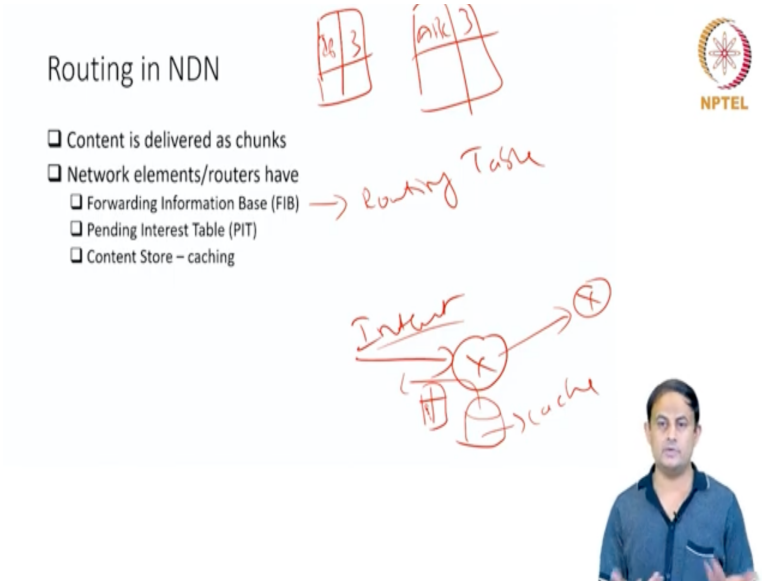
**Advanced Computer Networks**
**Professor Doctor Neminath Hubballi**
**Department of Computer Science Engineering**
**Indian Institute of Technology, Indore**
**Lecture 67**
**Named Data Networking Part 2**

(Refer Slide Time: 0:16)




So, now let us look at how the routing happens in the NDN. So NDN routing operation actually uses the name-based routing that is consistent with the ICN philosophy, and the routers themselves have maintained three different data structures or the elements inside them. So one is called as the Forwarding Information Base, so this is exactly similar to that of the routing table. So whatever the IP address table is there, and the forwarding table is there, so that is the Forwarding Information Based on the TCP /P network.

Similarly the named tables are there inside the NDN architecture that is one piece of information. So that will help you find where to get the content. So in the TCP/IP communication given a destination IP address, I am finding the IP address given, on which port number I want to pass on, and what is there in the NDN architecture is given a name, let us say ABC and where to go to get that content. So IP addresses are simply replaced with names including the prefix. So here, in this case example that I took, there is no prefix but it can be a prefix name. So that is the difference. This is called FIB information.

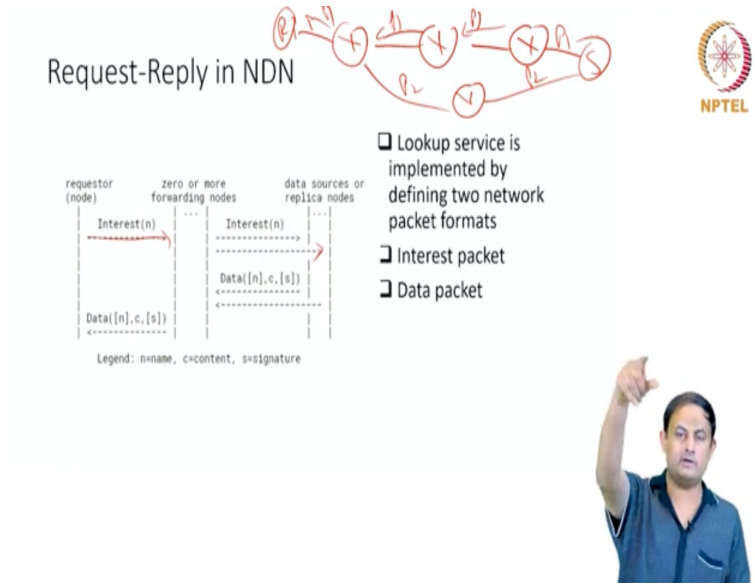
And then the second component is something called as the Pending Interest Table. So whenever an Interest Packet comes to a router, there are two operations that happen; one is you consult what name chunk it is requesting for and check whether the content that is requested or the named content that is requested is there inside your own cache and if it is available and you supply this, and if it is not available you forward this interest packet to the next hop router along the path to where the content is available.

So while this is getting forwarded, the interest packet gets forwarded, it remembers that I have forwarded a request for the content so and so, maybe the content ABC and it is pending. So whenever the response comes then who actually made that request, against whose request I sent this interest packet in the first place, and then when the data packet comes from the other end you match it and then you forward it to that particular host in the reverse direction. So this mapping is done using the Pending Interest Table. And in this process whenever the new content is actually received by this router it may decide to cache the content.

So if the space is available, it can cache it; if the space is not available still, it can cache it by replacing some of the existing content, and you can also make the decision to cache the content or not to cache the content based on how popular the content is. I can use some notion of the popularity so whether this content has been used in the past or not based on that I can decide to cache the content or not to cache the content. So these are the possible things.

So you have a Forwarding Information Base that is similar to that of a routing table which will tell you where to get the content and the Pending Interest Table will have the information who has requested for the data in the first place and when the data comes from the other end, to whom it has to be forwarded, that is that information is kept in the Pending Interest Table, and the content store simply has the cached content in this particular place.

(Refer Slide Time: 04:47)



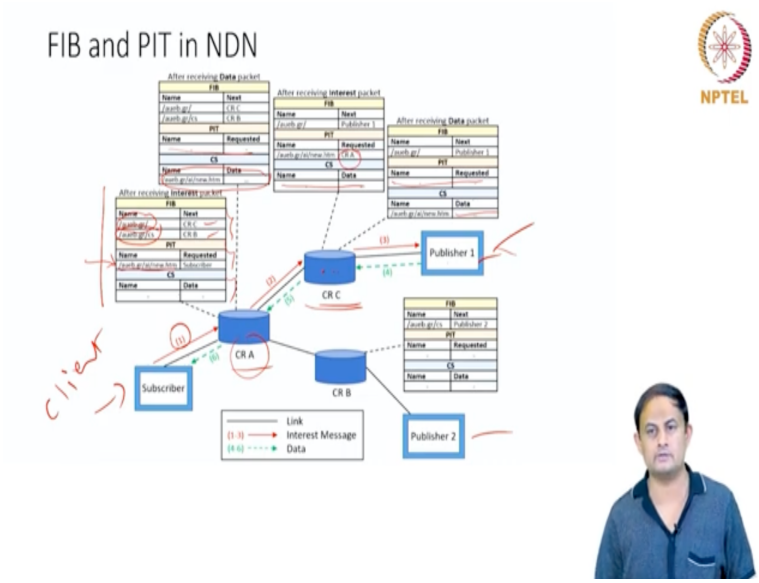

So this is how the request reply movement actually looks like in the NDN architecture. So the interest packet, as I said, is flowing from the requester node to the data source. So this data source may be the original data source or a node or a replica node or a router which is having the content, and in the reverse direction, the data itself has three things; one is the name that is requested in the interest packet and the content itself, and then the signature and this actually flows in the reverse direction. So you can think of this communication as symmetric in nature; symmetric in the sense that, let us say here is the recipient, and these are the three routers, and here is the sender, and if there is an alternative path available from this sender to this receiver; in the IP model what can happen is the packet number 1 can take this route and through these set of the routers the packet can reach the receiver. The packet number 2 is independent of the packet 1 and it can flow through the second direction.

So this can happen. So every packet is carrying an IP address, and they independently flow in the network, and there is the intended recipient. But unlike that NDN communication is symmetric in nature on per request basis, meaning wherever the request, through which interface, through which sequence of the routers, the interest packet has gone, in the reverse direction through the same sequence or visiting the same sequence of the routers, the content is actually delivered to the intended recipient. So the path taken in the forward direction and the path taken in the reverse

direction are just opposite to each other, this communication is called as symmetric communication. And the NDN mostly follows the symmetric communication paradigm.

But in the TCP/IP network, we might have asymmetric communication as well. So although on a larger basis majority of the routes might take still the symmetric sequence of the path, but occasionally you might have some packets taking a different route. This might be simply for the load balancing purpose or any other purpose or any other reason. But in NDN communication also different requests can take different routes, but on a per request basis it is going to be symmetric. Wherever the request has gone, the data chunk replied to that is actually coming in exactly the opposite direction.

(Refer Slide Time: 08:04)




So this is how the routing and the three things PIT, content store, and the FIB information in the NDN architecture look like. So every router maintains all three of them, and there are two data producers here, these are publisher number 1, the second one is publisher number 2. And here is the client who is actually making the request which wants some content to be delivered to it. So, as usual, because NDN is a receiver-driven architecture, the client is the one who initiates the communication to get some content, and it sends a request, the sequence, packet number 1 is an interest packet going from the client to the first hop router and when this request comes so the

router number 1, here in this case is CR A is the one which looks at its own Forwarding Information Base and then its own content store.

First is whether this content is available in my content store. So at this point of time, this is the structure of all the three entities, from here to here is the FIB, from here to here is the PIT, and from here to here is the content store. So when the interest packet arrives at this router, it looks that here is the content store there is nothing there, it is blank, so that means that the content is not there in its local cache, it cannot serve, and then it then looks at its own FIB information. So assuming the content queried is for something which has a prefix aueb.gr. So it looks upon which of the next hop that I need to forward this interest packet to, and the FIB entry says that you need to forward it to the CR C, content store number C, and this particular router. So you forward that request, at the same time it is actually logs that request, the details into its own PIT table.

So this is the exact content that is asked for aueb.gr/ai/new.html. So that is actually produced by this producer. So there is one client that has asked for so and so content, and the FIB information said that I need to send it to the next hop or this particular router, and then send it to that router, it sent and then created one entry. And again, the next hop router, this router will also do a checkup, it also looks at its own content store here, after receiving the request, it looks at this content store is empty; it also does not have the content, and then it creates its own entry in the PIT and looks at the FIB where it has to forward. So it says that anything that has the prefix aueb.gr, you should forward it to publisher number 1, which is actually giving you the data, you make an entry in the PIT, send it, and then you get the data.

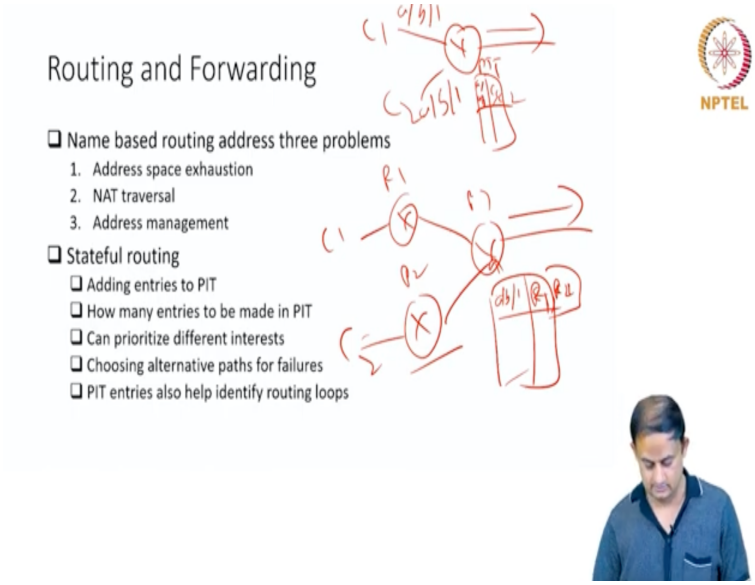
So publisher is the one that gives you the data, this dotted two lines, the green colored lines will indicate the data in the reverse direction. Once the data comes to this router and it again checks okay, this is the content name and so and so prefix, and I have a PIT entry already with that particular name and who actually queried it I need to send it to the CR A from which router, I got it from this router, I need to send it to him and he sends to this router, he arranges the entry PIT entry after the content is received from the publisher and then it will forward it to this router and arranges the PIT entry because the content has been already delivered.

In this process, it can cache the content and update the same content that I am forwarding now is also available in my cache so it is stored and then it is forwarding the content. And now once this

router also receives that content, it will also look at its own PIT table which looks like this, and it arranges the PIT entry, and it might also decide to cache the content and create an entry in its cache store and finally after caching this content, CR A might deliver the content to the subscriber. So this is how actually it is going to operate.

So, in summary, FIB information will tell you which is the next hop where you have to forward the interest packet to get the content, and then the PIT table tells you who has made the request and whose request is yet to be served, only those entries are there in the PIT table and content store is actually having the names of all the contents and also the data itself in its memory; so that is what the structure is. So, again the next hop information is found out by using a longest prefix matching mechanism, so here, in this case, the content that is requested is aueb.gr/ai/new.html but there is also one more entry with aueb.gr/cs, so that is a different prefix. so if anything having the aueb.gr matching with only this one then you need to forward it to C otherwise you need to forward it to B that is what these two entries actually indicate. So just like the IP prefixes are the longest prefix matches here also, it is the longest prefix match using the names, so that is what the NDN routing architecture would look like, so what the routers actually have and keep updating the information.

(Refer Slide Time: 14:43)



So this kind of the name based routing mechanism of the NDN or, in general, any ICN architecture actually helps in many ways, so here are a few of the prominent ones. It is also because the names can be arbitrarily sized, and it helps us to get rid of or address the problem of the address space exhaustion. We all know IPv4 addresses are limited in number, and we have exhausted all the IPv4 addresses and now IPv6 version has come and there is no guarantee that IPv6 also at least if not now, 50 years down the line or 100 years down the line, it might be exhausted. So then, how do I do that? So if you have an arbitrarily sized name and then that problem is solved, I no longer have fixed-sized addresses or fixed-sized names, so any name I can give so that problem can be inherently addressed in the NDN architecture.

And because of the IP address restriction, and scarcity of IP address availability, we came up with this NAT technique where the public IP address is mapped to a private IP address to establish communication between a device that is behind a NAT firewall with an outside device. So that is also not required here because every entity is independently named; everybody is pairing with an independent name, there is no need for having the NAT traversal. So in effect, these two things will make address management much simpler; address management here, in this case, is name management.

And unlike the TCP/IP model where IP uses a stateless communication mechanism so here in this case, the routers are becoming stateful because they are keeping the entries in the PIT table. So, how does it help is, if two different clients, $C_1$ and $C_2$, request for the same content, let us say a/b/1, and he also request for a/b/1. So if the $C_1$ is the first one that has queried for a/b/ 1 and subsequently $C_2$ queries, the router notices that there is one entry in its own PIT table with the same name a/b/1, and it has been requested by the $C_1$, and now there is no point in sending the second request from the $C_2$ to the same data producer or the desktop router. So what it does is whatever the content that is a/b /1 is requested by the client $C_1$, and now when the second request comes, it will add an entry with the name $C_2$.

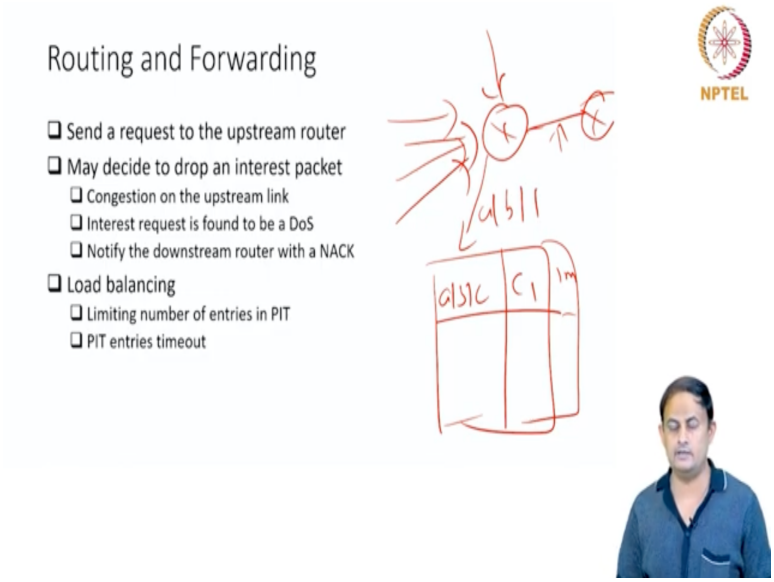
When a/b/1 comes back from the other end, I need to deliver it to both $C_1$ and $C_2$ that is what it actually means. So if there is already a Pending Interest Table entry for a particular named chunk and a new client or the new subscriber makes the request for the same content, then the interest packet is not forwarded to the next hop, instead of that, it is dropped there itself but a PIT entry with a name with the previous hop information here in this case the subscriber is added to the

PIT table. So this is happening even if the requester is not the client or subscriber but a different router then also the same mechanism would work.

So, for example, if I have two different routers client $C_1$ is connected to the first one, and client $C_2$ is connected to this one, and if the PIT entry is both of them, assuming $C_1$ and $C_2$ request for the same entry, the same content a/b/1 and assuming $C_1$ request first and then it is already there and this router has already sent that interest packet to the next hop and through this intermediate router the $C_2$'s request will also reach this router then also it will just create an entry with the name $C_2$. Here, in this case, if you name these routers as $R_1$, $R_2$, $R_3$, this would be $R_1$, this would be R2, so I need to send it to the router number $R_1$ and also to router number $R_2$; that is what it means.

So that is how the PIT table is actually maintained, and it will help minimize the number of the interest or requests going to the content serving node. So, unlike in the IP model, what was happening so if multiple client requests, all of them are independently getting the content, multiple times the content is getting transmitted, that is actually minimized here in the NDN architecture. So often because the content is stored in many, many different places inside the network, the same content might be available at two different routers and a particular router when it is forwarding the request for one particular data chunk so it can still do the load balancing, in the sense, if let us say router number 1 and 2 have the same content and if the same content is requested which is available at both places sometimes I might send it to the router number 1, and sometimes I might send it to the router number 2, and you can still do the load balancing that is still possible inside the NDN architecture as well. And of course, if you forward the request in one direction, if the connection fails, then if multiple other sources also have the same content, then you actually try one path, if you do not get the content, then you can go and check in or explore the alternative paths as well so that is possible inside the NDN architecture.

(Refer Slide Time: 21:46)




So there are some mechanisms to also handle the inherent problems that are available in the TCP/IP communication, which is congestion, because of the congestion, the routers might decide not to send the interest packet in a particular direction but it can actually send it in an alternative path or it might not be able to handle the interest request because of some other reason.

So let us say I have a router here, and the interest packet is coming from some source or some other router to this particular router, if there is congestion, assuming this is the only next hop path from here, I can get that content; let us say take the same example a/b /1, and this is the only interface on which I can send it and if I know that this particular router and this link is heavily utilized and it is congested, and this request that is coming from the other particular router or the client, may be dropped right here without creating an entry. So, PIT entry is created only when the interest packet is actually forwarded to the next hop. So if you are not forwarding the interest packet then there is no PIT entry in the table or in some other scenario for security reasons.

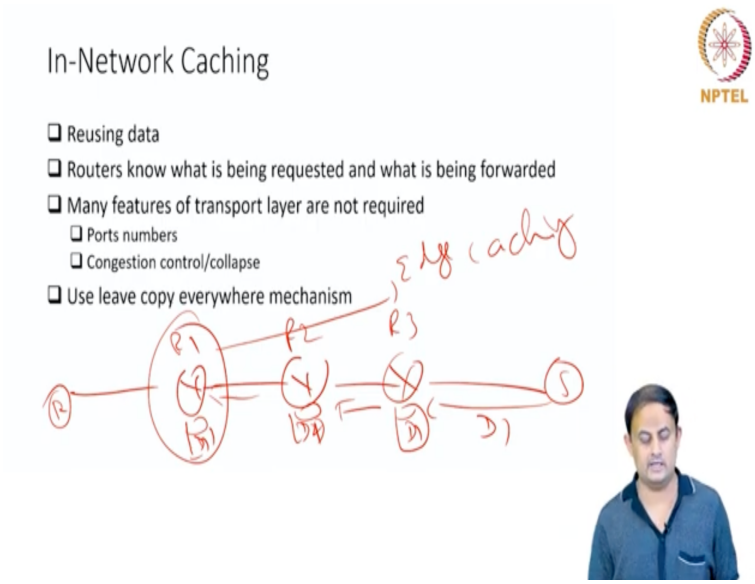
So in the previous lectures, also we discussed one, somebody can generate too many numbers of the named content and try to publish in the network or overwhelm the network. The other one is somebody can maliciously keep sending the request to a particular router, I want a/b/1, I want a/b/2, I want a/b/3, something like that, or to the same in content many many different requests are sent so that this guy actually spends a lot of time in servicing those requests that are the case of the Denial of Service. So if the router learns that a particular request coming from a particular source or a particular port number or actually a malicious setup of the request, they are intended

to kill or burden the router itself then it might drop the request without forwarding it to the next hop. So whenever it drops the content, it might also tell in the reverse direction the requested content is not available with me; you can try elsewhere, or it is not possible to serve your request at this point of time.

So if you use this kind of load balancing or selective forwarding mechanism for whatever the reasons that I just mentioned. So that will help you to manage the PIT table entries. So assume this case if you do not discard the interest packets, if some malicious guy is sending too many number of the request without any purpose then because he is asking for so many number of the data chunks, you keep on sending the next hop, they may not even be available. So then your number of entries in the PIT table would grow substantially. So that is not desirable. So you may not be able to accommodate the new genuine request coming from some other sources. So one way is you drop it without creating the entries in the PIT table, or you send the negative acknowledgement saying that so and so content is not available or you put another way of dealing with that is you put a timer for every PIT entry.

So a/b/c is requested by client 1, and the time for this one is, let us say, one minute. So within one minute, I should get the content from the other end. So to whom I sent a request from that guy, I need to get back the content within one minute. If it is not coming within one minute then you purge the entries in the from your PIT table. So this will help you to keep the PIT table at a manageable level otherwise, the number of entries will go substantially, and that will become a burden for the operation.

(Refer Slide Time: 26:34)




So that is about the routing and the forwarding, and the next feature of the architecture is the caching. How does the caching actually happen in the NDN networks? So caching, as usual, helps reuse the data, and it will also help you to serve the content quickly thereby reducing the latency of the transmission. So that can only happen if you make the router's content aware. So meaning, what kind of content is being requested by which client, and what is available in their content store. And this caching works without the notion of the port numbers, and there is no danger of congestion collapse, and congestion control mechanisms are not required because I am no longer establishing the connection between the two parties. It is just that if I send one request against that, some data would pop up at my door.

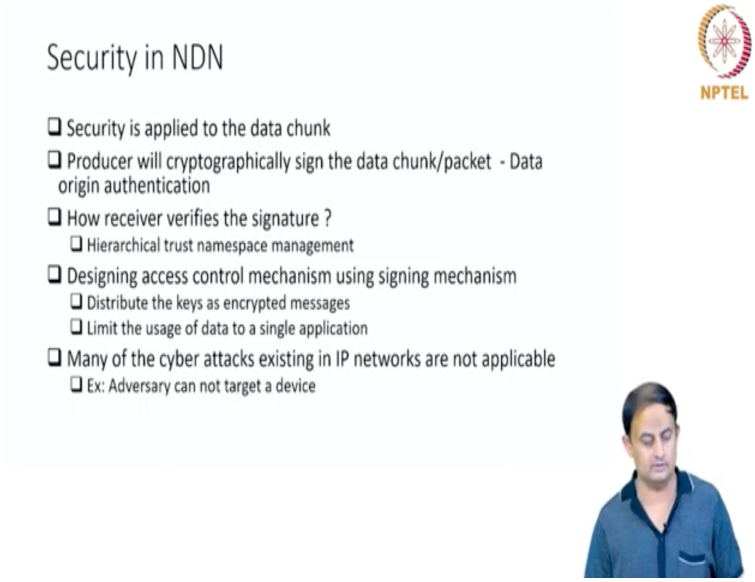
So by using that simple mechanism of forwarding the request and response, the NDN proposes to use a very simple caching technique which is something called the leave copy everywhere. What it means is, if the requester is here or the receiver is here and the sender is here, and the intermediate routers are these many, so if some data comes let us say the data is $D_1$, so $D_1$ is available in the content store of this router may be the $R_3$ and $D_1$ will be stored in the content store of the $R_3$. So $D_1$ is the data that is the $R_2$ and the $D_1$ would be stored at the content store of the $R_1$ as well. So, through what all the routers, the content is moving in all of those router's content store, you put this copy of the $D_1$. So that is called as the leave copy everywhere.

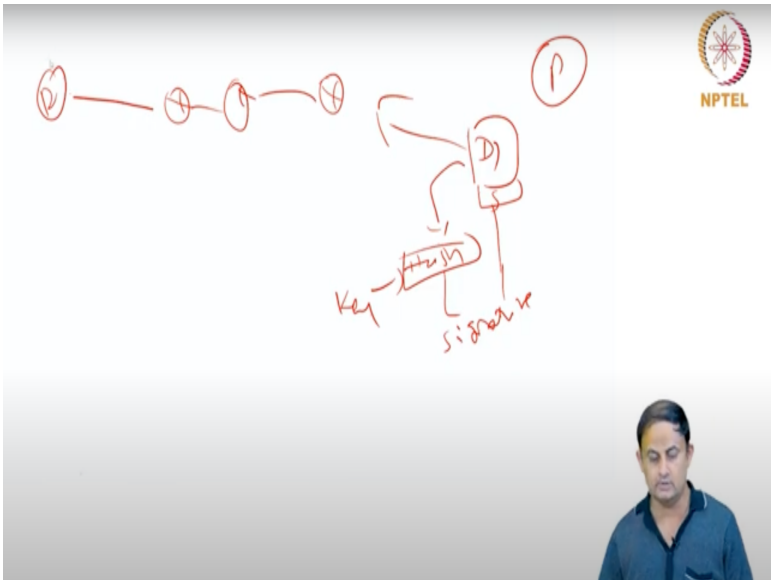
Wherever you visit, you just leave a copy there and then move. So $D_1$ came here, it left a copy in the cache of the $R_3$ and then moved towards the $R_2$, and then it left a copy there, and then it

moved to the next hop and so forth. So this is one of the simplest caching mechanisms that you can use, but there are other caching techniques as well. So this one is the basic caching technique but the other techniques which actually maximize the utilization; a simple technique might be I need not cache it on all the routers but only on the router which is directly connected to the client. So this is called as the Edge Caching mechanism.

So Edge Caching means that I cache the content only in the content store of that router to which the requester or the client itself is connected to. So that is called as the Edge Caching. So this is another caching mechanism. There are many, many other different caching mechanisms that are developed over a period of time, but the default and the simplest one is this leave copy everywhere mechanism, which actually stores a copy of the content $D_1$ in every router's cache.

(Refer Slide Time: 30:05)






So that is about the caching and the security in the NDN is in line with the security of the required in the ICN network. So we provide the security to the data chunks itself. So how does that operate? You take the content and the content itself is actually digitally signed by the producer.

So the requester is here, and the producer is here. The producer produces some content, let us say $D_1$. So you can see the content let us say $D_1$. So you take the $D_1$ and pass it on to some hash-based algorithm. You get a signature that will use a key whatever the signature that is generated, you attach to this, and then you transmit this to the intended recipient through some series of intermediate routers. So this is how any piece of the data, you take that, and then you attach the signature using a signature generation algorithm which is actually basically a hashing algorithm that uses a key, and then you transmit it over the network towards the receipient. So this chunk along with the signature, is actually going through this series of routers and then finally reaches the recipient.

So that is how the security in the NDN networks actually works. You basically sign the data chunks using a cryptographic algorithm. So that brings us to the question how does the receiver actually verify the content? So in order to verify the received content, you need to know the key that is used to sign the data. So assuming there is an offline mechanism to share the key with your recipient, then the recipient will be able to verify the content. So there are, again, different ways of delivering the keys or sharing the keys with the intended recipient.

So remember earlier discussion, we said that not all content can be shared with everybody. So we want to put some kind of restriction selected set of people who would only be able to view or access the content. So that is the access control mechanism, and we can actually control that as well. So if we have a key distribution mechanism, offline mechanism, or some other way through the NDN medium itself, we were able to deliver the keys to the intended recipient only they can actually verify that signature and read the content and then make sense of that content. Otherwise, it is not possible, through this mechanism, I can actually enforce the access control for the data that is generated by some data source.

And this kind of mechanism, providing security to the data chunk itself has got many advantages. So many of the cyber attacks that are possible in the IP networks are not exactly applicable in the NDN network. So, for example I cannot actually bombard a particular recipient with too many number of the packet because the routers themselves do not forward anything unless somebody has requested something.

Let me summarize what we discussed. So NDN is a clean state design. So it operates using the names, the data chunks are named; Hierarchical naming convention is preferred, and the flat naming convention is possible, and the namespace itself is not controlled by the or dictated by the NDN architecture. So, as long as you use any of the supported naming conventions you can use any possible name of any length in the NDN architecture, and the routing itself happens through the use of the three elements inside the router; one is the FIB store, the second one is the PIT table, and the third one is the content store. And the FIB serves to help you find out from where to get the content, PIT table tells you to whom to forward the data that is received, and the content store will keep the copy of the content, and the caching mechanism in the NDN is a default mechanism, very simple, which is the leave copy everywhere, all the routers which receive the content will store a copy in their caches and the security in the NDN is given through cryptographically signing the data chunk that is generated and which the receiver can actually verify subsequently. So that is the big picture of the NDN network, we will stop it here, we will come back and discuss the few other architectures in the ICN paradigm.