


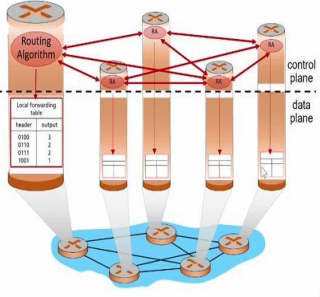
**Advanced Computer Networks**  
**Dr Sameer Kulkarni**  
**Department of Computer Science Engineering**  
**Indian Institute of Technology, Gandhinagar**  
**Lecture 33**  
**Software Defined Networking – Part 2**


(Refer Slide Time: 0:17)

### TRADITIONAL NETWORK ROUTING

- Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables.
- Internet-network (L3) layer:** historically has been implemented via **distributed, per-router** approach
- monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
- Destination based forwarding* – makes it impossible to distinguish and apply different forwarding modes for different flows.
- different "middleboxes"* for different network layer functions: firewalls, load balancers, NAT boxes, etc.
- ~2005: renewed interest in rethinking network control plane







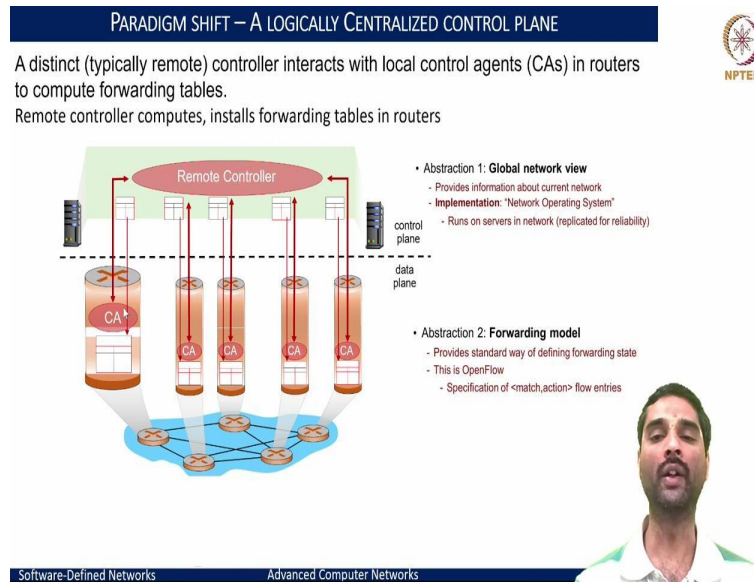
Software-Defined Networks
Advanced Computer Networks

Let us now look into how the control plane operates to build the forwarding tables. Consider the topology shown here. And on top of this, now if the control plane has to interact and build the routing information, we need to make these routers that are spread across the network to interact and build the information. And this is done by facilitating each of the routers with a specific routing algorithm that would run in them. And this could be either the link state routing algorithms that are based on the entire flooding info or this flooding infrastructure, or it could be a distance vector that will just exchange information with the neighbor.

Eventually, when they converge, each of these routers will have the information of the network topology, and then they can run the specific routing algorithm and basically reduce and deduce what would be the forwarding tables in order to reach from one point to the other. And this way, the routing algorithms in each of the routers would set up their own local forwarding tables. So note the interaction happens at the control plane for deducing the network topology information. And the routing algorithm would be run on each of the routers to provide the local forwarding table.

Now, let us try to fit the two of the abstractions that we just discussed into this model, and how we would have that paradigm shift?

(Refer Slide Time: 2:06)



Here, if we take the same base model, what we observed is that the control plane is all embedded within each of these devices. And hence the routing algorithms are all run locally on each of the devices. And our first abstraction, in this case, was to provide a global network view. That means we did not want each of the routers to have a network view that is global but local to itself, but a view that is common across all of these routers. And to do this, what we would have to build is ensure that we have the information that is extracted besides these indistinct individual routers.

And that way, if we shift the control plane up, that is, externalize the control to a remote controller, which is now able to map and spawn other requirements to build the global view, then we would have achieved our first abstraction. And what that means is to take out the routing algorithm that is running within each of these routers and externalize it to a separate server machine that is running outside. And this what we call as a remote controller, is our implementation of a network operating system.

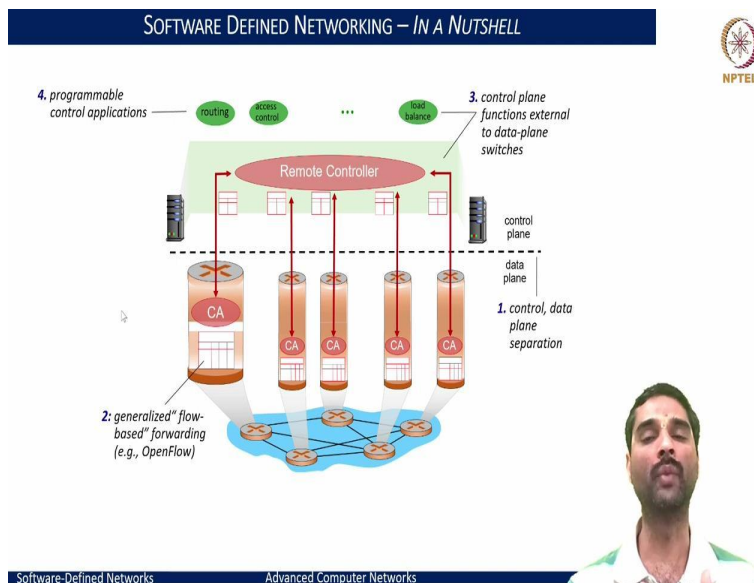
And then, to speak of our means, once you have a remote controller that is able to get the global network view, it can compute the global routing table. Note that this routing table reflects the Global Map independent of each of these routers. At this point, now, if we have to plumb the

forwarding table information back, we need a mechanism to put the information that is derived at the remote controller to be updated at each of the routers. And for that, we use these the controller agents to which the remote controller can communicate and ensure that it can push the corresponding forwarding table.

So, this way, now we can fit when we want to build this communication. We want to build a generalized forwarding model that is exactly our abstraction 2. So, that whether one of the routers is a Huawei router, the other is a Cisco router, or Arista router, it does not matter. What matters is we have a unified API to push the forwarding table into each of the routers. And that is exactly our abstraction 2 facilitated through a control agent that would run on each of these routers and ensure that this control agent will operate on a unified API model from a remote controller.

And then would do target specific aspects to eventually build the forwarding table into the range of routers.

(Refer Slide Time: 5:13)



And thus, we would have achieved our goal of decoupling the aspects of control and data to ensure that we are able to control the entire network homogeneously. So, to sum up, the software-defined networks in a nutshell, what it would look is, the first is the separation of the control and data plane, where the control plane is basically externalized. And through this

externalization of the control plane from the devices, these routers, and switches, we have a chance to build a unified interface for operations.

And what this also allows is, means to have a unified interface to configure each of these networking elements. And this is achieved through generalized flow-based forwarding. And that is, for example, OpenFlow API, which will allow us to set the match action tables into each of these routers. And from the remote controller point of view, having externalized the control plane, all the control plane functions can now be run on one single server. You do not have to worry about the hardware aspects of these individual routers from which OEM the routers are coming from, all of these can now get abstracted and built on top of the remote controller in a common fashion.


And this is our third aspect of control plane functions, now becoming external to the data plane switches, which means they can be run on the commodity hardware now. And more importantly, now, because we have the control functions that can be run on our regular machines, the control applications become programmable. In the sense that every individual can now program and control the way, the routers would want to behave and dictate much more easily what objectives need to be achieved.

So through this simple decoupling of the control and data plane and centralization of the remote control, we are able to now see a paradigm shift in terms of how the networks can be operated, and networks can be managed in a much simpler and more centralized way. And that is where the SDN started to gain attraction and became very popular.

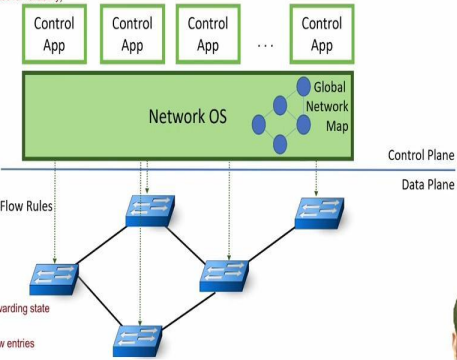
(Refer Slide Time: 7:59)

## SDN CONTROL AND DATA PLANES


- 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



Software-Defined Networks
Advanced Computer Networks

And to fit it now, in very simple terms, the centralized remote controller, the abstractions that it provides can be thought of as a network OS, and it could build the global network map and allow the control applications to flourish on top of this network OS which provides the common abstractions. At the bottom, we have the bare-bone switches, where a control agent would be able to run and establish the forwarding model of providing the simple match action patterns to be run on these data plane elements.

In essence, the network elements like the routers and switches now truly begin to behave as data plane elements while the control is taken out. Nonetheless, it does not make these devices any dumb. The misconception also has been that the SDN makes these devices dumb is not true. In a way, it simplifies these devices and ensures that now you can build the means to program any of these devices with a lot more ease and reuse the same code that you would build on top of the network OS for a variety of hardware manufacturers or hardware vendors. And that is basically the openness and innovation that SDN tries to bring forward.

(Refer Slide Time: 9:25)



Why a *logically centralized* control plane (Network OS) ?

- Programmability – Full control and visibility over the network.
- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based generalized forwarding allows “programming” the routers
  - *centralized* “programming” easier: compute tables centrally and distribute
  - *distributed* “programming”: more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- open (non-proprietary) implementation of control plane
  - Foster openness & innovation!



So, we may wonder why we should have a logically centralized control plane or a network OS? We said with a logically centralized control plane or common network OS the applications can flourish very easily because you have a fixed API's to worry about. And that means the programmability is very easily applicable over these network OS than on the dedicated hardware-software combinations that OEMs used to ship. So, this would give full control and visibility over the entire network.

Second, it becomes a lot more easier to manage the network because we are now trying to manage and configure the rules from just a single point, not that we are avoiding the configurations on multiple routers because we still have to configure the rules on each of the routers. But the decisions are taken centrally rather than the decisions earlier that used to be taken by each of the routers independently, which made it very difficult to debug and know what which router has chosen, what paths, and for what purpose.

But now we have complete control. And that makes the configuration a lot more easier, and we could avoid any of the router misconfigurations that otherwise would happen. And we can even get rid of the count to infinity problems that would have otherwise been seen in earlier models. And this also provides greater flexibility; why? Because now the forwarding tables can be built on each of the routers independently, we can base this not just on the destination IP or the destination-based forwarding but make use of the generalized forwarding where we can take into account the characteristics of even the source IP. So, based on the source and destination IP

combinations, based on the port or the protocol combinations, we can choose to have different paths that can be used for the same set of flows originating from the same source and ending at the same destination. So, this gives us a lot of greater flexibility for us to manage the traffic flows.

And most importantly, now when we look at the second abstraction that we set the table based generalized forwarding. In essence, this is exactly the programming of the routers in terms of what the data plane entity of the router is supposed to do. So, the flexibility of how we want to program is offered at a single point at a controller, and that is centralized programming.

So, it becomes a lot more easier to compute these match action tables and distribute these tables across all the routers that are participating in the network. Unlike the earlier distributed programming, where it was a lot more difficult, and you had no clue about what each router has eventually ended up. And there was no control either on how to choose the path because that was all embedded within the routing devices themselves.

And overall, such a framework leads to openness; that is, no proprietary aspects could hinder us from implementing anything new as we have the same open interfaces on which we can build the networking control applications, as well as experiment on how to route and forward the packets at the data plane. So, this automatically fosters openness and innovation. And that is where SDN made the mark and has been revolutionizing the networks, the way we deploy, and the way we think of it.

(Refer Slide Time: 13:14)



- Centralizing the control plane enables more powerful abstractions
  - E.g. X and Y should be able to communicate
  - Express intent network-wide
- Central control means a single API for the network, rather than an API per box
- Distributed systems techniques to make central control scalable and fault tolerant
- Networks provisioned by software, not humans
  - Disaggregation → innovation
  - Network-wide intent → better control and better security



So, to take out some of the specific insights here, with SDN, it all began with decentralizing of the control plane. And it is a very powerful abstraction, because now anyone who would want to communicate needs to just express the intent of how they would want to communicate and the rest of the complexities of making sure that forwarding tables, everything can be done, would be done through a programmable way at centralized controller. And this centralized controller also means that it is a single API for the entire network, rather than thinking of API for each box, where you would go to the routers and configure the parameters, which will differ from different hardware vendors.

Now, no more of that hassle, and you can control all of these devices sitting at just one point. Now, the question may arise like when we have such a framework, would not the centralized controller or the network operating system, the machine on which NOS is running be a bottleneck, and how would it really scale. And that is where we bring the notion of a logically centralized, but meaning you have one view of a system, but it does not necessarily mean that it is just one machine on which the network operating system is running.

So, we can, in fact, leverage the distributed systems techniques that is to build a distributed framework where the network operating system is a distributed OS running and replicated on several of the servers, which can basically coordinate and interact to facilitate the updates onto the hardware plane. Thus we can have centralized control, no matter on which of the device you would work, you would have a unified view. And we will also get rid of a single-point failure.



That is, even if one machine fails, one of the controller devices fails, you would have other controllers to work operate on and continue the operations.

And as well, we could load balance across these multiple controllers as we scale the underlying network. Thus, these networks can now be provisioned by software. And you do not have to spend human resources trying to provision and spend hours of time in getting the network set up. Going forward from the decoupling of the control and hardware, software control, and data plane, the SDN evolved a lot more towards bringing what we call a disaggregation. And this disaggregation between the hardware and software of the networks, how we manage and control.

And on the other end at the application space, they have made the network operators' life lot more simple by enabling the operators to just specify the intent rather than worrying about the integrities of how to make a specific policy; we update it or worry about the mechanisms that made the policy to be implemented. And this provides in a way that better control and better security to the network operators.

(Refer Slide Time: 16:29)


**EVOLUTION: SDN ⇒ DISAGGREGATION**

- SDN was invented in 2009. Then: SDN:
  - Separation of control and data planes
  - Centralization of Control
  - Standard Protocol between the planes
  
- Now: Software Defined = **Disaggregation** of HW/SW
  - Commodity hardware
  - Software on commodity HW
  - Legacy protocols survive

```
graph TD
    subgraph SDN_2009 [SDN 2009]
        CP[Control Plane] -- OpenFlow --> DP1[Data Plane]
        CP -- OpenFlow --> DP2[Data Plane]
        CP -- OpenFlow --> DP3[Data Plane]
    end
    subgraph Disaggregation [Disaggregation]
        O[Orchestrator] --> SW1[SW]
        O --> HW1[HW]
        O --> SW2[SW]
        O --> HW2[HW]
    end
```

Ref: D. M. Batista, G. Blair, F. Kon, R. Boutaba, D. Hutchison, R. Jain, R. Ramjee, C. Rothenberg, "Perspectives on software-defined networks: interviews with five leading scientists from the networking community" Journal of Internet Services and Applications 2015, 6:22, <http://www.csc.wustl.edu/~jain/papers/jis15.htm>  
J. Skorpupa and D. Cisco, "State of SDN: If You Think SDN is the Answer, You're Asking the Wrong Questions," Gartner Report G00325601, 24 August 2017, 9 pp.  
Washington University in St. Louis [http://www.csc.wustl.edu/~jain/talks/fosec\\_jain.htm](http://www.csc.wustl.edu/~jain/talks/fosec_jain.htm) ©2019 K

Software-Defined Networks      Advanced Computer Networks



So, if we look into the evolution of SDN, we started and learned the journey up till here to say that we separate the control and data planes. And we would centralize the control into one that is logical, centralized network operating system. And once we do that, we also need the standard protocols to operate between these planes. That is the control plane and data plane. So, we look

into next about the OpenFlow. And with the evolution, what we see of SDN is not just the decoupling anymore, like I said, this is to do a lot more with disaggregation or in fact, you virtualize the way the hardware and software interplay happen for the networking devices.

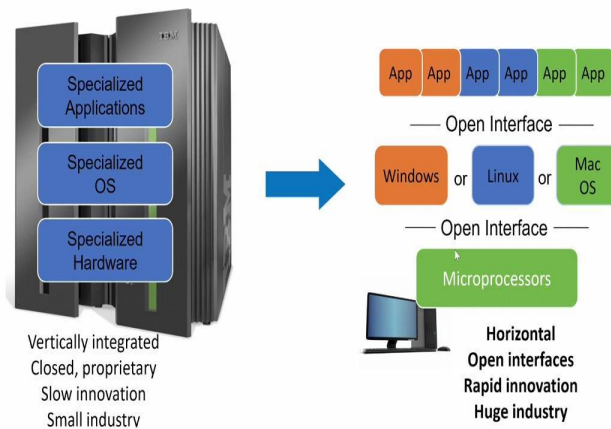
What this really means is that software-defined is now a disaggregation of hardware and software. So, think of the control plane, the logical controller on the network operating system that we built as an orchestrator that is able to manage multiple of these networking devices. And each of these networking devices, we would have the control agent that would run and facilitate configurations. And we would have the hardware underlying hardware, which will dictate what hardware-specific aspects we would want to do.

Taking this a bit forward, if we have to say that we can disaggregate this into rather a commodity hardware itself, that is just as how we use our PCs, which are the commodity or off the shelf hardware, we could use and build these routers on top of our regular laptops or personal computers and make use of hardware and software to build them. And in a way, we can have full control over what hardware we can use and what software layer that we would want to push.

And this entire SDN framework would allow us to build these entire networking elements at a very low cost and in an effective manner. Nonetheless, it would not mean that we are going completely away from the legacy protocols, the legacy protocols could still survive. And we could have the combination of hardware, proprietary hardware or proprietary software to make them operate. But we would have the generalized interface for orchestrating on each of these devices.

(Refer Slide Time: 18:58)

## DISAGGREGATION OF COMPUTING INDUSTRY: MAINFRAME TO PERSONAL COMPUTERS



Software-Defined Networks

Advanced Computer Networks

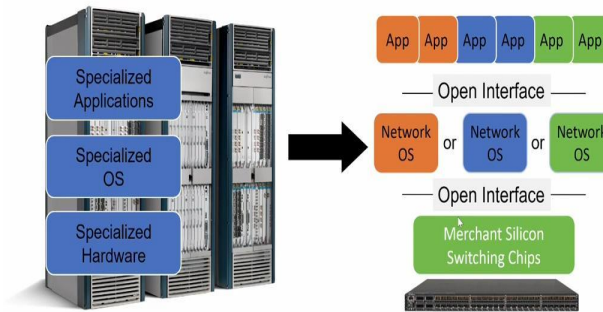
So, to give a good analogy of what it really means, if we think of the evolution in the personal computers, how this all began the journey in the early 1960s. We had these mainframe devices, which were the mammoth devices that would almost fit half the room's size. But the main aspect there was they had specialized hardware. And they would be operated with specialized operating systems, and only specialized applications would be run. And there was no scope for customization or enabling any user to try out different things. Whatever was shipped if you had to take that for your needs. And this changed over time.

And instead of having these vertically integrated or closed systems, which were not open to innovation. And because of that, they were also deployed on very small scales. As the processor industry developed, and we started with the era of microprocessors, which defined the open interfaces in terms of the instruction sets, like x86 as an ISA. Now, which could be used and built multiple of operating systems that would work on top of these processors, like Windows, Linux, Mac OS, or any of those.

And these operating systems, in essence, provided the open interface for the applications to flourish. And that is what led to the massive growth of personal computers. So, these open interfaces are crucial for evolution and innovation. And the same holds good now for the networking industry. Now with SDN, what we are trying to achieve is these open interfaces, which enables us to have rapid innovations as well.

(Refer Slide Time: 21:00)

## SDN = DISAGGREGATION OF NETWORKING INDUSTRY (DEDICATED TO COMMODITY)



Software-Defined Networks      Advanced Computer Networks



So, to bring the analogy here, what we had were dedicated switches and routers, where they were specialized hardware built by specific OEMs or vendors, and they had their own operating systems. And they would also have tailored applications that will be running on those dedicated hardware. Now with SDN, what we have moved is from this dedicated hardware, to the merchant silicon switching chips, the switching chips are the commodity hardware, like for example, this could as well be just our personal computers.

And now, we can run on top of these personal computers, the network operating systems. And once we have these network OS that provides these open interfaces, we could build readily any of the applications at the top to utilize and run the required characteristics on these networking devices. And the open interface at the bottom would also mean that these commodity hardwares can now be programmed for the way that we would want the routers and switches to behave.

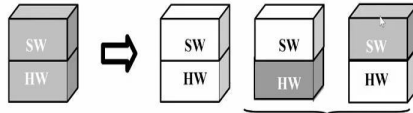
And thus, this disaggregation of the network industry has had a wide impact on the evolution of the 5G networks, the telecommunication networks, and the data center networks. And in fact, many of the campuses have started to deploy the SDN architecture for building the campus and enterprise networks.

(Refer Slide Time: 22:31)

## DISAGGREGATION: BLACK BOX TO WHITE BOX



- Differentiation via software ⇒ White box networking
- **Black Box:** Proprietary HW with Proprietary SW
- **White Box:** Open Source Hardware and Software
- Software on a different hardware
  - ⇒ hardware can change Different software on a hardware
  - ⇒ Software can change
- **Bright Box:** Branded White box =  
Branded SW on open HW or Open SW on Branded HW



Ref: A. Lemer, "Branded Switching + White-Box Switching = Brice-Box Switching," Nov 14, 2014, <https://blogs.gartner.com/andrew-lemer/2014/11/19/bricefuture/>

Washington University in St. Louis

[http://www.cse.wustl.edu/~jain/talks/ntac\\_jain.htm](http://www.cse.wustl.edu/~jain/talks/ntac_jain.htm)

©2019

Software-Defined Networks

Advanced Computer Networks



So, this is a slide from Professor Raj Jain's presentation. But I want to emphasize on how this disaggregation really moves us from a black box to a white box model. And what this really says is earlier, we had everything proprietary, and that was exactly the hardware and software, which were completely as a black box. And now, with the disaggregation that SDN brings to the table, we can have a complete white box model where the hardware is a commodity hardware and the software is a commodity network operating system that we would want to run.

But it is not just confined to having a complete white box approach. We could have still the hardware manufacturers from different OEMs like Arista, Cisco, or whoever they be, take Intel, and all of those can still deliver the custom hardware. But on top of these custom hardware, it is now because of SDN we are able to write the control agents for each of these onto the software and keep the software on top of which the applications are going to be written. That is the network operating system, the same that we have disaggregated the hardware from the software and enabled these open software implementations to run on these custom hardware.

And the other aspect, it could be an open hardware, but we wanted to experiment and run custom software that are developed by these OEMs. Even that is possible now because if we can add and document the software to utilize the open interfaces that were being built, then we would have a framework to test out those proprietary software's also on this open hardware. And this is what was branded as the branded white box or the bright box model.

So, this SDN, now with a disaggregation pattern, is able to facilitate many mechanisms of how to run the network elements with proprietary or commodity hardware and software stacks.

(Refer Slide Time: 24:44)

**A REVOLUTION IN NETWORKING**

**IO BREAKTHROUGH TECHNOLOGIES** 2009

Link: <https://www.technologyreview.com/technology/tr10-software-defined-networking/>

TR10: Software-Defined Networking

Nick McKeown believes that remotely controlling network hardware with software can bring the internet up to speed.

**KATE GREENE**  
Thursday, February 24, 2009

For years, computer scientists have dreamed up ways to improve networks: speed, reliability, energy efficiency, and security. But their schemes have generally remained lab projects, because it's been impossible to test them on a large enough scale to see if they work. The routers and switches at the core of the internet are locked down, their software the intellectual property of companies such as Cisco and Hewlett-Packard.

Frustrated by this inability to fiddle with internet routing in the real world, Stanford computer scientist Nick McKeown and colleagues developed a standard called OpenFlow that essentially opens up the internet to researchers, allowing them to define data flow using software—a sort of “software-defined networking.” Installing a small piece of OpenFlow firmware (software embedded in hardware) gives engineers access to flow tables, rules that tell switches and routers how to direct network traffic. That protects the proprietary routing instructions that differentiate one company's hardware from another.

With OpenFlow installed on routers and switches, researchers can use software on their

Software-Defined Networks

So, to think of what we have seen is SDN bringing about a revolution in networking and I want to point to this article from the MIT Technology Review, which was published almost in February 2009, when Professor Nick McKeown made the statement about the SDN, and what it really meant, it is just a one-page article here, I have also given the link for the interested readers to go and take a look. And this beautifully summarizes in terms of the challenges that then the network industry faced, and what are the means and ways through which the software-defined notion can be brought in, and how this would even break the ossification or the impasse that we spoke of, and open up for innovations in a much more ready manner.

(Refer Slide Time: 25:39)

## HISTORY AND EVOLUTION OF SDN



~2004: Research on new management paradigms

RCP, 4D [Princeton, CMU,....]  
SANE, Ethane [Stanford/Berkeley]

2008: Software-Defined Networking (SDN)

NOX Network Operating System [Nicira]  
OpenFlow switch interface [Stanford/Nicira]

2011: Open Networking Foundation (~69 members)

**Board:** Google, Yahoo, Verizon, DT, Microsoft, Facebook, NTT  
**Members:** Cisco, Juniper, HP, Dell, Broadcom, IBM,.....

2022: Latest Open Networking Summit (Nov 15-18, 2022, Seattle)

**In 2020:** ~1300 attendees, ~71 countries Emphasis on 5G, SDN and NFV



Software-Defined Networks

Advanced Computer Networks

And to summarize, we have looked at the SDN in terms of how it all started with the RCP and 4D works. And we started to look into the emergence of SDN, starting from the early 2008, which it meant to just centralize the controller and logically decouple the control and data planes. And now, to the stage where we are even thinking of disaggregation of the hardware and software stacks to build Open Networking elements.

And this SDN, it all started with Nicira as a company that started these OpenFlow standards, which eventually got grown up and taken up by the IETF standards, and then the SDN working group started in the IETF. And then eventually, it became a member of the Open Networking Foundation, which has many of the board members that we all know of Google, Yahoo, Verizon, Cisco, Juniper all of them. And now, this community in itself, the Open Networking Foundation community, which started in around 2011, has grown bounds and leaps.

And in around 2022, there was a recent networking summit that happened at the very end of the last year. And the attendees were across from several countries, several of the interests were shown, and now we are seeing this SDN to have a greater emphasis and shape on the 5G and the NFV. The other tangential thing that will later go into discussing the work, and this all has happened because of the way the visions were set for SDN.