very important part of this protocol, particularly from a legal angle, so a large part of this protocol is available online, it is available on the web, so you can download it from freenet.sourceforge.net. So last time we actually checked the link was active, but it has been a while now. So you can look it up.

(Refer Slide Time: 3:25)



Overview
Details of the Protocol
Evaluation

Queries
Data Storage

A Freenet Node

A Freenet Node

- Each node maintains its local data store
- Dynamic routing table: address of other nodes and the keys that they (might) hold *<Name> key*
- A node knows only about its immediate neighbours (not others)

Freenet Query

- Queries are sent to a node that can pass it to its neighbours.
- Each query has a TTL (time-to-live field) that is decremented at every hop
- A query has a pseudo-random identifier. This ensures that there are no cycles introduced while forwarding queries.

NPTEL

Smriti P. Sarangi Freenet 5/26

So overview, let us look at the queries. So a Freenet node looks like this. So each node maintains a local data store. So along with the local data store there is a routing table, which has the addresses of other nodes and the keys that they might hold. So we have the notion of a key similar to the one that we had in Pastry. So let us say we take the name of the file, we hash it and we create the key.

So the routing table is not the similar as it was in Pastry or as it is in a DHT, but the idea is similar where we have a set of keys and we have a set of nodes that may possibly hold the value associated with the key, in this case the contents of the file. So node as such is not aware of the entire network, because if it is aware of the entire network that is a security hazard. It is aware of its immediate neighbors and there may be a degree of trust between the node and its neighbors.

So a Freenet query is like this, it is similar to a Nutella query, where it is sent to a node that can pass the query on to its neighbors. Each query has a TTL field, a time-to-live field; that is decremented at every hop. So, this ensures that we do not flood the network with messages. Furthermore, every query has a pseudo-random identifier, so this ensures that there are no cycles while forwarding queries, otherwise what will happen is that we will forward a query and then you know it can go around in cycles.

So to ensure that there are no cycles, we have a pseudo-random identifier. So this basically means that a node does not forward the same query twice. If it sees that it has already forwarded this query it will not forward it once again.

(Refer Slide Time: 5:27)

Overview
Details of the Protocol
Evaluation

Queries
Data Storage

Steps in a Query

hash(name) → key

- 1 The *node* hashes the name of the file. This is the *key*
- 2 In its routing table, it looks up the key that is closest to the *key*, and passes the request to its *owner*.
- 3 If a node finds the file, it returns the contents, along with its address (saying that it is the *owner* of the data).
- 4 Otherwise, it finds the nearest *key* in its routing table, and forwards the request to that node.
- 5 If the request is ultimately successful, then the nodes on the way will:
 - 1 Cache the file
 - 2 Create an entry in their routing tables, and record the *original source*

NPTEL
Smruti R. Sarangi Freenet 6/26

So the steps in a query are like this that the node will hash the name of the file, it will create the key. So in its routing table it will look up the key that is closest, it will look up which key is closest to the key and it will pass the request to its owner. So it basically, this is like adding a little bit of Pastry to Nutella, so if you do not have the key at least you go to the owner of the closest key. If a node will find the file it returns the contents along with its address and it always claims that it is the owner of the data even though it may not be.

So this is, as we will see later, this is a way of essentially adding more noise, more entropy to the system, but always whenever a node forwards, it says that it is the owner of the data or it is implied. So this is essentially a way of not having a legal trail. Otherwise as we discuss, it finds the nearest key in its routing table and forwards the request to that node. So given that there is a TTL field, a time-to-live field, the messages will not percolate in the network forever.

But if the request is ultimately successful, then what will happen is similar to Nutella, the message will come back hop by hop. So each node on the way will cache the file. It will create an entry in its routing table and record the original source, whoever provided. But the original, so we do not know, this is not the, original source in the sense it is not the file that was the original owner, but it is one of the owners that is faking it and saying that look I am the owner.

It is essentially the node that is providing the file, but that actually may not be the original owner, it is the source, but the original owner can be somewhere over here. Maybe some other query happened and that is how this node got it, but as far as we are concerned the nodes on

the path, on this path will record that look this is the key and this node over here had actually supplied it.

(Refer Slide Time: 7:50)

Overview
Details of the Protocol
Evaluation

Queries
Data Storage

Steps in a Query - II

- 1 If a node **cannot** forward the request to another node:
 - Creates a cycle
 - Failure
- 2 Try the key with the second closest distance.
- 3 At every hop decrease the TTL till it reaches 0.
 - To reduce the network load, the TTL can be dynamically decreased.
 - Nodes can be decided to process which request **next** based on the TTL

Freenet = Gnutella + P2P + anonymity

NPTEL

Smruti R. Sarangi Freenet 7/26

So the main idea is that to kind of take care of legal issues, the more that you can confuse the better it is. So the important point over here is why are we introducing this? So the reason we are introducing this is, of course, yes, BitTorrent and Freenet have been used to do a lot of bad in the world, but as computer science researchers, if let us say you want to stop it or regulate it or control it or investigate a crime that has been committed on these networks, you need to understand how it works such that you can do good to overwhelm the bad.

So that is the main reason that this is being discussed. So, now let us come back to our original point. So what we discussed is that anytime a node finds it again the search result goes back along the original path, and of course, each one of them caches a copy of the file and it creates an entry in the routing table, which points to over here and this may not be the original source though, because this might have cached it for some other query.

So if we cannot forward the request because it is creating a cycle or something we just declare failure. Then what we can do is that the protocol does not have to give up. You can try the key with the second closest distance instead of the closest distance or you can keep trying. At every hop we decrease the TTL by one, the time-to-live field by one and to reduce the network load the TTL can also be dynamically decreased.

If let us say you are finding that there are too many messages in the network, so let us say this is a network and there are too many messages, then you can dynamically reduce it. So nodes can decide to process which request next based on the TTL. So needless to say, what we are essentially doing is we are adding, we can have a simple equation over here that Freenet is essentially equal to Nutella in the sense that we are sending it to neighboring nodes.

So recall that we had a lecture on Nutella, so if you are not, if you are new to this lecture watch the Nutella lecture first, plus Pastry in the sense that we have the notion of keys and routing tables. We also had a lecture on Pastry, look that up. And furthermore there is a degree of anonymity in the sense that we are trying to confuse people with who the actual source is and clearly whoever is the original requester that information is not being propagated.

So other than my immediate neighbor nobody else knows that actually I requested for the file. So this is why I had originally said that there has to be a degree of trust between a node and its immediate neighbors, because this immediate neighbors are aware that at least a node may not have requested for it, but it forwarded a request.

So here is also the thing, let us say between a node, I am, this node is me, and this node is my immediate neighbor, see, if let us say I forward a request, the immediate neighbor does not actually know for sure that whether I requested it or not. Maybe somebody else requested it and I am forwarding it on that somebody else's behalf, so the immediate neighbor will not know, but at least it will know that I am participating in the protocol, so that degree of anonymity is there.

(Refer Slide Time: 11:31)

The slide is titled "Operation of the System" and is part of a presentation on "Overview: Details of the Protocol Evaluation" and "Queries: Data Storage".

(a) Routing Table

key	address	content (?)
key	address	content (?)

(b) Example: Multiple messages being sent

The network diagram shows six nodes: a, b, c, d, e, and f. Node a is connected to b (edge 1) and b to a (edge 12). Node b is connected to c (edge 2), c to b (edge 3), d (edge 6), d to b (edge 7), e (edge 4), e to b (edge 11), and f (edge 10). Node d is connected to e (edge 5) and e to d (edge 8). Node e is connected to f (edge 9). The diagram illustrates a circular loop: a → b → c → e → d → b → a.

So the routing table is simple, it is basically the key, the address and the content, and we end up sending multiple messages, so we can look at this, a sends a message to b, b sends it to c, c says look I do not have it, and then it sends a message to e, e sends it to d, and then d sends it to b, but then we realize that there is a circular loop, so it comes back. Then d says look I do not have it, it goes to, from e it goes to f, maybe f has it, so then it again walks back. And, so that is how we send multiple messages in this protocol.

(Refer Slide Time: 12:17)

The slide is titled "Search Quality" and is part of a presentation on "Overview: Details of the Protocol Evaluation" and "Queries: Data Storage".

- Gradually over time, information disseminates.
- Nodes start aggregating files with similar keys (numerically close)
- Popular data gets replicated at a large number of nodes.
- Routing tables gradually get bigger.
 - New entries get created all the time.
 - Nodes can discover more of the network without disclosing their identity.

The diagram shows a central node labeled "file" with arrows pointing to a group of nodes labeled "like servers".

So what happens is the search quality gradually over time information disseminates. So gradually over time what happens is that the information that which node has which key

gradually disseminates. The nodes start aggregating files with similar keys because we go with this numerical closeness, so with similar keys gradually what will happen, if you keep on running the protocol is that for a single node, data with similar keys will start getting aggregated.

Popular data, especially where data is popular, it will start getting aggregated at a large number of nodes in the network. Routing tables will gradually get bigger, because new entries are getting created all the time and nodes are also discussing, discovering more of the network without actually revealing identities. So with this what is happening is that there was some degree of anonymity in Nutella already, but there is a little bit of additional anonymity here.

The primary reason being that also we are, for every real file we are creating multiple fake owners, so for every real file we are creating multiple fake owners. Multiple ones, where pretty much whichever path it travels they will cache a copy and they will all claim that they are owners. So given the fact that let us say in a large network there are hundreds of nodes that claim to be the owner you do not know which one to catch.

And all of them are just participating in the protocol. So this is similar I remember to one such experience in my high school where we all did some mischief. Well, actually a few of us did some mischief, but essentially the entire class said that each one of them did it, so the teacher did not know whom to punish, because it would not have been a wise thing, a right thing to punish the entire class because 90% of the class was innocent.

But he did not know which 10% was involved, so this is something very similar where we all create fake owners and nobody knows who is supplying and also nobody knows who is requesting, because since all the nodes are participating in the protocol you do not know if a neighboring node is actually requesting for the file or it is forwarding somebody else's request. So both, provider as well as the requester, are to a very large extent anonymous.

(Refer Slide Time: 14:50)

The slide shows a navigation menu with three main sections:

- 1 Overview
 - Queries
 - Data Storage
- 2 Details of the Protocol
 - Message Format
 - Naming and Searching
- 3 Evaluation
 - Setup
 - Results

At the bottom, it says 'Srnuti R. Sarangi Freenet 10/26'.

The slide titled 'Storing Data' contains the following list:

- User first creates a file key.
- She then sends an insert message to its own node (file, key, TTL)
- If the node finds the key in its routing table, it returns the contents of the file.
- Otherwise, finds the closest key in its routing table and forwards the insert message to it.
- If that insert causes a hash collision, the node passes data back to the upstream requester.
 - Cache the file locally, and create a routing table entry (in response to insert).

At the bottom, it says 'Srnuti R. Sarangi Freenet 11/26'.

Now let us look at data storage. So the user will first create a file key. She will then insert, send an insert message to its own node with the file, the key and the TTL. So, if the node finds the key is already there in its routing table it will return the contents of the file, otherwise it will find the closest key in its routing table and forward the insert message to it. So, here is the fun part, the fun part is that any node, which is the original provided actually does not store it.

It uses the same routing mechanism to send it to a place which has the closest keys. Even at the beginning itself if you think about it in the network, if let us say I want to store some piece of data, I do not necessarily have to have a copy of it. I just send it far in the network such that it is stored in wherever the closest key is, such that it ultimately gets stored somewhere else and that is far away from me.

If there is a hash collision, of course, the node is passed back to the upstream requester and the file is cached locally and we create a routing table entry in response to the insert. The hash collision we do not forward it, but the, it is sort of the request bounces, so whichever node had forwarded it, it stores it. So the important point should be rather clear over here, the import of this entire conversation.

That if this question has not naturally arisen in your minds, it now should that let us say that there is a file which will not get heavily shared, then it will be possible to kind of find out who was the original inserter or the file into the network, if there is such a term. What this means is who brought this file into the network for the first time? But to thwart that what we do is that whenever we insert a file into the network we do not keep it with the node that is actually inserting it.

We essentially let the file propagate through the network towards the node that has the closest key and it is stored pretty much at the node that has the closest key. So let u's say there is a node and then none of its neighbors have any closer key, so then at that point it will store it itself, otherwise it will keep on forwarding. So this will ensure that the file actually propagates quite a bit is far away from the original provider, original owner.

And it gets stored over there, sort of by the proximity of its keys. So that later on pretty much the legal liability is gone, it is not possible to identify who had introduced this file into the network.

(Refer Slide Time: 17:56)

The slide is titled "No Hash Collisions" and contains the following text:

- If the TTL field becomes 0, without any collisions, then it means that the insert is successful. *Closest key*
k-hops
- Let the original requester know.
- Each node on the path adds the file and key to its routing table. It also caches the file.
- The original node also adds the file and key to its store, and routing table.
- How do you beat security ??? *key, ...*

Below the first bullet point, there is a cartoon character of a boy with orange hair. To his right, there are two more bullet points:

- Nodes along the way lie about the data source
- Add their own address or some other node's address

The slide also features a navigation sidebar on the left with various icons and a footer at the bottom with the text "Smriti R. Sarangi Freenet 12/26".

So then assume that there are no hash collisions, so the TTL field becomes 0, without any collisions, then it means that the insert is successful and which essentially means that it is kind of, let us say the TTL field was k and it just kept going on and on and on without any hash collisions, we can have a successful insert. So, there are essentially two ways of terminating the process of insertion, one is an approach that we outlined on the previous slide.

That we stop the search when we are there at the node, which has the closest keys and no neighbor has closer keys, so let us call it closest key. So this is a recent addition to such Freenet kind of protocols and the other is that we just send it k hops away, just send it k hops away in the direction of the closest keys and then when the TTL field becomes 0, you store it. So regardless of whatever method is chosen ultimately if the node is inserted the original requester is known by sort of back propagating the message.

Each node on the path will add the file and the key to its routing table, it will also cache the file and the original node also adds the file and key to its store. It may do it, but just to beat security it may not also do it. Now, how do you beat security? So nodes along the way will lie about the data source, in any case they do not know about the original data source, because they are just forwarding the data.

So they will lie about it and they will say I am the owner, everybody will say I am the owner and furthermore, what they can do is that in their key, what they can do to add further anonymity is that with the key we have an address. So they can put their own address and they

can fake that I am the owner or they can say, or let us say if they do not want to have the content, so in many cases for storage reasons, we may not want to have the content of the file.

We just want to have an address and redirect a request over there, in that case they can add their neighbors address or who they are forwarding it to, so they can slightly randomize that as well, so that will introduce more entropy into the system and make it even harder to find out who is the actual owner.

(Refer Slide Time: 20:28)

The image shows a presentation slide titled "Advantage of this Mechanism". The slide is displayed within a software interface that includes a navigation bar at the top with options like "Overview", "Details of the Protocol", "Evaluation", "Queries", and "Data Storage". The slide content is as follows:

Advantage of this Mechanism

Advantages

- Newly inserted files are placed on nodes with similar keys.
- Information about newly inserted files can quickly be disseminated.
- An intruder will find it difficult to **deliberately** introduce hash collisions.

The footer of the slide includes the NPTEL logo, the name "Srinu R. Sarangi", the word "Freerit", and the slide number "13/26".

Advantages of this mechanism - Newly inserted files are placed on nodes with similar keys, so this aids the search process. Information about newly inserted files can quickly be disseminated and furthermore, if a file is being introduced into the network, inserted into the network, it is actually being added to a node that is possibly far away from the original owner. An intruder will find it difficult to deliberately introduce hash collisions. It might be possible that you want to deliberately introduce hash collisions, but there is a mechanism to detect that and backtrack as we have seen.

(Refer Slide Time: 21:11)

The screenshot shows a presentation slide titled "Data Management" with a blue header. The slide content includes:

- The routing table and data stores are **finite** structures. They follow a LRU (least recently used) replacement **policy**.
- This ensures that old files get **deleted** from the system.
- **Legal issues :**
 - A data store can **deny** the knowledge of the files it has.
 - Take the hash key, and encrypt the contents of the files with it.
 - Any node can always **decrypt** a file, if it knows the key
 - However, it will not be able to find out what the original key was.
 - Example: A data store can always say that it didn't know that it had that many pop songs. It **didn't know** which file was a pop song.

Handwritten annotations in red ink include: "Owner" with an arrow pointing to "encrypt", "encrypt" written in a circle, and a small box containing "E".

At the bottom of the slide, the text "Smruti R. Sarangi Freenet 14/26" is visible.

So, now let us come to the data management aspect of it. So the routing table and data stores are essentially finite structures. So they have a limited amount of storage. So they can be made to follow a LRU or least recently used replacement policy, where once a file gets very old it is not used, it is deleted, some legal issues. So up till now we have said that look the owner of the file that is inserting the file into the system for the first time or the consumer of data, both of them can kind of deny. But what about the nodes that are forwarding the information?

So nodes that are forwarding all of this, they can deny the knowledge of the files that they have by saying that look I was just forwarding, I did not take a look at it. It may cut ice with the law enforcement or it may not. The other idea can be to take the hash key and encrypt the contents of the files with it. Any node if it knows the key, it can always decrypt the file. However, it will not be able to find out what the original key was.

For example, a data store can always say that it did not know that it had many pop songs, because it did not know which file was a pop song. Well, fair enough, if we are dealing with encrypted data, then we can essentially pass around encrypted data all over the network as long as the final consumer has an idea of what the key was. So, what is the idea over here? The idea over here is while inserting the file, the original owner encrypts the data.

And it is assumed that all the consumers have some idea of what the key was and then it is kind of encrypted and the encrypted versions are transferred all over the network. So then deniability does exist in the sense that a node can say look I did not know what I was forwarding because it was encrypted and I had no way of seeing what it was. The only catch here is that the final

consumer needs to know the key and that cannot happen via Freenet. That needs some other mechanism of disseminating the key.

(Refer Slide Time: 23:39)

Overview
Details of the Protocol
Evaluation

Message Format
Naming and Searching

Outline

- 1 Overview
 - Queries
 - Data Storage
- 2 Details of the Protocol
 - Message Format
 - Naming and Searching
- 3 Evaluation
 - Setup
 - Results

NPTEL Smruti R. Sarangi Freenet 15/26

Overview
Details of the Protocol
Evaluation

Message Format
Naming and Searching

Freenet Message

- Packet oriented, self-contained messages: TCP or UDP
- Every transaction (search or insert) has a **unique ID**
- All messages contain: **64-bit transaction ID**, TTL counter, and a **depth** field.
 - An attacker can guess the identities of nodes by scanning the TTL value.
 - To thwart this: With a finite probability when the TTL field reaches 1, keep propagating the request to other nodes.
 - Have another field called **depth** that is incremented at each hop. It starts with a (> 0) random value.
 - Before the **destination** sends the message back to the **source**, set the TTL = depth. This ensures that the message will not die before reaching the requester.

Handwritten notes:
A → B
k = n
TTL = k
TTL = 15 > k

NPTEL Smruti R. Sarangi Freenet 16/26

So this issue is there with encryption. This is a serious issue, but as I said in many cases just participating in Freenet is a crime by itself even though they may not be fully aware of what exactly they are keeping and transmitting and transferring, but nevertheless just participation can be deemed criminal based on the jurisdiction and the laws of the region. So the message format, well, so we can either have TCP messages or UDP messages.

So TCP is a reliable form of transmission, UDP is an unreliable form, where we can drop packets. Every transaction has a unique ID. So all messages will contain a 64 bit transaction

ID, a TTL, a time-to-live counter, and a depth field. So what the attacker can do? Well, the attacker is basically maybe a member of a Law Enforcement Agency, it can guess the identities of nodes by scanning the TTL value.

So it will look at the TTL value and figure out where the original request came from, if let us say, always the TTL value is set to 10 and let us say I am seeing the TTL value as 5, I can say that the original source was maybe 5 hops away. To thwart this with a finite probability, when the TTL field reaches 1, it is kind of keep propagating the request to other nodes or will dynamically add some random noise to the TTL field.

So this will thwart such attacks probabilistically. Another approach is to have a new field called depth, which we have not discussed up till now, that is incremented at each hop. It starts with a random value. So before the destination sends the message back to the source, we set the TTL equal to the depth, so this ensures that the message will not die, before reaching the requester. So let me explain the context of the depth field once again.

So the idea over here is that along with TTL we have another field called depth, which is incremented at each hop, but it starts from a random value, that is very important. It starts from a random value. So now when we reach the destination, the message has to come back and it will use the same TTL mechanism to come back, but the point over here is that if the $TTL < k$, so in this case what will happen is that if, then the message will die, if it is exactly.

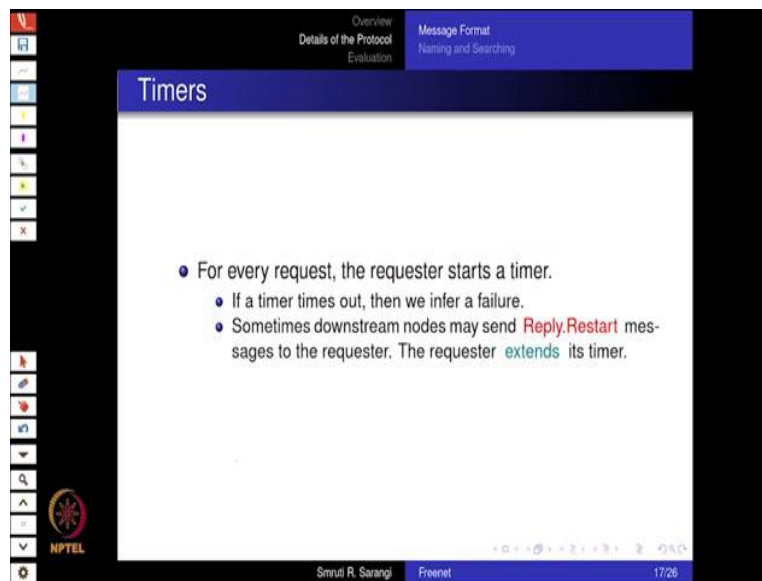
So, let us say that the destination and the source are k hops away and from the destination, which is the provider of the file, the message is coming back, so then also we will have the TTL mechanism to ensure that all messages ultimately are removed from the system. So let us say this is k hops and let us assume that from the destination to the source, when it comes back we send $TTL = k$. Then if let us say one of the nodes along this path, let us say belongs to the police, it will have an exact idea of how far the source original, requesting source is from it.

And this is something that we want to avoid. So you want to avoid the fact that even if there is any intruder in the network, it will not be able to get enough information about the source. So what we instead do is that we use the depth field which has been initialized randomly. As we go from the source to the destination or the requester to the provider, we keep incrementing it. So let us say $k = 10$ and the depth was initialized to 5, so this will become 15 over here.

So when you are coming back, we actually sent $TTL = 15$, so this is guaranteed to be let us say $\geq k$, so the message will come back, it will reach the source, it will not die in the middle, it will not be removed from the network in the middle and furthermore, if there is any intruder, it will not get an exact idea of how far the source is. Primarily because of the randomized value of depth. So this is an important concept.

Kindly go over this what I said and also go over this description in the paper, but the key idea is come back to how far the destination and source are, say if there k hops away TTL cannot be equal to k , otherwise it will give an intruder an exact idea of how far the source is, so this is revealing a lot of information. It cannot be less than k , because then the message will die in the network. It has to be greater than equal to k and how much it is greater than equal to has to be a random value, so that will confuse the intruder, that is exactly what we are trying to do over here.

(Refer Slide Time: 28:51)

The image shows a presentation slide titled "Timers" within a software interface. The interface includes a top navigation bar with "Overview", "Details of the Protocol", and "Evaluation" on the left, and "Message Format" and "Naming and Searching" on the right. A vertical toolbar on the left side contains various navigation icons. The slide content consists of three bullet points: "For every request, the requester starts a timer.", "If a timer times out, then we infer a failure.", and "Sometimes downstream nodes may send Reply.Restart messages to the requester. The requester extends its timer." The bottom of the slide features the NPTEL logo, the name "Smiti R. Sarangi", the word "Freenet", and the number "17/26".

So, another method is for timers, for every request, the requester starts a timer. Say the timer times out, we can infer a failure, so this is basically an absolute time, so let us say we are not able to get a reply, we infer a failure, so downstream nodes may send replied or restart message to the requester, if there is a problem, if there is congestion, so then the timer will be extended. The requester will extend its time. So this is also another mechanism for ensuring some amount of reliability and robustness in the overall message transmission scheme.

(Refer Slide Time: 29:30)

The screenshot shows a presentation slide titled "Successful Request". The slide content includes a list of five bullet points. The first bullet point is "If the request is successful, the remote node will send the `Send.Data` message." The second bullet point is "It can send the id of the data source (or possibly *fake* it)." The third bullet point is "If TTL reaches 0, it sends a `Reply.NotFound` message." The fourth bullet point is "If there are no more paths left and ($TTL \neq 0$), then a `Request.Continue` message is sent." The fifth bullet point is "The requester can send the request to other nodes in its routing table." There is a handwritten red arrow pointing from the fifth bullet point to the text "*second or third closest key*". The slide also features a navigation sidebar on the left, a top navigation bar with "Overview", "Details of the Protocol", "Evaluation", and "Message Format", and a footer with "Srinati P. Sarangi", "Freenet", and "18/26".

- If the request is successful, the remote node will send the `Send.Data` message.
- It can send the id of the data source (or possibly *fake* it).
- If TTL reaches 0, it sends a `Reply.NotFound` message.
- If there are no more paths left and ($TTL \neq 0$), then a `Request.Continue` message is sent.
- The requester can send the request to other nodes in its routing table. *→ second or third closest key*

So if the request is successful the remote node will send the Send dot Data message. It can send the idea of the data source or possibly fake it, so it is important to send more information to confuse and fake. If the TTL reaches 0, it will send a Reply dot Not Found message. If there are no more paths, but the TTL is not 0, it will send a Request dot Continue message and if that is found, the requester can send the message to other nodes in the routing table which are not possibly the closest key, but as we have seen maybe the second closest key or the third closest key or something like that. so on and so forth.

(Refer Slide Time: 30:27)

The slide shows an 'Outline' section with three main items: 1. Overview (Queries, Data Storage), 2. Details of the Protocol (Message Format, Naming and Searching), and 3. Evaluation (Setup, Results). The 'Details of the Protocol' section is currently selected. The slide footer includes the NPTEL logo, the presenter's name 'Smruti R. Sarangi', the topic 'Freenet', and the slide number '19/26'.

The slide is titled 'Naming, Searching, and Security' and contains a section 'Organizing Files' with several bullet points. Handwritten notes in red and blue ink are present: 'legal issues' with an arrow pointing to the first bullet, 'directories and bookmark lists (dark web)' with an arrow pointing to the second bullet, and 'Smruti ranjan Sarangi' written next to the third bullet. The text 'Goes against our design goals: anonymity' is written in red. The slide footer includes the NPTEL logo, the presenter's name 'Smruti R. Sarangi', the topic 'Freenet', and the slide number '20/26'.

So naming and searching - So organizing files, it might be a good idea to have a directory of keys, like files containing similar data, but the main idea is we never did not want to centralize, so recall that after Napster our main aim over here was not to centralize data, was not to have a central server, was not to have a repository and it was mainly to avoid legal issues. Primarily it was to avoid legal issues that arise from doing so.

And the reason is that you immediately get a list of servers that will cause issues and whoever maintains this will be in trouble. If there are no issues we can have directories or bookmark lists, so it is common that the dark web, which is built on Freenet like technologies has such directories in countries that I will not name, who are beyond the legal and kind of judicial purview of many other countries, like ours, so it is not possible to trace them.

And so this is done but, of course, you need the kind of the stability of a country to do it, otherwise it cannot be in the same legal jurisdiction. The person who is maintaining the directory will be in trouble. Should we have a search engine like Google to do it? Well, it goes against our design goal anonymity. So the key solution is we will use a lot of indirect files containing metadata all over the network.

So then we will have a dictionary with list of keywords and keys of the files. The key idea over here is that let us say you are searching for, let us say my name. So if you are searching for my name, so let us say for my first name there will be a lot of places, so actually my name is not Smruti, it is Smruti Ranjan; that is the typical way that people are named with such ultra-long names in my home state.

But well, what to do, so this is my big long name. So you can have one set of directories that maybe have a list of servers that have this name, the first name, and then you can have another set of servers which have links to nodes that have my last name. So then they can be separate queries and then you can take the intersection of them to find servers that have links to my entire name. So this also is possible, different combinations.

So clearly there is a trade-off between visibility and efficiency. And also it is possible that within the network you might have intruders, which will tamper with the contents of the file and they will say that look this is the file, so to ensure this does not have, we can hash the contents, and the hashed values can be there with other nodes.

(Refer Slide Time: 33:43)

The screenshot shows a presentation slide titled "Security" with a blue header. The slide content includes a bulleted list of points about sender anonymity and pre-routing. Handwritten red annotations include "depth randomization" next to "Messages between pairs of nodes can also be encrypted (against eavesdroppers)", "decides" next to "The requester decides the routing path", "RSA-based encryption" next to "Encrypt a message with a succession of public keys", and "key" next to "No idea of the requested key". A small diagram of three nodes (N) is also present. The slide is part of a presentation with a navigation bar on the left and a footer at the bottom containing "Smruti R. Sarangi", "Freenet", and "21/26".

Overview
Details of the Protocol
Evaluation
Message Format
Naming and Searching

Security

- Sender **anonymity** is preserved because:
 - A node can never say if the node that is requesting a file is the original requester, or is merely forwarding a request.
 - Messages between pairs of nodes can also be encrypted (against eavesdroppers) *depth randomization*
- Pre-routing
 - The requester **decides** the routing path to a destination if it has detailed routing tables.
 - Encrypt a message with a succession of public keys (for all the nodes on the path) *(RSA-based encryption)*
 - A node will have no **idea** regarding the sender. It will only know the id of the next hop.
 - No idea of the requested **key**. *(N) (N) (N)*

NPTEL
Smruti R. Sarangi | Freenet | 21/26

So little bit of security. Sender anonymity is preserved because a node can never say that if the node that is requesting a file is original requester or merely forwarding a request. We have discussed this. Messages between pairs of nodes, well, TTL is one source of information, but we eliminated that source with this depth, field and randomization.

There could be eavesdroppers, so we can encrypt messages, but subject to the fact that the keys are being shared using some other mechanism, the requester may decide the routing path to a destination, if it has detailed routing tables. It furthermore can encrypt a message with a succession of public keys for all the nodes on the path. So what is the idea? So, of course, here the assumption is that you know what public private key or RSA based encryption is.

If you do not know what it is then kindly take a look at it and restart the video from this point. So the idea over here is that as we are traversing the network it is assumed that every node has a unique public key. So what we do is we keep encrypting a message with a succession of public keys along the path and when it comes back, we keep on decrypting it with the private keys. So there are advantages of such mechanisms.

I am not going into the details. There are tons and tons and tons of mechanisms of adding more security to Freenet, so I will not discuss this in great detail. I will just discuss a few of the solutions that do exist. So in general a node does not have any idea regarding the sender, because it only has 1 hop visibility in this network and it also does not have a clear-cut idea of what is the key that is being requested.

But the main issue with encrypting the keys in this fashion is that the protocol as we have explained, where you take a key and you do a lookup will not work, and you are also assuming, whenever there is some encryption that there is some other way in which the keys have been shared. So that is very important. So I am not discussing this in great detail because there is a lot of work already in this area and this is technically beyond the scope of this current discussion.

(Refer Slide Time: 36:27)

Overview
Details of the Protocol
Evaluation

Setup
Results

Outline

- 1 Overview
 - Queries
 - Data Storage
- 2 Details of the Protocol
 - Message Format
 - Naming and Searching
- 3 Evaluation
 - Setup
 - Results

NPTEL
Smruti R. Sarangi | Freenet | 22/26

Overview
Details of the Protocol
Evaluation

Setup
Results

Evaluation Setup

Setup

- Network has between 500 to 900 nodes.
- 40 items in each node.
- Routing table size: 50 addresses
- Network topology: Chain

0 → 0 → 0 → 0 → 0

NPTEL
Smruti R. Sarangi | Freenet | 23/26

So little bit about the evaluation results. If you see the paper they will look at a network between 500 to 900 nodes with 40 items in each node. The size of the routing table is limited to 50 addresses and it is a linear network, this is not the best network to test, but nevertheless it is a chain based linear network.

(Refer Slide Time: 36:48)

Overview Details of the Protocol Evaluation Setup Results

Outline

- 1 Overview
 - Queries
 - Data Storage
- 2 Details of the Protocol
 - Message Format
 - Naming and Searching
- 3 Evaluation
 - Setup
 - Results

Smruti R. Sarangi Freenet 24/26

Successful Requests(%) vs #Queries

- The number of queries were varied from 50 to 1200
- For 500 nodes, the percentage of successful requests rose quickly from 20% (at 50 queries) to 100% (at 300 queries).
- With 600 to 900 nodes, the percentages started at roughly 10%.
- They reached close to 100% for more than 400-500 queries.
- More are the nodes, lesser is the percentage of successful requests.

Smruti R. Sarangi Freenet 25/26

So the results that they show is that for a query rate that was varied from 50 to 1200, for 500 nodes the percentage of successful requests rose quickly from 20 to 100. So that is because as there are more and more messages in the network, information disseminates and that too it disseminates rather quickly. With 600 to 900 nodes things started slow at 10 percent, but after 400 - 500 queries they reach 100 percent.

So pretty much all such networks information does get disseminated quite rapidly, and ultimately once things stabilize most of the search results are successful. More are the nodes lesser is the percentage of successful requests, higher is the warm-up time, but the ultimate results are kind of good and acceptable, for at least what you get.

(Refer Slide Time: 37:50)

#Hops vs #Queries

- Experiments were conducted with 500, 600, 700, 800 and 900 nodes.
- The #hops reduced quadratically from roughly 50 (20 queries) to 10 (> 600 queries).
- More are the nodes, more are the hops.

20 → 600

Number of hops versus queries. So experiments were conducted with 500 to 900 nodes. The number of hops reduced quadratically from roughly 50 to 10. That is mainly because as you have more nodes you have more connections, so that is why the number of hops also reduced. So, I just slightly retract the statement. I did not mean to say more nodes imply less hops, what I said is more queries imply less hops. So, I stand corrected.

So what I am trying to say is as we increase the number of queries, so let us say as we go from, let us say 20 queries to 600 queries, the information gets disseminated all over the network, so you see a quadratic reduction in the number of hops, which is also what you would expect and also what you would expect is as the number of nodes increase the average number of hops would increase.

So these are not fundamentally earth shattering results, so you would expect a reduction, in this case it is a super linear reduction in the number of hops traversed versus the number of queries. And so given that we stand broadly speaking over here, we are now in a position, so this was by the way the last slide.

(Refer Slide Time: 39:23)

Clarke, Ian, et al. "Freenet: A distributed anonymous information storage and retrieval system." Designing Privacy Enhancing Technologies. Springer Berlin Heidelberg, 2001.

freenet

- 1-hop visibility
- Request denial
- Sender denial (fake owner)
- Insertion → remote node
- TTL (depth, randomization)

And this is the paper that describes the original Freenet paper; that led to a lot of work in this area, but the key points that actually anonymize Freenet are essentially one hop visibility. So this is called, let us call it request denial, which basically means that you will never reveal whether you are the original requester or you are just forwarding somebody else's request. Then the other is also sender denial, where you are, you will basically say that look either I am not the original owner but I will fake, I will have the notion of fake owners.

I will say that look I am the owner even though you may not be, and furthermore, in the routing tables if you are not actually storing the value, which means you are not storing the file you will point to one of these fake owners, then you have the issue that at the time of insertion, the node that is inserting the file will not keep a copy, but it will actually send it far away in the network. It will send it to a remote node with possibly the closest key.

So that will cache a copy of the file including some of the nodes in its neighborhood via of which the message reached it. And the last is that the TTL field can reveal some amount of information, so we will also incorporate a depth field, which will add a degree of randomization. So these are pretty much all the steps, which give Freenet a degree of anonymity, which is not there in other systems such as Pastry and Nutella and so.

And of course, we do not have any centralized server or any kind of a centralized directory, which has a big key and server list. So this is where Freenet ends. So Freenet has, of course, matured significantly, so there is a huge dark net and you have the Tor browser and so on, so you can take a look at the security issues involved in such technologies and how law

enforcement agencies try to keep a tab on these networks and find offenders. Thank you very much.