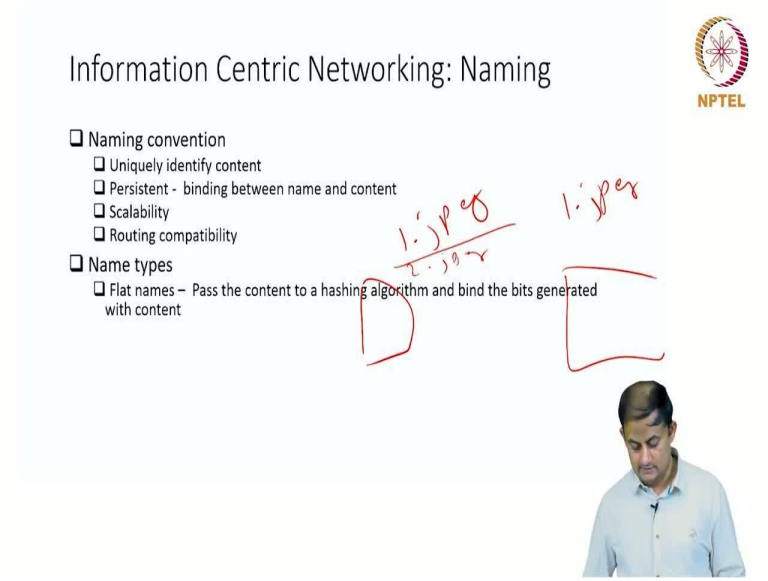
Advanced Computer Networks
Professor Doctor Neminath Hubballi
Department of Computer Science Engineering
Indian Institute of Technology, Indore
Lecture 64
Information Centric Networking Part 2

(Refer Slide Time: 0:17)

So, with that requirement, whatever the research projects are there, they actually fundamentally deal with these four kinds of design goals, one is naming the content, the second one is making the routers aware of the names, and bringing the ability for these routers to forward the request based on the name and store the names as a part of the routing table, and minimizing latency in the accessing the content, if I am able to find the content in one hop distance, so I will be able to get the content quite quickly. And then making the content itself secure rather than establishing a connection to the host which is actually having the content.

(Refer Slide Time: 1:15)



So, with that background in mind, let us explore each one of these requirements in a little more detail. So, I said that you require a mechanism to name the content in the Information-Centric Networking. And now, how do I name the content? I need to bring in a mechanism to name the content. Since the routers are doing the lookup operation, whether the content that is requested by any of the clients exists in their own local store or elsewhere in the network is done by the names. These names need to be globally unique, globally unique in the sense that on a network of the scale, as big as the internet that exists today where at least, if not billions, at least a million of users are connected to the network are constantly generating the data.

So, think of this case, everybody globally is using social media networks like Twitter, Facebook, to generate content or upload videos on YouTube and many other social platforms. So, everybody's content should have a nickname. So, if the one of the copy, one of the content that I generate and you generate have the same name, then the routers will have an issue in identifying the content, whether the client that is requesting the content of me or content generated by you.

So, we need a mechanism to name the content uniquely. And the binding between the name and the content itself, exists. And it is a persistent name; it should not happen that for some point of time, like currently, the content let us say the 1.jpeg is the name that is given to some content, some image that exists in one of the client repository or the router repository and the same 1.jpeg is also used for some other images which exist in some other router.

So, that is going to create conflict. And that we do not want. So, now on a global scale, and at least with the millions of users constantly generating the data, can we have a naming convention, which is actually robust enough to assign a unique name to every piece of content that is generated? So, that brings us to the question of scalability. So, can it scale to that level? So, how many such unique names I can create? And if at all I can create how do I create them? So, there is an inherent limitation on how many unique names I can assign to my own content.

So, I can give ebc.jpeg, 123.jpeg, but at some point of time, I might run out of the image,s or the names themselves become quite big, which actually causes another problem. So, it is not the ability to name, even though I can let us say, I can come back or assign a name which is a thousand characters in length. Can the users or the consumers remember that thousand character length name unique name and then make a request?

So, because the routers are operating only using the names, and if I am using a thousand characters or a thousand bytes names. So, it also means that the clients who are making the request also need to remember or at least know how to systematically query with the name, which is a thousand characters or bytes long. So, that is to be achieved. And in effect, the storage and the routing also is happening with the name. So, that ability also should coexist with the naming convention that we use.

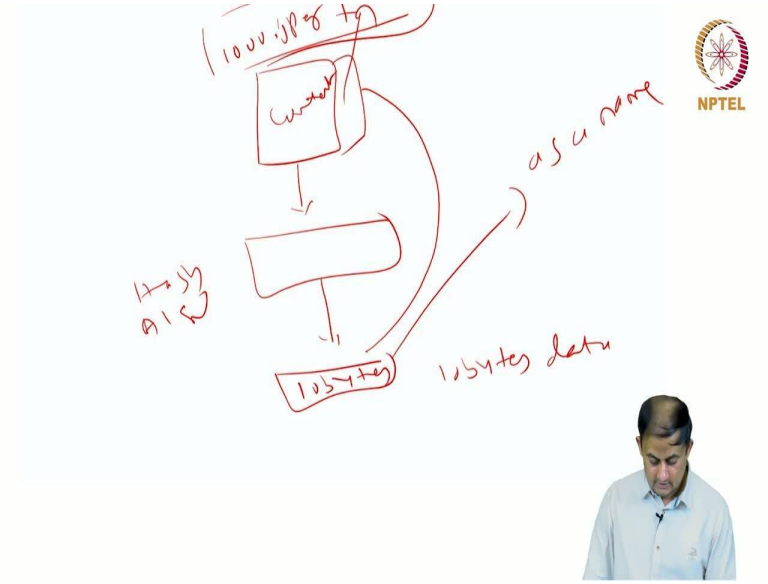
So, now let us try to understand how can we assign such a name. So, one of the mechanism that you can use to assign the names to content is something called as the Flat name. So Flat name, is it assigns a standard nickname something like 1.jpeg, there is one name, whether it is 1 dot jpeg, or 2.jpeg, 10.jpeg, 100.jpeg, 1000.jpeg, whatever is there.

So, this is a Flat name. So, what I can do is I can assign such a name to the content. Let us say hypothetically you want to assign a name, if I am the content producer and name, which is 10,000.jpeg, to an image that I just generated. So, how do we make sure that it is the 1000.jpeg does not exist in the network?

So, which brings us to the question that across all the routers, which are connected, which are part of the internet, or network, which is as big as the internet, in none of these router's cache, the 1000.jpeg exists. So, we need to first search, and then if it does not exist, you need to assign

the name 1000.jpeg to the newly generated content. So, that becomes infeasible searching in every router, which exists in the network is probably not feasible to do.

(Refer Slide Time: 7:18)



An alternative mechanism to assign the name to the content that is generated, you something called as the Hashing mechanism. So, let us say I generate a piece of the content, maybe an image, maybe a text file, maybe a movie, and this is the content. And what I do is, I take this piece of information and pass it on to something called an Hashing algorithm. And this Hashing algorithm gives me a piece of a unique fixed-length output. So, maybe let us say this gives you 10 bytes of the data, which is also called as the hash.

So, I take these 10 bytes of the hash. And then I can treat this itself as a name. Now, I generate some piece of the data, that is, an image, and then pass it on to a hashing algorithm. And then, as part of the naming convention, I use hash itself as the name. So, I come back and tag this hash to the content itself. And anybody who is interested in getting the content, this image that gets generated needs to query using these 10 bytes of the hash value.

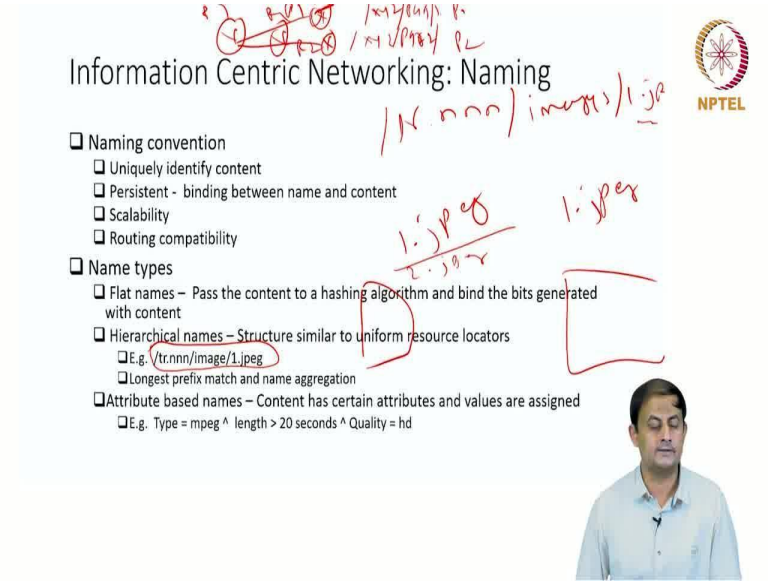
Now, this is what the Flat naming convention is. So, now how does an arbitrary user who is interested in some random content is going to know the hash value that gets generated? So, a compromise between the hash value and the ability of naming the content, is to combine the

user-generated names. So, for example, a user wants to name this content as 1000.jpeg, and what I am going to do is this 1000. jpeg plus this hash value becomes the name for this content.

So, at least if the user queries with the 1000.jpeg, then along with that, I am going to put this name and populate it in the network. Anybody who queries with the 1000.jpeg as a name should be able to identify all those pieces of information in the network which is tagged with or at least having the name as 1000.jpeg. So, that is what the Flat name naming actually means.

So, this actually, Flat naming is an idealistic scenario. And practically implementing that, or translating that into reality, is going to be challenging, because now hash is going to be part of the name itself, then you still need to make the end user who is querying for certain content, what would be the hash value that is part of the some name.

(Refer Slide Time: 10:36)



So, that is one way of naming the content. And purely just naming content from a generation perspective and also from the distribution perspective is not scalable. The second way of naming the content is using something called as the Hierarchical naming format. What it does is, instead of naming the content as 1.jpeg or 100.jpeg or 1000.jpeg, or something like this, I bring some structure to the content that I generate in a sense that, I am going to use a naming convention, which is looking much like an URL.

So, here, in this case, tr.nnn/images/1.jpeg, I am going to all the images that are generated having this prefix, tr.nnn/images. So, every content producer, if they are generating the image will have such a prefix used. So, if I am generating, I might use a prefix, which is ending with image and then the different images of my own; I can still use the 1.jpeg, 2.jpeg, and 100.jpeg as many images that I can create, but whatever the content that I generate is having the common prefix as this one.

So, someone else might have the prefix which is looking like this tr.nnn/images/1.jpeg as long as the prefix of mine and that user is different, then the routers in the network will have no problem in uniquely identifying the content. So, purely this mechanism of naming or naming convention is scalable. And as long as every distinct producer has a unique prefix assigned to it, then I can go and query that content and figure out where exactly that content is, I can borrow that content and then deliver it to the user.

So, this mechanism is somewhat similar to what is done in the current IP network. So, IP networks use the IP address as the means for locating the content or borrowing the content from that particular location. And it uses the IP prefixes, and the search operation itself is done using the longest prefix-matching technique. And it also uses routing aggregation, the common prefixes you take together, and then you condense them and then put it into region.

So, the same mechanism can also be used here as well. So, for example, if there are two producers, P1 and P2, and if these two producers use the common prefix, which looks something like this, so maybe P1 has the prefix xyz/pqr, and then 1.jpeg, 2.jpeg something like this, and the second producer has got xyz/pqr2 as a prefix and then he can name the content.

So, if these two producers are connected to two different routers, R1 and R2, let us call this R1 and R2, and a router R3 is connected to both R1 and R2, and R3 might store a kind of common prefix for both R1 and R2 or if both R1 and R2 say are on the same link somewhere here is the R1 and somewhere here is R2, then both the prefixes are going to the same next hop router.

So, I can use a common prefix which is xyz/something like xyz*. So, anything that is originating when the client makes a request irrespective of whether it is xyz slash pqr1, or xyz slash pqr2, I still send it to the same outgoing link or on the same next hop link. So, in effect, what I am trying to do here is the condensation or the aggregation of the unique names that exist in the.. exist in

the network. So, that is possible, if you use a structured naming convention like the hierarchical naming convention.

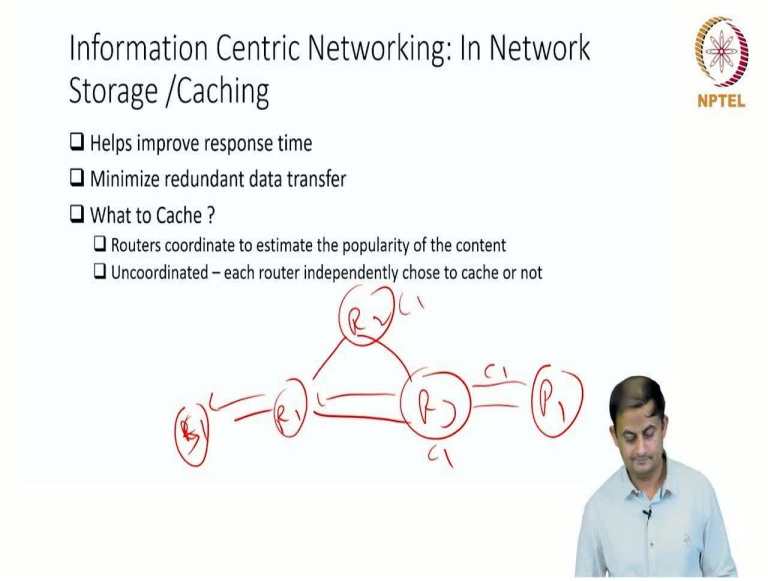
And the third way of generating the names or assigning the names to the content that gets generated is something called using the Attributes. So, every piece of information, so, be it the image, or the text, or the movie or whatever content that you generate, has some attributes. So, attributes in the sense something might be the size of the file, something might be the type of the file, whether it is the jpeg image or mpeg image and the length of the file and whether it is a high definition video or low definition video, whatever it is the compressed video, coupled with the name of the author these are all some of the possible attributes.

You take these attributes and then using that as input, you will pass on these, think of this as these are different attributes that are passed to a function, and that function using all these attributes comes back with a unique name, and that name you assign to the content and populate in the network using which the consumer actually subsequently raise the request and get that content.

So, let me summarize, Flat names use a hashing mechanism, and then Flat names not only pose challenges for naming, it also pose challenges for routers to navigate and locate the content. And if you have a Flat name convention, then the number of unique content that gets generated at least on a network as bigger on a scale as that of the internet. So, there are many, many, at least millions of unique content get generated every day. And storing those many number of any contents in every router is going to be quite a challenge.

And on the other hand, if you use the hierarchical naming convention. So, you can bring some notion of the structure, and then you can aggregate the names using the common prefixes, and then you can do the forwarding and the lookup operation. And the third way is to look at the attributes that are existing with the data itself and you give take these attributes and then you generate the unique name to the content that gets generated. So, these are the three possibilities that exist.

(Refer Slide Time: 18:09)



So, that is one challenge that we wanted to address in the Information-Centric Networking, which is the naming the content. The second mechanism that we want to address is something called as the Caching mechanism. So, meaning as I said, as the content moves from the producer to one of the recipients, all along the paths, routers can make an independent decision to cache the content.

Now, the question is, if every router in the network starts to cache the content generated by every producer that exists in the network, then how much storage capacity does the router need to have and clearly, it is imperative that every router cannot actually cache the content that is generated by every producer in the network. There is some limit on how much of the storage capacity you can put inside the router.

So, that brings us to the question, so, at what point of time the router make a decision to cache what content, is it based on some rational argument, or what is important to cache? If I am discarding something, then on what basis I am discarding something? So, those are decisions we need to make. So, that is what is called intelligent caching and storage.

And in effect so, by storing the content closer to the consumer, we are improving the response time. But that comes at the expense of the storage cost you want to put the storage at every router. So, now I can trade off. So, the more the amount of caching ability that I have, more the capacity I have in the routers, and more the content I can get stored, and that improves the

response time. But on the other hand, if my cache is limited, then if I do not have it, then I need to ask my own neighbor.

So, you need to cross some number of the hops in order to get the content, so there is a trade-off. Now, I want to minimize the response time. And at the same time, I want to store more number of unique contents as many unique content as possible, so that I minimize the number of times I actually go to the producer to get the content that is the objective.

Now, this can be achieved by kind of strategies, which broadly fall into two categories. One is called the Coordinated mechanism to do the caching. So, coordinate in the sense that assume this, if there are three routers or n number of routers in the network and they are connected in some fashion. And the R1 knows that some piece of the content C1 exist in R2 and since R2 is the direct neighbor of the R1. And when the same copy C1 is actually passing from some producer, P1, to some receiver S1, so if C1 is going from here to here, and then here to here, and then here to here. And R1 can notice that C1 already exists in R2, and probably R3 might have also cached a copy of that content, then probably R1 may decide not to cache that content C1 although idealistically, we wanted it to be cached at R1. But since there is cooperation, so, we want to judiciously utilize the collective storage capacity, the collective capacity of all the caches that exist in the network, to reduce the latency of the access for the end consumers.

So, if the routers in the network use such a mechanism to coordinate among themselves to intelligently decide what to cache and what not to cache. So, that mechanism is called as the Coordinated mechanism; some kind of cooperation exists that maximizes the utilization. So, more diverse content, you can actually put into the network. What I mean by that is that if every piece of content is stored by every other router, then when the capacity at the R1 gets full, if a new content comes, then you need to discard some of the existing content and store the new one.

So, there is a lease recycling mechanism when the storage gets full, you discard some of the old content that exists in your local stores and then store the new one. So, that also brings us to the question of what to discard. So, it is not only what to store but what to discard and at what point of time I am going to discard that we are to decide here. So, this is what the storing convention is. So, on the other hand, if every router is making its own decision independently whether to

store a particular content or not, irrespective of whether their neighbors have the same copy or not is called an Uncoordinated mechanism.

So, again, there is a trade-off here. If we use the uncoordinated or the second method, then the operation becomes simpler. Every router does not have to bother about what its neighbor is doing. And then, if I want to cache it, I will just cache it, if I do not want to cache it, I will not cache it. If I want to replace something, then I will replace something on my own. And if I do not want to replace something, then I decide on my own. So, the operation is simple.

On the other hand, if you use the first method to do the caching and the replacement decisions, then you need to know constantly, maybe at every regular interval of time, you will need to go and find out what is there in the router R2's cache and what is there in router R3's cache.

Now, the question is how far I can go, whether it is only one-hop neighbor or two-hop neighbor or n-hop neighbor that is the question. So, again the more the number of hops you go away, you go on, so the more diverse content you will be able to store in the network, and that comes at the expense of the communication, which is just to find out what is there in this neighbor's cache. So, even though consumers may not have asked for any data, still at regular intervals point of time I need to go, and the router needs to go and check what is out there in the neighbor cache.

So, again I am going to make an inclusion decision to cache and evict content from the local cache, but that comes at the expense of the communication overhead. So, what I can afford, whether it is the communication overhead or the ability of the storage that I want to put inside the router, that trade-off or the decision you need to make. So, that is the second objective that I want to bring into the Information Centric Networking.