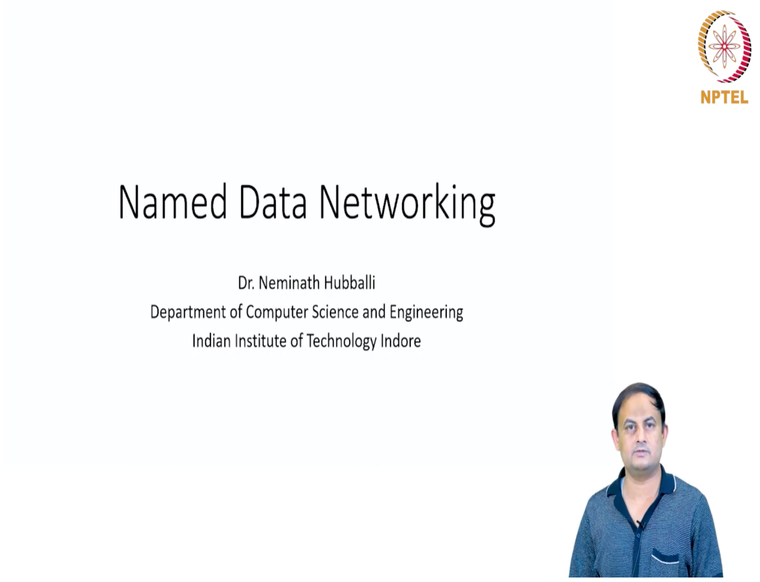
Advanced Computer Networks

Professor Doctor Neminath Hubballi
Department of Computer Science Engineering
Indian Institute of Technology, Indore
Lecture 66
Named Data Networking - Part 1

(Refer Slide Time: 00:16)

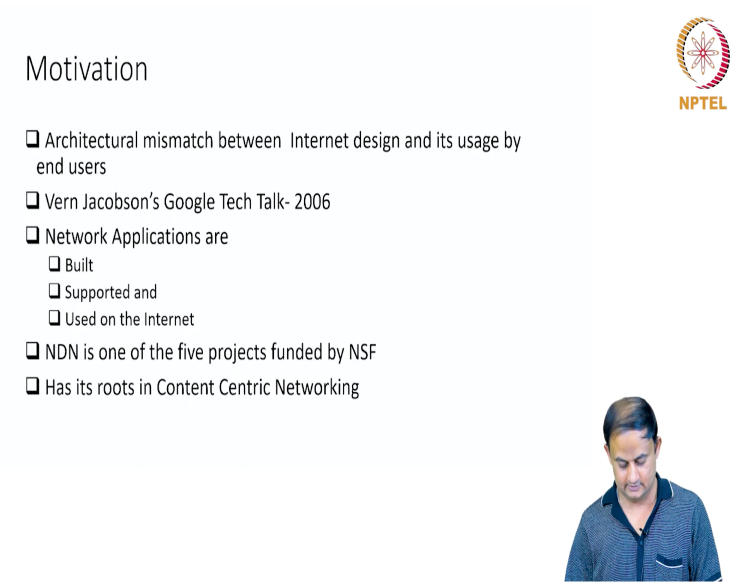


Named Data Networking

Dr. Neminath Hubballi
Department of Computer Science and Engineering
Indian Institute of Technology Indore

Welcome back. In the last couple of lectures, we were discussing about Information-Centric Networking. So, we spent a good amount of time understanding the background of the concept of information-centric networking and how it is actually supposed to work. So, what we will do today is we will take an example architecture of information-centric networking called Named Data Networking. So, we will go over the what is the architecture of Named Data Networking, how it works, particularly the four aspects of Information-centric networking, that is the naming the content, routing the content, security, and caching. So, these are the four components of information-centric networking; how each one of these actually works or at least has been proposed, we will discuss in today's lecture.

(Refer Slide Time: 01:18)

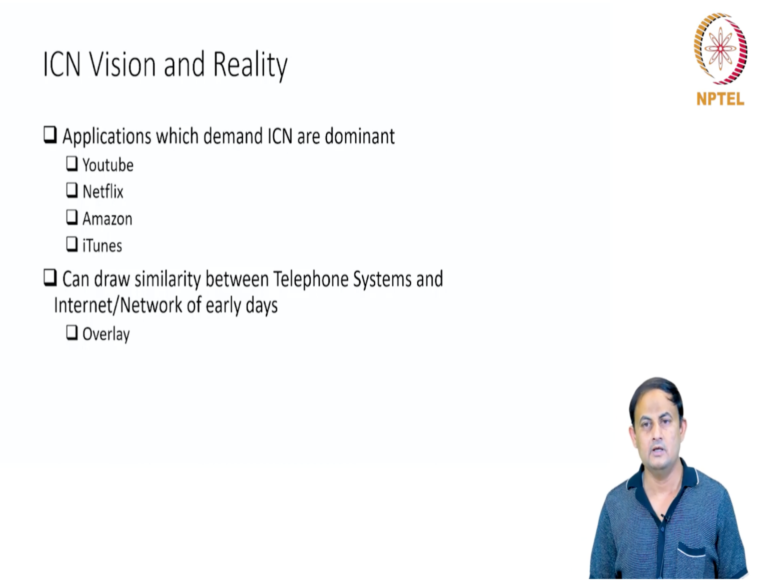

Motivation

- ❑ Architectural mismatch between Internet design and its usage by end users
- ❑ Vern Jacobson's Google Tech Talk- 2006
- ❑ Network Applications are
  - ❑ Built
  - ❑ Supported and
  - ❑ Used on the Internet
- ❑ NDN is one of the five projects funded by NSF
- ❑ Has its roots in Content Centric Networking

So, the motivation for this project, Named Data Networking, is again in line with what the Information Centric Networking actually proposed, to bridge the gap of mismatch between the architecture that the internet uses and what the users actually use in today's scenario. So, this is one of the projects that actually started along these directions, but not the only project; there are quite a few other projects also along this direction, but this is one of the prominent projects in this direction. And many of the projects that started in the Information Centric Networking actually in 2006, Vern Jacobson's talk at Google was the basis. So, he actually advocated the need for having such a network, and as motivated by that, this project actually started through a consortium.

And if you take the example of the applications or use cases in today's internet, the applications or network applications are built, supported, and used on the internet, but the usage is fundamentally driven by the IP addresses. So, we want to get rid of that IP address and have something different. So, that is what the NDN actually supports you. And this project, NDN (Named Data Networking), is one of the five projects that were funded by the NSF. NSF is the National Science Foundation which is a funding agency in the US. They funded five projects for developing the next-generation internet architecture, mainly the ICN-based architecture, and this is one of the projects that was funded for developing such architecture. And this project is Named Data Networking; although this is a unique project, it is based on an earlier project, something along the same direction, mainly the Information Centric Networking called the Content Centric Networking. So, many of the ideas in the NDN were borrowed from Content-Centric Networking or whatever, which seems to be promising; they are taken and then incorporated into the Named Data Networking project. We will see the details of how exactly these four things, what I refer to as the naming, routing, caching, and security, work in the NDN.

(Refer Slide Time: 04:08)



ICN Vision and Reality

- ❑ Applications which demand ICN are dominant
  - ❑ Youtube
  - ❑ Netflix
  - ❑ Amazon
  - ❑ iTunes
- ❑ Can draw similarity between Telephone Systems and Internet/Network of early days
  - ❑ Overlay

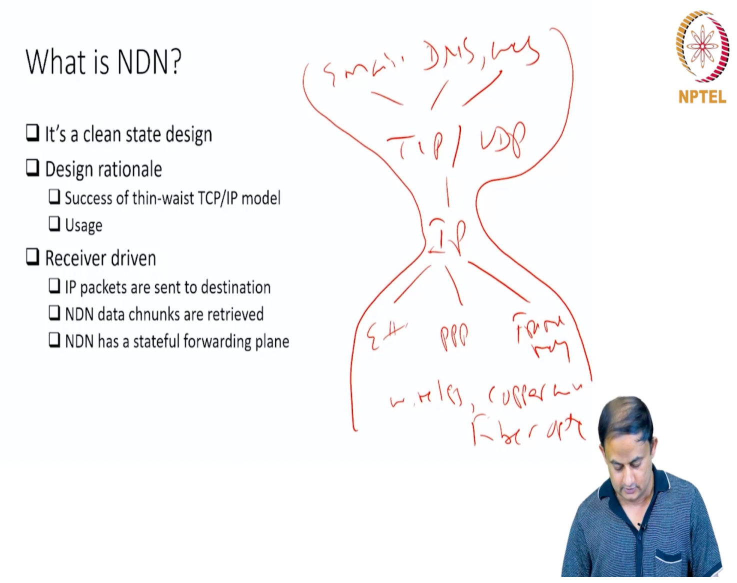
So, before we actually look into those features, the applications that actually demand the use of the ICN or ICN-based architecture, there are many of them in use in today's network. So, YouTube is one of the popular applications serving millions of users every day. So, Netflix is another application; the Amazon Prime video, which distributes video content to subscribers across the globe, is another application. So, Apple's iTunes is another application. So, all of these applications, if you recollect, are all content distribution applications.

So, mainly primarily, the video content is there, and that is actually distributed to consumers across the globe or at least whoever is subscribed to those applications. And in such kind of the video content delivery, people are interested in watching the video content a particular type of content and not worried about from where the content is coming. So now, these are the dominant applications on the internet today, and if the primary or majority of the consumption is driven by these kinds of applications now, do we still stick to the same old IP-based communication model, or we want to design something else? So, that is the question. So, NDN is an effort in that direction.

So, if I want to design an ICN-based networking architecture, there are two ways of looking at the design. One is so you can assume the IP network is there, it is working in some fashion, and I want to build something on top of the TCP/IP network as an overlay architecture. So, as an underlying communication medium, I can still use the TCP/IP communication model, but on top of that, I can engineer a different networking architecture. So, that is one way of looking or coming up with the new architecture.

The second way of looking at the problem is to look at something that is from the first principles you want to design the network. So, assuming the applications are looking something like this, the YouTube, and Netflix kind of applications are used in the network. So, keeping those use cases in mind, can I design a network from the first principles from a clean state?

(Refer Slide Time: 07:22)



What is NDN?

- ❑ It's a clean state design
- ❑ Design rationale
  - ❑ Success of thin-waist TCP/IP model
  - ❑ Usage
- ❑ Receiver driven
  - ❑ IP packets are sent to destination
  - ❑ NDN data chnunks are retrieved
  - ❑ NDN has a stateful forwarding plane

So, these are the two options the NDN actually takes, the second approach which is the clean state design. So, I want to design a network without carrying the knowledge of the previous history. So, what happens is when you want to come up with something new, so what is the existing, you tend to borrow the ideas or principles or tend to engineer something which is looking somewhat similar to the existing one.

But there is nothing wrong in using the old ideas that worked well. In fact, some of the ideas which work well in the IP network are also used in the NDN but you look at the problem from a different perspective. I want to have something like this from the first principle you start with, and then you borrow the ideas that work well or which actually fit into the new paradigm, and then from that, you take it forward. So, NDN said that we want to build a network architecture from a clean state. So, from the zero-knowledge, without borrowing the ideas or trying to build over something, over the existing ones they did not take that route but a completely different approach.

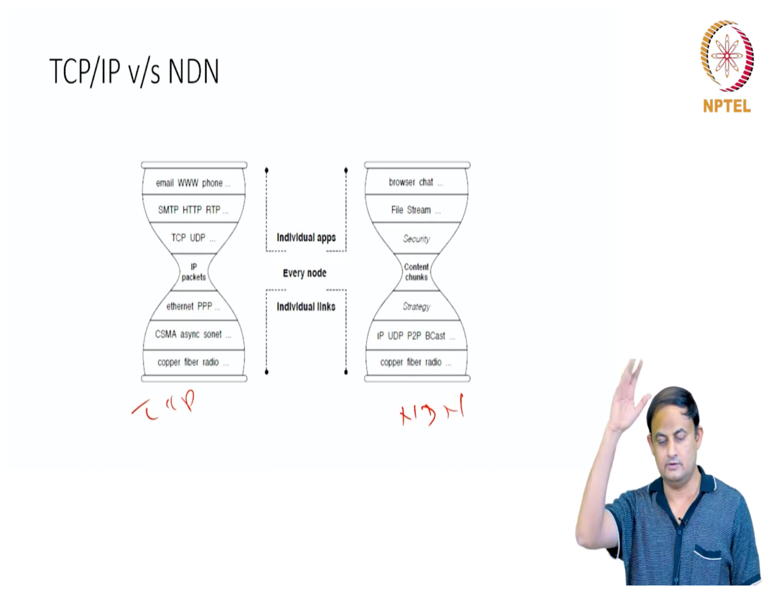
So, this NDN design as well was somewhat similar to the structure of the TCP/IP model. TCP/IP uses something called as the thin-waste model. What it means is the number of applications that we have is quite large at the application layer. The TCP/IP layer has these five layers. At the application layer, you have the email, you have the DNS, you have the web services but all of these actually use TCP or the UDP as a transport layer protocol, and then it uses IP as the one network layer protocol and network layer, this IP, in turn, uses many many different kinds of the link layer protocol, be it the Ethernet, be it the point-to-point protocol or frame relay whatever it is and then there are many many physical layer approaches maybe the wireless, maybe Wi-Fi is one approach, the copper wire, fiber optic communication all of these are physical communication means. So, what the thin-waste actually means is, that you have a bunch of things at the top and then you have one common thing in the middle and then many, many more things at the bottom.

So, this is called thin-waste model. So, this thin waste model is there in the TCP/IP communication model as I just drew, and can we take the motivation for design although I am designing a new network architecture from the first principles, I want to base that architecture on the success of this TCP/IP model which is looking something like this. So, now how do I engineer that? I want to look at the TCP/IP model only from the perspective how the TCP/IP model actually looks like,and then what I want to do is something that we said in the ICN.

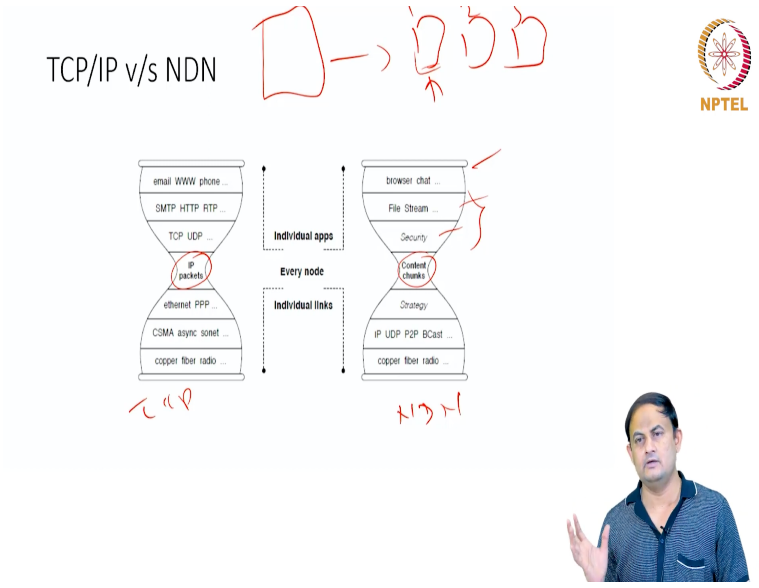
So, I want to build a network that is completely receiver-driven. So, as I said earlier in the previous lectures as well the IP packets are sent, based on the addresses, they go to a particular destination address and but in the NDN, we want to borrow, we want to bring the data chunks, named chunks from some place in the network to a particular receiver.

So, this is a completely receiver-driven operation, and unlike the IP model, which is a stateless communication model where the IP protocol does not remember whether the packet has been previously sent or not, unlike that, we want to have a stateful networking architecture.

(Refer Slide Time: 11:56)



TCP/IP v/s NDN



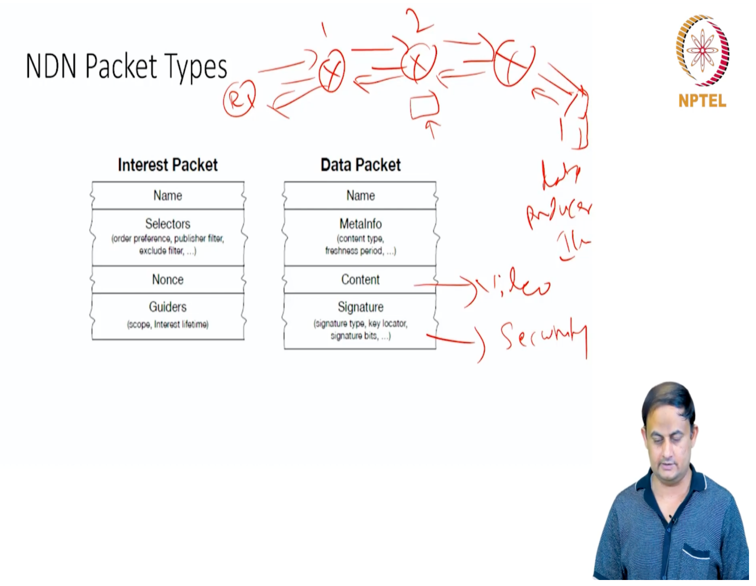
So, let us see how the NDN architecture actually compares with the TCP/IP model in terms of the overall hierarchy and also the features that the TCP/IP and the NDN have. So, on the left hand side, what you see on this page is actually the TCP/IP model, whatever I drew on the previous slide. And on the right-hand side, what you have is the NDN architecture. So, on the left TCP/IP model we just discussed, but on the right-hand side, the NDN has a bunch of applications again at the top.

So, this is also a thin-waste model, and then the data is getting transmitted from these applications using the security or inbuilt security to the data itself, and you generate something called as the content chunks instead of the IP packets in the TCP/IP model. So, it is worth noticing that here there is no transport layer protocol. So, only the file is broken into whatever data you want to transmit and is arranged in some pieces, and then the security is applied to those pieces, and they are independent chunks. So, you can roughly think of this, if you have a file of 1 MB to transmit, all that I can do is 1 MB file is actually broken down into many many small pieces, and then you apply the security to these pieces.

I give a name, and this is chunk number 1, this is chunk number 2, this is chunk number 3, and then I apply the security to these individual pieces and then transmit over the network, and whoever requests those chunks will be transmitted. So, that is the rough idea. So, the file can be anything, this can be a text file, this can be a movie or anything that you can think of, and then there is some transmission medium available. So, you can even visualize this as the even the entire TCP/IP architecture is one way of communicating the data chunks over the network.

But there might be other ways of doing it. So, irrespective of what kind of communication medium and the strategy that you have, the network itself is operating like this. You have some application giving you the data, I take that data, generate the chunks, apply the security, give a name, and then transmit and the network will make sure that this content is actually transmitted over the network and delivered to the intended recipient. So, that is what the NDN's overall architecture looks like. So, it is still a thin-waste model. So, content chunks are the primary pieces of the data exchange in the NDN network. So, that is how it compares with the TCP/ IP model.

(Refer Slide Time: 15:08)



NDN Packet Types

Interest Packet

- Name
- Selectors (order preference, publisher filter, exclude filter, ...)
- Nonce
- Guiders (scope, Interest lifetime)

Data Packet

- Name
- MetaInfo (content type, freshness period, ...)
- Content
- Signature (signature type, key locator, signature bits, ...)

And this communication NDN, whatever the data chunks that are getting transmitted happens through two different packet types, two different types of the data that get generated. So, one is something called as the Interest Packet and the second one is called as the Data Packet.

The Interest Packets flow from the receiver to the someone who is giving you the data. If my receiver is here R1, there is some producer at some point. So, here is the data producer, and the Interest Packets flow from here to the producer, and the data chunks are transmitted in the reverse direction. This is the overall flow of the two types of packets, but the interest packet and the data packets need not go all the way to the data producer. Since the routers in the ICN paradigm have got the storage capacity, you can find the requested content on the cache of the different routers.

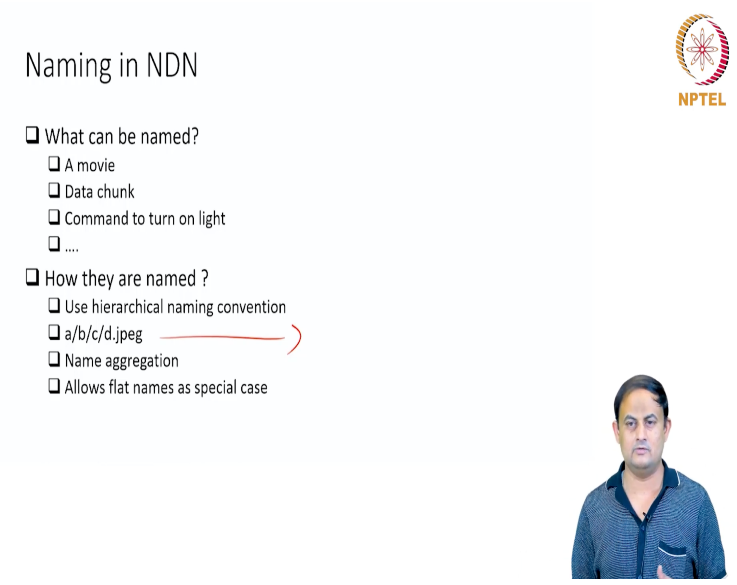
So, if some router along the path can actually give you the data, for example, router number 2, if it has the content, the interest packet will flow only up to router number 2 and then the content will be served from the cache of the router number 2. So, that is how it works, and the interest packet has the following components. So, it identifies the chunk with a name, the data chunk that you want to get it sends that name, I want this kind of data, and then it might also send you some kind of additional piece of the information. So, from whom do I want to get this data? I can set some filters to the data.

So, I want the fresh data. So, for example, if the content is generated one hour ago from the producer and that content because of the previous transmission has been cached at router number 2 and if the interest packet goes there and the interest packet might tell the router that we have the data which is not older than one hour then only you can send it. I want the data which is as recent as less than one hour. So, anybody who has that data only they will be able to serve. So, whether it is the intermediate router or if none of the intermediate routers have that fresh data, the request will go to the original producer, and then it will fetch the content. So, such kind of filters and additional constraints, I can put inside my interest packet, and it is the job of the network to find out whether those constraints are actually met or not met.

And then it has also got a Nonce, which actually identifies against which interest packet, this response, this data has come, and also it will help the intermediate routers to remember if they do not have the content they will forward this interest to the next hop and borrow the content from the either from the next hop or from the producer and then they can match against which interest packet, this data has come and then it will try to understand this response has come from so and so and against this request, I need to send it to the next hop on my left side. So, that is what the Nonce actually will help you.

And similarly the data chunk has also a name, and then it has got a meta information just like the interest packet can put some filters, the meta information can identify the content type. So, whether it is, a video content, whether it is audio content, or it can also have the size of the data and then how fresh the data is, and also the producers can sometimes tag, I generated this data if anybody is caching this content, how long can you do that so whether it is one minute, two minutes, 30 seconds, one hour, one week; whatever the lifetime or validity of that cache content is, you can put that information inside this and then the content itself if it is a video, the video itself is there and then at the end of this the security mechanism which gives the authenticity of this data which is called as the Signature. So, the Signature is the security piece. So, all this actually are four things that are part of the data packet of the NDN architecture. So, using these two packets only, the entire communication actually happens in the NDN networks.

(Refer Slide Time: 20:20)



Naming in NDN

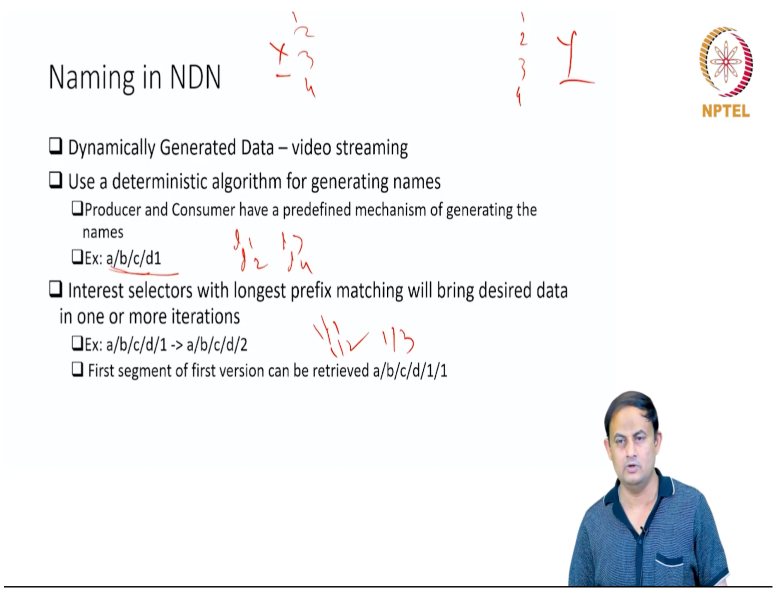
- What can be named?
  - A movie
  - Data chunk
  - Command to turn on light
  - ....
- How they are named ?
  - Use hierarchical naming convention
  - a/b/c/d.jpeg
  - Name aggregation
  - Allows flat names as special case

So, with that background in mind, let us see how the naming actually happens; the four things that I was referring to earlier, how the naming happens in the NDN. So, what are the available possibilities one is to use the Flat Name, the second one is to use the Hierarchical Name, and the third one is using some of the attributes you can generate the name. But the NDN primarily goes with a Hierarchical Name.

So, you can use something like a/b/c, just like the hierarchical naming convention that we discussed in the previous lecture using the same naming convention, I can name the content. And what you can name? So, in the NDN architecture, any piece of data, be it the movie, be it any text document, be it any command that you want to pass on to a different computer in the IoT networks where remotely, you can control some of the operations like I can turn on, turn off the light I can turn on, turn off the fan, all of that can be done. So, you can name the content to turn on light, turn off a fan, or something like that; that is also possible. The name can also be given to a data chunk that is transmitted in the vehicle or networks. So, these are some of the promising use cases for all of them; virtually any data type that you can think of, any application that is generating some data, all of them can be named. So, how do we name it? we use the hierarchical naming convention but that is the default and actually suggested way of naming the content, which actually allows you to do the aggregation and also makes the life of the NDN routers so simpler.

But that is not the only way to name the content if you still want to use the flat naming convention for some applications; if somebody wants to fall back to that flat naming convention the NDN still allows you to use that as a special case. So, a hierarchical naming convention is preferred but a flat naming convention is possible in the NDN networks. So, at any point of time you can think there are some data chunks which are named using the hierarchical naming convention, and some other data things are actually named using the flat naming convention.

(Refer Slide Time: 23:05)



Naming in NDN

- Dynamically Generated Data – video streaming
- Use a deterministic algorithm for generating names
  - Producer and Consumer have a predefined mechanism of generating the names
  - Ex: a/b/c/d1
- Interest selectors with longest prefix matching will bring desired data in one or more iterations
  - Ex: a/b/c/d/1 -> a/b/c/d/2
  - First segment of first version can be retrieved a/b/c/d/1/1

So, that means I said that the video, the audio, the text file any data that gets generated can be named in the NDN architecture, but the video content need not necessarily be static content like the one uploaded. So, think of this case if I am doing a video conferencing application, set of people are actually engaged in a meeting, and then this is dynamically generated data which is live streamed.

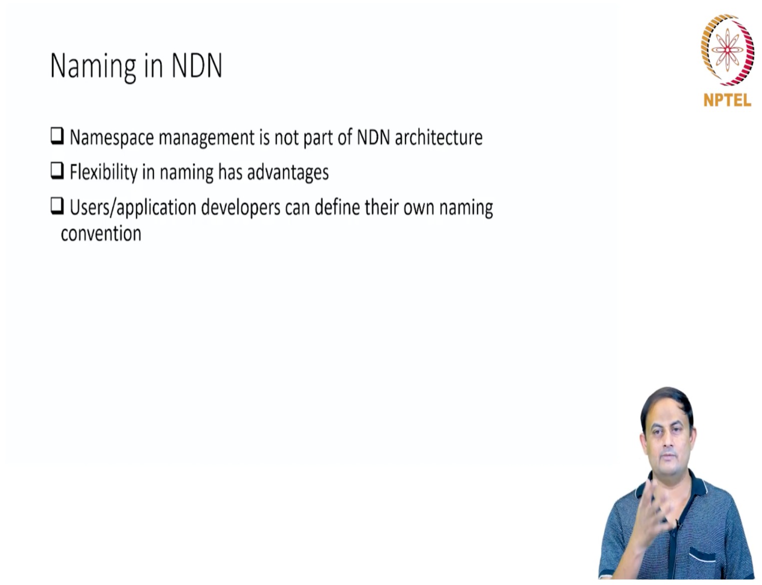
Now, how do I name the content in this kind of scenario. So, we need a mechanism to generate or assign some kind of names to this dynamically or lively generated data like video streaming. So, one of the approaches that you can take here is something like this. So, you use a deterministic way of generating the names of the data chunks in the sense that if X is the receiver of the data and if Y is the producer of the data, both X and Y know what algorithm to use. So, if I name, let us say the chunks as 1, 2, 3, 4, the X knows how to query the next content next chunk.

So, it will use the sequential numbering scheme 1, 2, 3, 4, something like this, and then it will be okay, you have sent me 1, and now you send me next data chunk 2, now you send me next data chunk 3, something like this. So, it need not be as sequential as this one but just an example; this is how it determines both the end parties that are engaging in the conversation. They need to know what is the next chunk, how the next chunk is actually named in this scenario. So, even if you use the hierarchical naming convention, I can still do this dynamic or deterministic way of assigning the names to the content something like this. I use a common prefix a/b/c like this and then I can see d1 is the first chunk, then d2 is the second chunk, and d3 and d4 something like this.

This is a deterministic way of using a common prefix, but I vary only the last portion, and then I generate the data chunks, and using the same naming convention, the receiver will also query the network, and the content will be delivered to the intended recipient. So, of course, you can keep the lifetime of these packets to a very small span of time so that the intermediate routers actually do not cache this content for a longer duration. But the point I am trying to make is, it is possible to name the content that is dynamically generated which has a very short span in the sense that if you have static content, be it the video, audio or text file, anybody can cache it as long as the lifetime of that content is very large if you have web page still that can be cached. So, those are the kind of the stuff that you can afford to cache, and the lifetime is larger, and the content can be stored across multiple routers in the network. But the applications which are actually operating in the real-time be it the audio conversation between two people using Voice Over IP communication and talking to each other or the video conferencing application, these are the stuff that actually warrant a little more investigation how do we name the content here and how do I make sure that the old content stored in the routers are not served as a response to the fresh request.

And this is one way of handling such cases. And I can even take a slightly different approach if a/b/c/d is there, and then I can make sure you send version number 1 (V1), version number 2 (V2), version number 1 or 3 (V1/V3) something like that. So, depending upon how you name the content, as long as the two endpoints actually know how the names are assigned and how to query, then the system will work in the NDN architecture.

(Refer Slide Time: 27:34)



Naming in NDN

- ❑ Namespace management is not part of NDN architecture
- ❑ Flexibility in naming has advantages
- ❑ Users/application developers can define their own naming convention

So, this kind of naming convention assuming you use the hierarchical naming structure, and because there is a way of assigning the names to the data chunk that gets generated, it brings the structure to the entire operation. So, one thing is common between the TCP/IP networks and the NDN network is both the name series management is out of the architecture. So, even the TCP/IP architecture does not tell you who should get what IP address, who should be owning that IP address, and who assigns that IP address to a particular computer. So, as long as you have an IP address, then the communication is possible.

So, just like that, similar to that, what the NDN says is as long as you have named content, you have assigned a unique name to the data chunks, then I will be able to carry those named chunks inside my network, that is what it says. So, this kind of flexible naming convention, a third party controlling the assignment of the IP address or a third party controlling the naming of the data chunks will actually bring flexibility or it has its own advantages at least from the design perspective, from the architecture perspective; it actually makes the life simpler. Otherwise, how to name the content if you put it inside the architecture then the flexibility is removed. If sometime later I want to change or bring some more innovation to the naming convention, if that is fixed and all your routers are expecting the same way of naming the content, I cannot really change that subsequently.

So, that kind of flexibility you can bring if you decentralize the naming in the sense that the NDN architecture itself is not proposing a mechanism to name the content. So, although hierarchical naming convention is used, two different people who are connected to the network can use two different prefixes and then generate the data. So, what prefix to be given to which data generating entity is the question that is the namespace management operation. So, that is out of the scope of the NDN architecture, which is what I am referring to. And users or our application developers can define their own naming convention.

So, as long as it is actually adhering to whatever the NDN architecture can understand using the hierarchical naming convention or a flat naming convention. So, it is good to go.