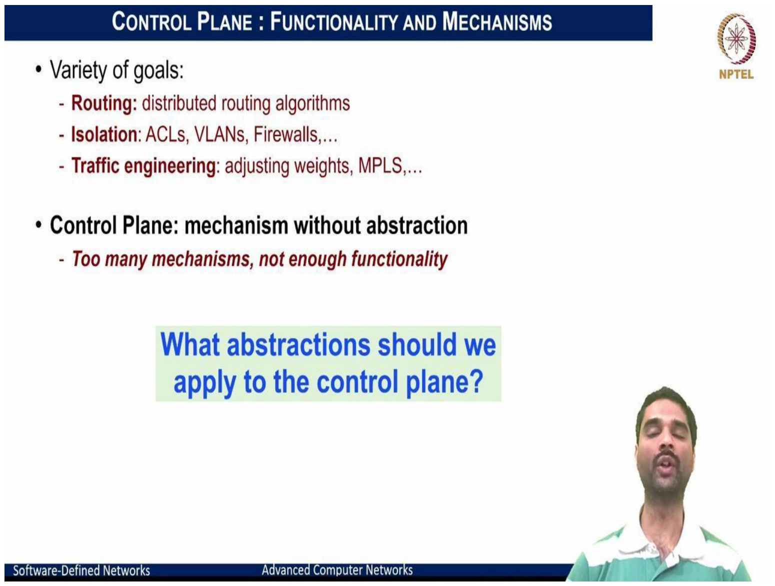
Advanced Computer Networks
Professor Sameer Kulkarni
Department of Computer Science Engineering
Indian Institute of Technology, Gandhinagar
Lecture 31
Control Plane Abstractions

(Refer Slide Time: 0:17)



Control Plane : Functionality and Mechanisms

- Variety of goals:
  - **Routing:** distributed routing algorithms
  - **Isolation**: ACLs, VLANs, Firewalls,...
  - **Traffic engineering**: adjusting weights, MPLS,...
- **Control Plane: mechanism without abstraction**
  - ***Too many mechanisms, not enough functionality***

What abstractions should we apply to the control plane?

Software-Defined Networks Advanced Computer Networks

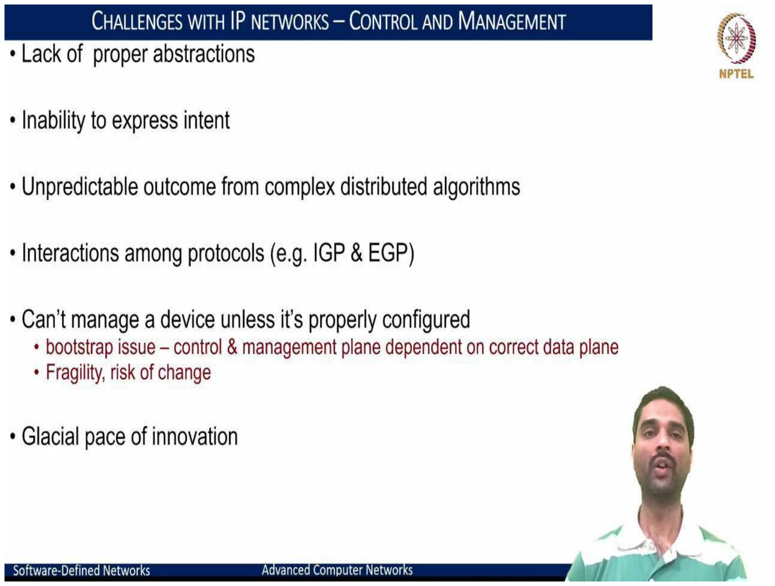
When we look at the traditional networking elements such as switches and routers, and try to see the role of the control plane, we see that they serve a variety of goals, foremost is the routing. Here, they may perform a variety of distributed routing algorithms; for example, it could be link-state, or distance vector algorithms that we can configure, and routers would interact and build the routing tables.

Second, they provide the mechanisms for traffic isolation by means of configuring the access control lists or setting up of the virtual LAN IDs, or even could provide the firewall functionalities to drop a specific set of flows. Third, they would also provide the means for control in terms of traffic engineering and the mechanisms to aid different aspects to do traffic engineering on those devices. By traffic engineering, what I mean is the process of reconfiguring the network in response to the changing traffic loads so that we are able to achieve the desired operational goals.

In essence, the traffic engineering would cover all the measures that would be needed to optimize and control the traffic that these specific network elements would process. And this would be able to guide us to have better throughput and desired QoS levels. And as such, traffic engineering is a part of network planning, operations, and management, especially from all the ISPs do this. And this would include the mechanisms such as admission control, differentiation of the services, and even building failure resilience. You might have heard about the MPLS; it is multi-protocol label switching. And also, you would have learned in your undergrad about how you set the weights for each of the links in order to meet specific aspects of what links need to be considered. When you want to route that is done through the adjusting of the weights.

So, we can see that the current state of the control plane is quite complex, with rich set of mechanisms, but no abstractions as such. Hence, the question would be to see if we should apply the abstractions, then what kind of abstractions should we be applying for the control plane?

(Refer Slide Time: 2:59)



CHALLENGES WITH IP NETWORKS – CONTROL AND MANAGEMENT

- Lack of proper abstractions
- Inability to express intent
- Unpredictable outcome from complex distributed algorithms
- Interactions among protocols (e.g. IGP & EGP)
- Can't manage a device unless it's properly configured
  - bootstrap issue – control & management plane dependent on correct data plane
  - Fragility, risk of change
- Glacial pace of innovation

Software-Defined Networks Advanced Computer Networks

To do this, let us look at the key challenges with respect to the control and management of these network elements. One, we already outlined that there is a lack of abstractions that would enable us to build the right means to interact and facilitate the services. But second, when we look at these abstractions, we also have to think of, why abstractions. In essence, what the abstractions would allow us is to preserve the core essential characteristics of the device and take out the unwanted complexities that we may have to deal with when we are trying to configure and manage these specific devices. What this means is, from a network operator's perspective, if we think what we would need is not how we should be able to update but understand that we should be able to express our intents in a very simple form. Like if I want to configure a specific routing algorithm, I would just express the intent that I want to set link state routing. And once we do that, we also want to control specific aspects of link state routings to be expressed like in terms of what weights we want to set in an easier and more coherent manner.

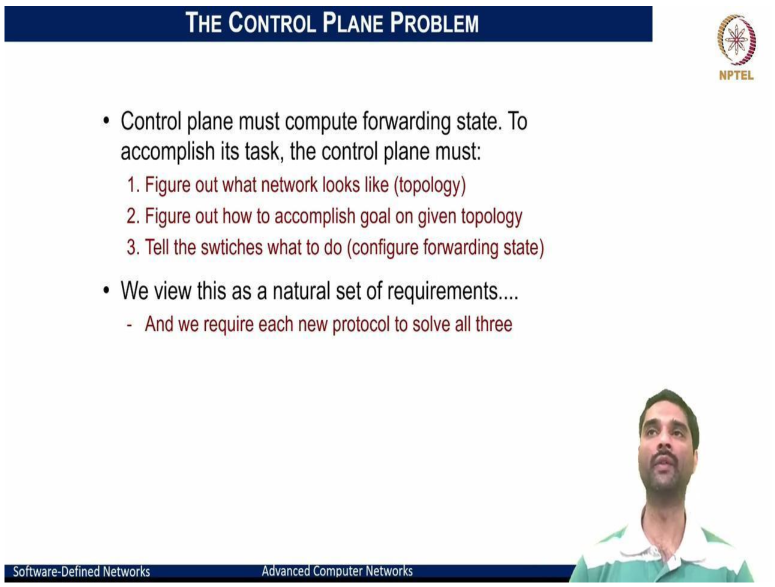
Nonetheless, what we see in the current model when you mentioned the routers configure the updates on the router, these routing algorithms are often distributed and run on their own. That means the intent what a user wants for a specific flow to take a specific path may not be met at all, because these routing algorithms would learn based on the desired weights. And many a times you would have to go back and re-adjust the weights to see, okay are we meeting our outcome? And this makes it much more complex to operate with this kind of element.

Further, when you have these multiple routing algorithms like internal and exterior gateway protocols that have to co-interact and enable you to build the necessary forwarding rules. These interactions also in a way, affect the way that this device would behave in terms of the forwarding plane. Hence, it becomes really unpredictable. And also, it becomes difficult to manage the way the devices will be configured properly, as we expect with our intention for the device to work.

Moreover, there are also bootstrapping issues that would come up when we want to have the correctness of the data plane reflected by the aspects of how the control plane is setup. And there is also this fragility or the risk of change that would occur when any of the things in the data plane change or any of the configurations that would impact back on the control plane setup.

All in all, what we see is that this control plane management is more or less what is driven completely by what the vendors would provide to us, and has become an impasse, say in terms of what we can do the innovations here. And this is something that we also need to take into account and ensure that we are able to overcome this to provide better innovations be done on this control plane.

(Refer Slide Time: 6:40)



The Control Plane Problem

- Control plane must compute forwarding state. To accomplish its task, the control plane must:
  1. Figure out what network looks like (topology)
  2. Figure out how to accomplish goal on given topology
  3. Tell the swtiches what to do (configure forwarding state)
- We view this as a natural set of requirements....
  - And we require each new protocol to solve all three

Software-Defined Networks Advanced Computer Networks

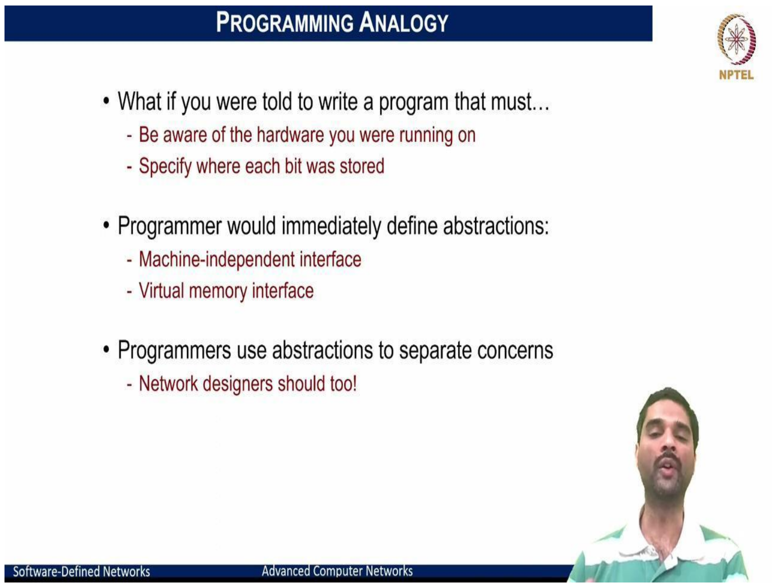
And to do that, we need to look at what exactly are the control plane aspects that we need to be thinking of to provide the right abstractions. And we know that the control plane must compute the forwarding state. And to accomplish this task, the control plane would have to figure out what the network looks like, that is, what is the underlying topology, what are the links and the interacting neighboring devices, and overall the devices that are in the current topology, so that we can come up with the forwarding state.

Second, once they would identify the key aspects of how the topology looks like, then they would have to go and run the algorithms that would enable them to build the routings in a way that we want the weights to be accounted for be it the weights in the inverse model to say that the weights correspond to either the delay or in a direct proportion to say whether the weights correspond to the bandwidth and then actually affect the routing intentions that we want to carry out.

And eventually, once we have figured out a particular route or a path, then we would also need the mechanism to tell the switches or routers to exactly update the corresponding forwarding rule, which would then make sure that the packets would go and take that particular role in getting forwarded.

So, we need the right abstractions at all of these layers. And these would be the natural requirements for us in terms of how we want to build the control plane aspects when it comes to managing the forwarding state. And if we have to do this, we need to look at what are the wanted aspects. Here, are the three that we outlined would be the most desired aspects when we want to set up a forwarding plane, and when we build this, any of the aspects that are unlikely for us to be able to manage could be left out at the device-specific as properties.

(Refer Slide Time: 9:01)



Programming Analogy

- What if you were told to write a program that must…
  - Be aware of the hardware you were running on
  - Specify where each bit was stored
- Programmer would immediately define abstractions:
  - Machine-independent interface
  - Virtual memory interface
- Programmers use abstractions to separate concerns
  - Network designers should too!

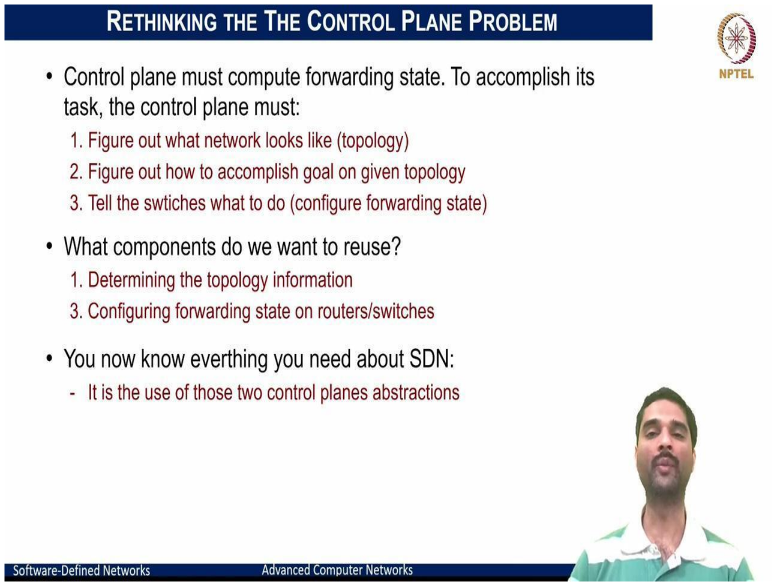
Software-Defined Networks Advanced Computer Networks

So, consider this analogy that we would build in programming. And what I mean by this is suppose you were told to write a program wherein you have some specific part of the program that is hardware dependent, and maybe some of the information that you would want to store is dependent on certain hardware characteristics.

And if we are given such a programming problem, a natural thing for us as programmers is to think of the right abstractions that we would want to build and we would want to modularize our code and try to build the interfaces in such a way that we abstract out the machine dependent characteristics from the machine-independent part and handle the machine specific concerns in separate internal modules. This way, we would be able to reuse the machine-independent modules across different machines or different hardware. While we may only have to rebuild the machine-specific characteristics to suit the underlying hardware requirements, such as being the case, when we look at most of the operating systems that we use today be it the Linux, the windows, etc. And that is where this abstraction enables much of the reuse and is easy to facilitate and suit any kind of hardware.

And now we can think of applying similar aspects to the networks as well. And what it means is, we need to rethink of the control plane mechanisms that we discussed before, and try to come up with the right abstractions, which would make them work across multiple devices.

(Refer Slide Time: 10:51)



Rethinking the The Control Plane Problem

- Control plane must compute forwarding state. To accomplish its task, the control plane must:
  1. Figure out what network looks like (topology)
  2. Figure out how to accomplish goal on given topology
  3. Tell the swtiches what to do (configure forwarding state)
- What components do we want to reuse?
  1. Determining the topology information
  3. Configuring forwarding state on routers/switches
- You now know everthing you need about SDN:
  - It is the use of those two control planes abstractions

Software-Defined Networks   Advanced Computer Networks

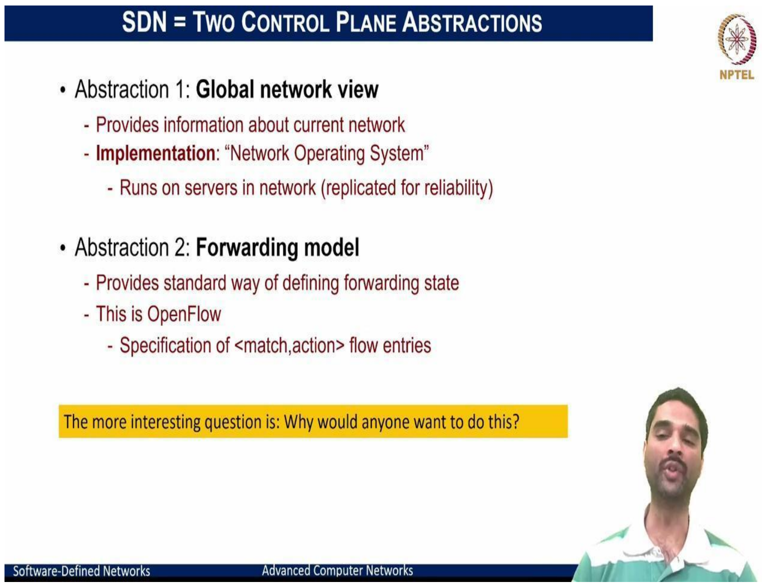
In this, we re-think of the three key characteristics that I outlined, we need to ensure that we have the right abstractions; where if we have to figure out what the network looks like, or what is the topology like, we should be able to reuse this information in providing a multiple of the control plane functions, not just to ensure that it is applicable only to set up the rules for forwarding. But it could be used for various other purposes.

Second, if we write the set of interfaces, which we would see how to configure the forwarding state in a very generalized fashion, then this can be adapted to any of the hardware to suit how the actual configuration of the forwarding would be done. Think of this as an instruction set architecture that we use like whether it is a RISC or x86 machine that we most commonly use.

Once we define the instruction set, these can be adapted, and the underlying microarchitecture at that time, or the computer organization, really does not matter that would operate the way the hardware would try to use, but the interface remains fixed. And the same, we can now think of the forwarding state; if we define the right set of states or the interface to configure the forwarding state, we can take it and apply it to any of the hardware devices. This way, we can see that we can establish that the topology information could be reused.

And the way that the forwarding states can be applied as a machine-independent characteristic would become very handy to apply it across different routers or switches. And this is all the SDN as a control plane abstraction is all about in terms of trying to build a better model of abstractions to the control plane.

(Refer Slide Time: 12:56)



SDN = TWO CONTROL PLANE ABSTRACTIONS

- Abstraction 1: **Global network view**
  - Provides information about current network
  - **Implementation**: "Network Operating System"
    - Runs on servers in network (replicated for reliability)
- Abstraction 2: **Forwarding model**
  - Provides standard way of defining forwarding state
  - This is OpenFlow
    - Specification of <match,action> flow entries

The more interesting question is: Why would anyone want to do this?

So, if we think of the SDN control plane, we can think of two specific abstractions that we would want to build here. The first is the global network view. That is, we want to provide the information about the current network. And this, when we think of an implementation, we can term this as the network operating system wherein it provides exactly the information of the underlying network and abstracts out what each element is all about, allow the user to use the networking elements to set the intention on the networking element without worrying about how and where aspects of the networking elements.

And when we think of this network operating system as a rich abstraction, this needs to be run somewhere and that is we can think of as running on one specific dedicated server, which could be just as a computer PC that we use or the server machines. And for reliability purposes, we may even think of having this server replicated across multiple machines, or you can think of this network operating system as a distributed platform where a user would see and interact with such a system at one point, but it may be physically replicated at multiple ends.

And the second most fundamental abstraction that we would want is the forwarding model or what we can call as a generalized forwarding model, which would provide a standard way of defining the forwarding state, and such a way of defining a forwarding state is what we learn as the OpenFlow. And once these specifications of the flows are defined, and in this, it is OpenFlow paradigm that is called as a <match, action> pattern where you define the set of constraints on which you would want to match the data plane packets and pick out the set of headers and try to match the key characteristics from there. And for a given match, you would want to take a specific set of actions. And the actions here is basically what to forward or transform the packet, modify the packet headers, or even to drop the packets as well.

So, in this model of the generalized forwarding with this kind of specification of <match, action> on the flow entries, note that we are moving away from the destination IP-based forwarding that occurs within routing devices. And this gives us a lot more flexibility and a lot more control over how I would want to forward the packets.

So, having established these control plane abstractions and having enhanced our SDN model of this separation beyond the separation of control and data plane building the right abstractions at the control plane. Now, we would have to reconsider our earlier question of why we would want to do this?