

Advanced Computer Networks
Doctor Sameer Kulkarni
Department of Computer Science Engineering
Indian Institute of Technology, Gandhinagar
Lecture 28
Road to SDN

In the last week, we looked at how network virtualization and overlay networks helped overcome the internet impasse, and enable the researchers to continue experimentations over the virtual test beds. This week, we will continue and see what other developments happened towards network diversification. And to facilitate better control over the network.

(Refer Slide Time: 00:39)

ADVANCED COMPUTER NETWORKS, OUR LEARNING JOURNEY

Traditional Networking: ~~Basics of communications;~~ Principles of networking, Network design philosophy, networking stack, What is E2E? Issues with the Networking Principles and architecture. Internet Impasse! and Ossification.

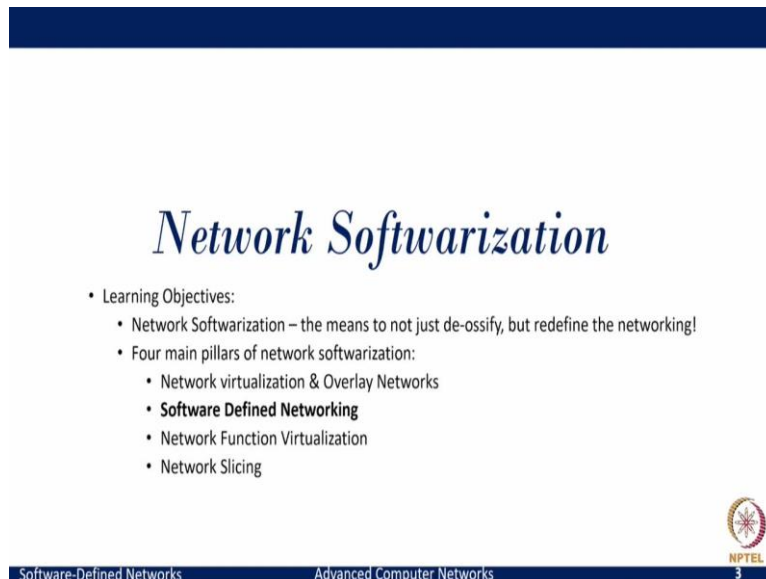
Modern day Networking: ~~Network Virtualization,~~ **Software Defined Networking**, Network Function Virtualization and use cases.

Future Networks: Softwarized & Programmable Networks: P4-Programming, Protocol-Independent Packet Processors, In-band network telemetry, Green & Sustainable Data Centers, Serverless Computing, ~~Zero-trust and Blockchain.~~



Our primary focus for this week would be Software Defined Networking, or more popularly known as SDN in short.

(Refer Slide Time: 00:48)



Network Softwarization


- Learning Objectives:
 - Network Softwarization – the means to not just de-ossify, but redefine the networking!
 - Four main pillars of network softwarization:
 - Network virtualization & Overlay Networks
 - **Software Defined Networking**
 - Network Function Virtualization
 - Network Slicing

Software-Defined Networks Advanced Computer Networks

NPTEL 3

And this SDN when we say it refers to a networking architecture, that virtualizes the network, where the network control functions are decoupled from the network forwarding functions, and also enable the network control to be made directly programmable. We will look at these two aspects in detail.

(Refer Slide Time: 01:17)



"There is a tendency in our field to believe that everything we currently use is a paragon of engineering, rather than a snapshot of our understanding at the time.

We build great myths of spin about how what we have done is the only way to do it to the point that our universities now teach the flaws to students (and professors and textbook authors) who don't know better."

John Day (Internet pioneer)

Software-Defined Networks Advanced Computer Networks

NPTEL 4


And before that, let us try to see what actually happened and what were the points are reasoning's that were made for this SDN. And I often point to this famous quote by John Day. And this seemingly a bold but a courteously a very true statement, wherein he pointed out that we often believe the things to be the paragon of engineering at a given point, rather

than understanding that they are the snapshots of our understanding at that specific point in time, which is also true when we look at internet as a marvel of engineering.

And this often leads to how we try to advocate about the merits and demerits likewise. But often, what we end up doing is profits on specific aspects than re enquiring and looking at the other aspects. And this is what John Day pointed out rightly, towards what happened with the Internet. Nonetheless, we do need to understand that learning is a continuous process. And progressive evolution is a natural phase, only when we continue to reason and inquire a good habit.

(Refer Slide Time: 02:38)

READ THIS INTERESTING AND INSIGHTFUL PAPER!





The Road to SDN: An Intellectual History of Programmable Networks

Nick Feamster
Georgia Tech
feamster@cc.gatech.edu

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

Ellen Zegura
Georgia Tech
ez@cc.gatech.edu





ABSTRACT

Software Defined Networking (SDN) is an exciting technology that enables innovation in how we design and manage networks. Although this technology seems to have appeared suddenly, SDN is part of a long history of efforts to make computer networks more programmable. In this paper, we trace the intellectual history of programmable networks, including active networks, early efforts to separate the control and data plane, and more recent work on OpenFlow and network operating systems. We highlight key concepts, as well as the technology pushes and application pulls that spurred each innovation. Along the way, we debunk common myths and misconceptions about the technologies and clarify the relationship between SDN and related technologies such as network virtualization.

1. Introduction


Computer networks are complex and difficult to manage. These networks have many kinds of equipment, from routers and switches to middleboxes such as firewalls, network address translators, server load balancers, and intrusion detection systems. Routers and switches run complex, distributed control software that is typically closed

network's data plane elements (i.e., routers, switches, and other middleboxes) via a well-defined Application Programming Interface (API). OpenFlow [51] is a prominent example of such an API. An OpenFlow switch has one or more tables of packet-handling rules. Each rule matches a subset of traffic and performs certain actions on the traffic that matches a rule; actions include dropping, forwarding, or flooding. Depending on the rules installed by a controller application, an OpenFlow switch can behave like a router, switch, firewall, network address translator, or something in between.

Over the past few years, SDN has gained significant traction in industry. Many commercial switches support the OpenFlow API. Initial vendors that supported OpenFlow included HP, NEC, and Protonic; this list has since expanded dramatically. Many different controller platforms have emerged [23, 28, 37, 46, 55, 63, 80]. Programmers have used these platforms to create many applications, such as dynamic access control [16, 53], server load balancing [39, 81], network virtualization [54, 67], energy-efficient networking [42], and seamless virtual-machine migration and user mobility [24]. Early commercial successes, such as Google's wide-area traffic-management system [44] and Nicira's Network Virtualization Plat-

Software-Defined Networks

Advanced Computer Networks


NPTel
5

So, let us begin our journey on SDN and try to briefly look at one of the most cited papers. And this paper was actually almost a decade ago published a decade ago. And that clearly illustrates what things happened in breaking out this SDN. And in fact, this paper traces the intellectual history of programmable networks, including the very early efforts of active networking, and the efforts to separate the control and data planes, including the works on open flow and network operating system. So, I will try to summarize the key aspects that are discussed in this work in a very brief manner.

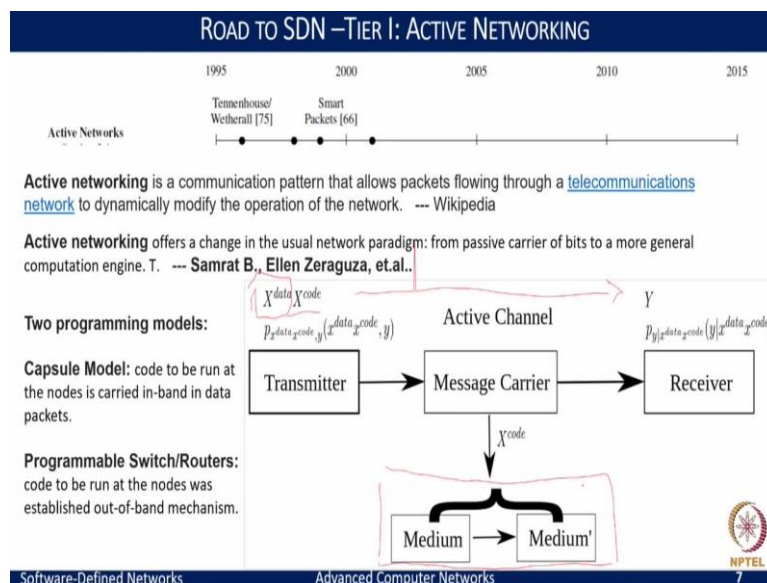
(Refer Slide Time: 03:21)

ROAD TO SDN – THREE TIERS OF NETWORK EVOLUTION

1. **Active Networking:** Notion of Programmable Networks
2. **Separation of Control & Data Planes (Software Defined Networking):** Disaggregation with Open Interfaces.
3. **Network Operating Systems:** Widespread adoption of Open Interfaces and the evolution of Network controllers with OpenFlow APIs.

So, we can think of this paper as three specific aspects. First, is the active networking, which in fact, was the first of its kind to introduce the notion of programmable networks. So, let us look at active networking, and then jumping to the other two aspects of separation of control and data planes and the emergence of network operating system.

(Refer Slide Time: 03:44)



So, the first, as you see the active networks, were the first of the attempts that were made to make the networks programmable. And this even the advancements of the processors, or the processor technologies were in the significant reduction in the computation costs actually propelled for the research, active networks.

In a sense, what active networking represented is a radical approach to network control. By envisioning a programming interface or a networking API's that are defined and exposed. For example, the processing, the storage, the packet queues for each of the functions that happen within a network, the API's be exposed on each of the individual network nodes and be supported in terms of a custom functionality that can be modified by the incoming packets or a subset of the packets that traverse through the networks can modify these functionalities within the network.

So, now you cannot think as packets as just as passive data that is being transmitted from one end of the device to the other. But also, the data contains the active part, which can manipulate and process and change the way the network behaved for processing of these packets. And that is what really led to active networking.

In a very nutshell definition. It is like a communication pattern that allows the packets flowing through a telecommunications network to dynamically modify the operations of the network itself. And to understand it more prominently, let us try to see what actually active networking offers?

It is basically a networking paradigm where the bits that are carried are in more general constitute compute and data just as what we do a program, program contains a set of instructions that enable you to operate on the data and manipulate the data. Likewise, now the packets that are transmitted contain the code, which can be operated over the networking elements themselves, to instruct or to modify the existing connections.

So, if we see this figure here, as a transmitter, you want the data X to be transmitted, which constitutes of both the data that you eventually want the receiver to receive, say, or Y or not necessarily the same data as what you transmit, but a function of that data that will eventually result in the receiver receiving the data Y on the other side. And to make this happen, you also add the code that would operate on this particular data X within the network. And where this manipulation or the function when it is run would actually translate this X of the data into Y which the receiver needs to receive.

And at the same time, if there is a code that needs to be operated within a medium and make sure that some properties of the medium be changed from the initial state of medium to medium prime, then this X code would facilitate to make this channel operate and bit. Let us

think of a very simple and an example of how the NAT operate. And what happens when we try to do the NAT, you need to have the private and public IP matchings that are being set.

Now, this allows anyone from inside the network to communicate outside to a public IP. But what about the other way? We know that we cannot reach back to a private IP unless we know that there is a mapping that is being cited on that device preset that is a rule that exists to say that what would be the public IP and port to which it can translate back to the private IP and private port.

And this is not possible without the usage of this turn or turn aspects very hard to write the traversal mechanisms and say that you can reach the device that is within a private network. Now, if we are able to embed the code and data and ensure that as the packet traverses, it can set up this functionality of mapping within an ad device so that now we are able to access and reach the other side of the private network, or job is done.

So, that way the active channel is actually building the rules or mappings as you go. And this is exactly in a sense of what active networking, try to offer. But for different ownerships, like when you want to manage or when there is a change that you need to do on the network routers or switches, you can instead of going to program get to the device and change you could as well send the data that could make the updates that could even reflect the status back to the user and help user to the necessary connections. And this is where the active networking started. And this offered a major change in the networking paradigm.

Nonetheless, this was not as much successful because of the complexities that had in terms of the deployment. But it essentially paved the way of how we can think of networks to be active. And now if we look at it, in another sense, the networks in the earlier week last week, when we discussed about the end to end arguments, we said the network's will be bare minimal and dumb bytes that will just transmit the data.

And now we are talking completely contradictory to it where we are saying emphasizing that networks can even manipulate the data and act on it. It is as well as empowering the network to generate data on its own for transmit the data to the receiver, the way that the receiver would want to get rather than what the transmitter would want to send. So, these conflicting aspects also brought in, but for the good in terms of how we can think of networks as no more as a passive elements, but as active elements that can also help aid for better running better management, better control over the network.

And there were several of the intellectual contributions that were made by these active networks, one to say that they enabled the programmable functions in the network. And what we see with middle boxes is exactly the same where you have the middle boxes that manipulate on the data (10:40) what we discussed, it actually translates the packets, headers, right IP header and well 4 headers, the port and IP mappings are changed.

And likewise, with the firewalls and load balancers. So, these are all as a program we will functions which are now being considered part of networks. And this notion of programmable networks, in a way lowers the barrier for network innovation, so that we can think of adding innovations right into the network. In fact, Data Plane programmability is again coming to the forefront in what we are going to learn as network function virtualization.

The whole second aspect with this active networks is the network virtualization, wherein it provides the ability to de multiplex, the software programs based on the packet headers, and what that means is we could embed the programs onto the packet data and based on the type of packet data, different programs can be executed within the network and manipulate all the traffic before we transmit and forward, in a sense, is a kind of a virtualization that a network provides over the functionalities that it can build.

So, the construct of this active networks was based on two kinds of programming models, one as a capsule model, where the code that needs to be run is carried in band within the data packets, as we saw in this diagram here, where a packet would carry the X code, and X code is the one that is being operated on the network medium and transition its state from medium to medium.

And this was in fact, the most powerful of the model that became very popular, while the other was a programmable switch or a router module, where the switches or routers, which where we want to run the code could be established in an out of band fashion. And the functionality that you would achieve either way is the same but not necessarily that the data the code is carried in band within the packets, but this can be specific sets of controls that you want to build. And this is again, what we see when we want to manage specific devices. This out of band model really suits it. And this, in a way, very active networks, although it did not succeed over time, but it paved the way for how we can see the network's to be programmed.

(Refer Slide Time: 13:08)

ROAD TO SDN – THREE TIERS OF NETWORK EVOLUTION

1. **Active Networking:** Notion of Programmable Networks
2. **Separation of Control & Data Planes (Software Defined Networking):** Disaggregation with Open Interfaces.
3. **Network Operating Systems:** Widespread adoption of Open Interfaces and the evolution of Network controllers with OpenFlow APIs.



Software-Defined Networks

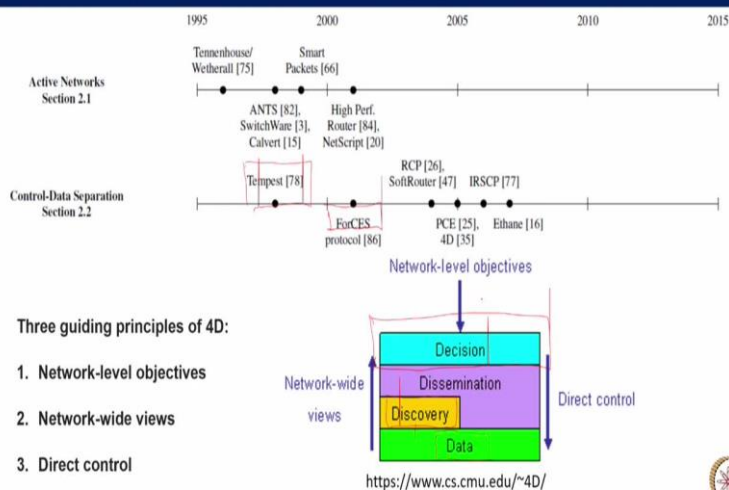
Advanced Computer Networks

8

The second of details is the separation of control and data planes and which is what we actively refer to or know as the software defined networking. In fact, this was the initiation journey for SDN, wherein the disaggregation of the hardware with respect to the software that would manage and control started with the integration of open interfaces.

(Refer Slide Time: 13:38)

ROAD TO SDN –TIER II: SEPARATION OF CONTROL AND DATA



Software-Defined Networks

Advanced Computer Networks

9

And in here, let us look at some of the key words that set the path for the separation of control and data planes. The first of the work is the tempest that was put forth, late 90s. And here, the authors discussed about how to allow the third party to access the network without jeopardizing the network integrity, or in a sense, how, as a network owner like a campus or

enterprise network, how I could accommodate legacy networking solutions, and then also have a programmable enrollment defined for my custom networks.

That was the core of saying how you would want to separate out the control over the network and what in a sense, the data part of the network needs to do. And this in fact, led to the setup of IETF working group called the forces or forwarding and control element separation. And this work, in fact, this IETF working group led to a series of RFCs charting out the key requirements, the framework for forces and the protocol specifications, and many things started to emerge in the plane of control and data separation.

In fact, you we had also led to a protocol as a means to separate the forwarding and control elements from the transport. And if you look at TCP, we have so many of the flags that try to control the way the data would move and needs to be handled. But now, if you think of transport protocol, which is specifically meant on one side to carry the data, the other side to carry the signals, and you want to separate these out.

And in fact, this led to what we call as the SCTP, or stream control transmission protocol, which is widely used in many of the real time streaming applications, and various of the live calls, etc. And this has played a major role in setting up of many of the aspects when we look at the framework for the control and data plane separation.

Next step, there were RCEP, soft router and PCE 4D works. Let us just try to look at what this PCE and 4D really meant. PCE stands for Path Computation Elements, which brought the notion of having a dedicated server, which could basically decouple the computation of the parts or the network parts. But from theouters, I do the computation of the parts on behalf of the network routers. So, we all know that when routers compute the part, they all do it independently.

Now, think of one server that can do this computation for many of these routers. And what this means is this path computation offload that happens to a dedicated server, it could be general purpose machine, it could be one specific router itself. And then this, in a way ensure that you are now able to virtualize the way the routing is done for a specific domain. And that was the initial separation of how the functionality of routing could be even though as a separate entity, which can be decoupled from the routers, tempest.

And this centralization of path computation, in fact, enables the operators to customize and control the routing policies and algorithms that you would want to run from a single location.

That is more important, because now if you want to change anything, you are just to go and update at one place rather than going and updating it, several of the routers that are spread in the network. And that is the flexibility the species started to provide.

And this work in fact influenced and led to what we see as a 4D project that started which was in fact the Clean Slate architecture for network control and management. And this work was guided by three principles. First is the network level objectives. That is, when you try to run a network which needs to have robust data characteristics, that is in terms of satisfying the performance objectives, or the reliability objectives or the policies that you want to ensure to be met when you want the network to transmit specific packets or flows of packets, you want to guarantee those network level objectives.

And to do that assurance of net meeting the network level objectives, what you really need is a network wide view which is more timely, accurate, and is able to provide the precise characteristics of what is happening at each of the links or the topology of the network, which can eventually be able to help achieve the network level objectives of providing a robust network.

For example, if there is a link failure, you want to have that view then and then rather than waiting for the routers to converge and decide what would be the alternative path, you would want to know the instance the link fades and free route or change the path of packets so that you are able to adapt pretty quickly and in real time. And you thus you are able to meet the network level objectives.

So, this means that there is a need to have the network wide views in a timely and accurate fashion. And in order to have this timely and accurate updates over the entire network, it would also call for having a direct control that is once I know that okay there is a failure of a link, I should have the control at each of the elements directly to say what would be the best alternate path which can be taken immediately rather than waiting for the routing algorithms to converge and that set up the alternate paths.

So, the decision logic should be provided to the network operators with a direct interface so that they can configure the network elements much more precisely in real time and these were the key guiding principles of the 4D project. And what this 4D really meant? Is to look at the network or view the network elements in 4 distinct layers of what we call the Ds

And the ground most at the lowest layer is the data element where it is confined towards processing of the packets alone when you it is just as the dumb pipe that would take the data and based on the rules that are being set it would forward data. So, forwarding as a key aspect. And the rules that are necessary for forwarding have to be configurable in real time.

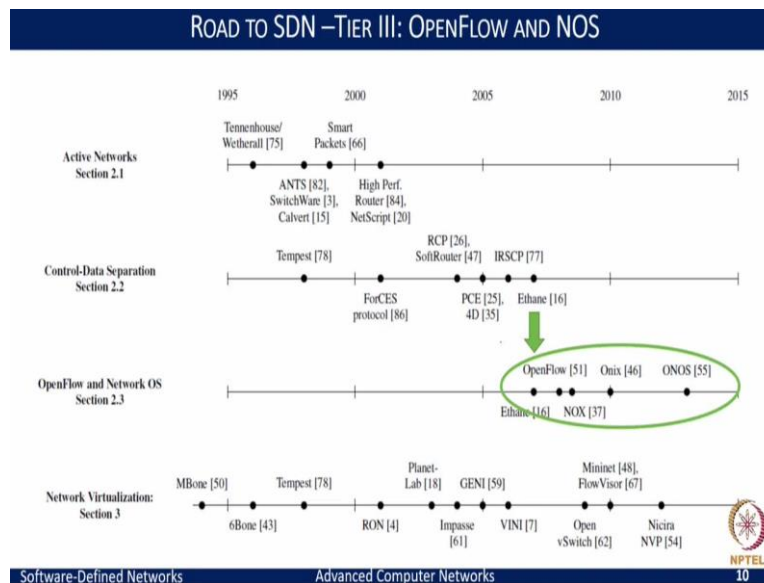
And the second of the layer that is built above is the discovery layer. And this discovery layer is crucial in providing the topology characteristics at a given time and also the traffic measurements in terms of which devices have the link bandwidths that are getting over utilized that are getting underutilized or what is the queue occupancy at each of the devices, what is the current communication topology looks like which links are up which links are down, kind of information in real time. So, that is being managed by the second of the D that is the discovery layer.

And once you discover the aspects in a network, you would want to quickly disseminate this information so that the decisions can be taken in real time. So, the third of the layer is the dissemination plane. And this dissemination plane enables the information collected by the discovery and the local information to be sent out to the network operators for installing the packet processing rules that are necessary or changes that need to be made in real time.

At this dissemination layer, once it is done, it is also important to have decision plane and this decision plane can be logically a centralized place where in the decision for the entire network not the distinct elements, but for all the elements when a network can be taken at one place. So, this consists of a logically centralized controllers to think of that can convert the network level objectives that you have based on the information that get disseminated to them from different entities in the network.

So, this is like a point where I can have a software or a program that can run adapt and tune the network characteristics to make and meet the network level objectives of having a robust network. And, in fact, this work paved the way further for how we think of today's software defined networks.

(Refer Slide Time: 23:01)



And the third of the tier is in fact, the evolution of the open flow and network operating systems. And in here, you can see a lot of works that started with ethane leading to open flow, Onix, ONOS, NOX and various other works have come in recent years. And the most fundamental work in this is the ethane, which published I think around 2007. And in fact, was influenced by the 4D project from a (())(23:40) and this ethane brings the vision to life, when you want to concretize several important aspects like how to boot up what is the registration policy setup, deployment aspects of the networking devices like routers and switches.

And this ethane work has been seminal in many aspects, wherein it paved the way to move the control plane out of the switching elements and also towards building a centralized controller wherein one element can take the decisions on behalf of the other switches and routers in your network. And Ethane's approach to define the API's or mechanisms to make this communication from these switching and router elements to one of these centralized entity as the network API's lead are influenced towards the creation of open flow and open flow started around 2008.

And eventually, this Ethane and open flow led to a startup called Nicira which was later acquired by VMware and what we now know as the Open flow southbound API's and the SDN with the control plane has become a common norm. And this in a nutshell is the eventual is the roadmap of how the SDN what we see today started, and much of the works are there to follow. And in fact, over the last decade, there have been more than thousands of publications in the area of software defined networks.