The Journey to SDN: A Peek into the History of Programmable Networks

M.Bindhu[1], Ramesh.G.P[2],
Research Scholar, St.Peter's University, Chennai, India[1],
Professor ,Department of ECE, St.Peter's University, Chennai, India[2],
deepbind@gmail.com[1] ,rameshgp@yahoo.com[2]

Abstract: This paper proposes to do a survey and study of the state-of-art programmable networks with a focus on SDN. It will present a historic outlook of programmable networks from early concepts to very recent progress. This is followed by a study of SDN architecture and the OpenFlow standard. Currently available options for implementation and testing of SDN-based protocols and services are also discussed. SDN applications for current and future are also examined.

Index Terms-Software-Defined Networking, Data plane, Control plane, Mininet, Virtualization.

1.Introduction: A typical computer network comprises of a huge number of network devices such as routers, switches and various types of intermediary devises such as a firewall which will also manipulate traffic though it is not for packet forwarding purposes. Many complex protocols are implemented on these intermediary devises. Network events and applications are managed by network administrators by configuring policies. This is done manually by transforming the high level-policies into low-level configuration commands which will adapt to the changing network conditions. For achieving this very complex tasks , very limited tools are currently at their disposals. Because of that, network administration and fine tuning of performance is quite challenging and thus error-prone .Vertical integration of network devises aggravates the challenges faced by network operators and administrators. "Internet ossification" is another almost overwhelming challenge encountered by network practitioners and researchers. Very large deployment base and the fact that now computer network is considered as one of our society's essential infrastructure (like power and transportation services), makes it extremely difficult for the Internet to evolve in terms of the physical infrastructure as well as its protocols and performance. Present and future Internet applications and services are becoming progressively more complex and demanding. This makes it crucial that the Internet be able to evolve to address these new challenges. The paper is planned as in Section 2, begins by saying about Literature Survey. Section 3 illustrates on overview of SDN architecture which describes the switches and Open Flow protocol. Section 4 and 5 states development tools of SDN implementations, verification and debugging tools. In Section 6, we talk about several SDN applications in areas such as data centers and optical networking.

2. Literature Survey Of SDN: SDN is undoubtedly one of the most promising concept which can change the way networks function, and particularly OpenFlow has been termed as a "radical new idea in networking" [1]. SDN is proposed to bring in changes ranging from centralized control, simplification of algorithms, network hardware commoditization, eliminating middle ware, to facilitating the design and deployment of third-party 'apps'. Though OpenFlow received considerable industry attention recently only, the concept of programmable networks and decoupled control logic has been there for many years. Under this section, an overview of early programmable networking efforts, predecessors to the current SDN paradigm (that laid the foundation for most of the ideas we are seeing today), is provided.

2.1 DCAN: An initiative that started in the mid 1990s is the Devolved Control of ATM Networks (DCAN) [2]. The objective of this project was to design and develop necessary infrastructure for scalable control and management of ATM networks. The hypothesis was that the control and management functions of many devices (ATM switches in this case) should be decoupled from the devices and assigned to external units dedicated for that purpose. This

basically is the idea behind DNs. DCAN assumes a simple protocol between the manager and the network, which is similar to what happens today in applications such as OpenFlow.

2.2 Active Networking: Another initiative started in the mid 1990s, the Active Networking [3], [4] suggested the idea of a network infrastructure that could be programmed for customized services. Two main approaches were being considered, namely: (i) user programmable switches, with in-band data transfer and out-of-band management channels; and (ii) capsules, which were fragments of program that could be carried in user messages; which would then be interpreted and executed by routers.

2.3 Open Signaling: The Open Signaling (OPENSIG) working group started in 1995 as a series of workshops committed to "making ATM, Internet and mobile networks more open, extensible, and programmable" [5]. According to them a division between the communication hardware and control software was necessary but challenging to implement. This is mainly because of vertically integrated switches and routers. In addition to that the closed nature of these switches and routers made rapid deployment of new core proposal was to give access to the network hardware via open, programmable network interfaces, allowing the employment of new services through a distributed programming environment.

2.4 4D Project: Started in 2004, this concept proposed to give the "decision" plane a global view of the network, which is serviced by a "dissemination" and "discovery" plane, for control of a "data" plane for forwarding traffic. These concepts formed the basics for later works such as NOX [6]. NOX proposed an "operating system for networks" in the context of an OpenFlow enabled network

2.5 NETCONF: The IETF Network Configuration Working Group suggested NETCONF [7] in 2006, as a management protocol for updating the configuration of network devices. NETCONF allowed an API to be exposed for network devices, through which all configuration data could be sent and retrieved. A NETCONF network should not be considered as fully programmable because any new functionality has to be implemented both at the network device and the manager level so that proper management can take place. NETCONF is primarily designed to facilitate the configuration automatic and does not provide direct control of state data.

2.6 Ethane: The immediate predecessor to OpenFlow was the SANE / Ethane project [8], which, in 2006, defined a new architecture for enterprise networks. Ethane's focus was on using a centralized controller to manage policy and security in a network. A notable example is providing identity- based access control. Similar to SDN, Ethane employed two components: a controller to decide if a packet should be forwarded, and an Ethane switch consisting of a flow table and a secure channel to the controller. Ethane laid the foundation for what would become Software Defined Networking. To put Ethane in the context of today's SDN paradigm, Ethane's identity-based access control would likely be implemented as an application on top of an SDN controller such as NOX [6], Maestro [9], Beacon [10], SNAC [11], Helios [12], etc.

3.SDN Architecture: The Open Networking Foundation (ONF) [13] is a leading organization responsible for standardization and advancements in SDN. The clients who request for a service, a server which promises to deliver the service and a forwarding device responsible for carrying out the communication in a network are common components. Present condition for the customers in network devices comes with preinstalled software from manufacturer and leave a small pace for customization option for the customers. The huge exploitation base of internet and its stiffness is referred to as "Internet Ossification" and it is impossible to implement any major changes to it . The vertical integration of traditional networks that is tight coupling of forwarding plane and control plane, delay innovation. The SDN architecture introduced plays a vital role in splitting the networking devices into data plane and control plane. Forwarding packets very fast and efficiently is done by Data plane ,which is not intelligent but can perform physically carrying data packets from one port to another by rules that are programmed into the device hardware. Control plane has greater responsibility in deciding the logic to program the data plane. The communication of two planes as standard protocol over a secure channel is called OpenFlow [13]. Fig 1 shows the logical organization of SDN. Decoupling control plane from data plane makes control plane programmable which

enables abstraction of network device from application and service layers which in turn acts as virtual entity. It makes enterprises to gain vendor independent control over network there by simplifying the design, operation and maintenance. Hence SDN devices can no longer need to store the logic telling what to do when packets arrive; instead they depend on the controller for the operation. Hence provides flexibility for traffic engineers to develop and deploy applications like routing and security at one place instead on each and every device thereby reduces the time involved in adapting to changes in policy of controlling the underlying devices.

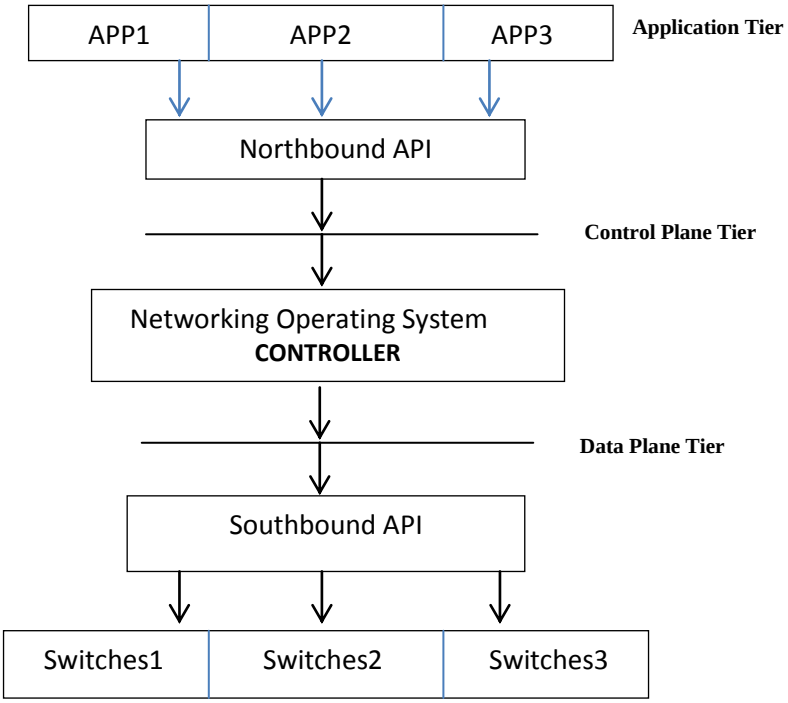

Figure1.SDN Architecture

3.1 Data Plane: It is forwarding plane which has flow tables with flow entries and an abstraction layer which can safely with controller through OpenFlow protocol. The flow entries firstly consists of match fields with packet header, ingress port and data. Secondly counters to know about statistics of received packet with bytes &duration of data and finally actions to be taken upon the matching. On the packet arrival at OpenFlow switch, packet headers are matched with flow tables entries. If matching is found, flow actions take place as needed . If the matches doesn't take place, action is forwarded to table miss entry in flow table .This action will direct the packet to drop, continue with next flow table ,forward it to controller for further actions. Several versions of OpenFlow is defined but 1.1 versions has multiple entries with pipeline method and also uses hybrid switches which are commercial available as it uses for traditional and SDN networks.

3.2 Control Plane: The logically centralized is in charge of (i) the requirements are translated from the Application layer down to the Data paths and (ii) providing the Application plane with an abstract ,slicing view of the network. An SDN Controller has north bound interface agents, Control Logic module, and the interface driver. Applications is written in Java which can interact with the built-in controller modules via a JAVA API. REST API is used for other applications which uses different languages and interact with the controller modules. The example of controller allows the execution of built-in modules that can exchange of with their implementation of the OpenFlow controller (i.e. Services). On the other side the controller, can communicate with the data plane via the OpenFlow protocol through the abstraction layer which is present at the data plane hardware.

4.Development Tools

Mininet :Mininet is a network emulator used to create scalable SDNs via Linux processes. topology, traffic flows, new services ,protocols ,applications are developed and tested in this environment before developing hardware. Topology is created with a Python script and the traffic flows are established from a OpenFlow controller.

EstiNet 8.1:This is a recent entrant (not a freeware) , in the field of OpenFlow network simulator and emulator. The most unique feature is its capability to run controller application program, created by any OpenFlow researcher, on an EstiNet8.1[36] simulated OpenFlow network without any changes. This is made possible due to the innovative "kernel-reentering simulation methodology" used by EstiNet. It is advantageous than present Mininet platform as packet forwarding capability of a switch in Mininet is unpredictable.

Software Switch Platforms: Currently numerous SDN software switches are available to run an SDN test bed also in developing services over SDN. Table I represents current software switch implementations with language used and the OpenFlow standard version that supports. Table II says about the native SDN commercial switches available, with their manufacturer and the version of OpenFlow .

Open vSwitch [34]	c/python	Openflow 1.0 import on ASIC
ofsoftswitch13 [36]	c/c++	Openflow1.3
Indigo [37]	c	Openflow1.0 with ASIC/Ethernet
Pantou/OpenWRT [35]	c	Wireless router

TABLE I:CURRENT SOFTWARE SWITCH WITH OPENFLOW STD

Maker	Switches	Version
HP	8200zl, 6600, 6200zl,500/3500yl	1.0
IBM	RackSwitchG8264	1.0
juniper	Junos MX-Seri	1.0
Brocade	NetIron CES 2000	1.0
NEC	PF5240 PF5820	1.0

TABLE II: MAIN SOFTWARE SWITCH MAKERS WITH OPENFLOW STD

Controller Platforms :The Table III provided here gives a brief view of the listed controllers. For example the Beacon controller is java based developed by Stanford cross platform, modular, Java based OpenFlow controller that supports event-based and threaded operations. The Floodlight [14] Java is by BigSwitch , Java based OpenFlow controller (supports v1.3), based on the Beacon implementation, that works in physical and virtual OpenFlow switches. The routing engines produces the forwarding information base (FIB) to the Linux IP tables by the routing protocols configured (e.g., OSPF, BGP). The RouteFlow extension is presented in [15], which discusses Routing Control Platforms (RCPs) in the context of OpenFlow/SDN. The controller centric networking model with a prototype implementation of an autonomous system with BGP routing service.

Controller	Source code	View
POX[33]	Python	Nicira,opensource
NOX[6]	Python/C++	Nicira,opensource
Trema[30]	Ruby/C	NEC,Freesource
Beacon[10]	JAVA	Stanford, event based
Floodlight[14]	JAVA	Bigswitch,openflow1.3v
Flowvisor [31]	C	Stanford, special controller implementations
RouteFlow [32]	C++	CPqD, special controller implementations

TABLE III: CURRENT SDN COMMERCIAL SWITCH WITH OPENFLOW STD

5.DebuggingTools: NICE [16] a testing tool used to help correct bugs in OpenFlow programs via model checking and executing symbolically. Anteater [17] helps by attempting to check network invariants which exist in the data plane, for connectivity or consistency. It will also catch errors that result from faulty switch firmware. VeriFlow [18] goes one point further by proposing a real time verification tool that cites between the controller and the data plane. OFRewind [19] allows all network events to be recorded at different levels and later produced for granting the opportunity to localize and troubleshoot the events that are caused by the network difference. The ndb [20] implements breakpoints and packet back traces for SDN. With software debugger gdb, it can pinpoint events that lead to error by pausing execution even at a breakpoint also by using a packet back trace, which show the sequence of forwarding actions seen by that packet. STS [21] is a SDN troubleshooting simulator. which is in python and works on POX. Under test conditions the simulation of the devices in a given network is done and identifies the set of inputs that produces a given error.

6. SDN Applications: In networked environments, Software Defined Networking has got a wide range of applications. By separating the control and data planes, programmable networks facilitate the following i) More flexibility and customization in the control of the network. ii) Elimination of many middle boxes currently deployed in the network.iii) Rapid development and deployment of new simplified network services and protocols. Different environments for which SDN solutions have been already implemented or proposed are discussed below.

6.1 Enterprise Networks: Large networks run by enterprises demand strict security and performance requirements. Moreover, different enterprise environments will have very diverse needs, characteristics, and user numbers. Campus networks are a typical example of an enterprise network. In a University network environment, most of the connected devices are temporary. SDN gains significance while considering management of Enterprise Networks. SDN allows many functionalities of network middle boxes to be integrated within the network controller. This eliminates the need for additional network hardware and facilitates rapid scalability and deployment. Some of the prominent examples of this functionality integration using SDN are: NAT, firewalls, load balancers [22] [23], and network access control [24].In order to specifically address the issues faced by enterprise networks, OpenFlow network architecture was designed based on the Ethane [8], platform.

6.2. Data Centers: In recent years, Data centers have grown at an amazing pace both in terms of size and complexity. Data centers handle huge data volumes. Any disruption in service or delayed service delivery may result in enormous losses both in economic terms and productivity terms. This can adversely affect millions of people across globe. Because of this Data centre traffic management and policy enforcement becomes very critical functions. Though simple traffic management and visibility offered by SDN are useful, it must be prudently balanced with scalability and performance demands. Curtis et al. [25] suggests that OpenFlow couples central control and absolute visibility excessively. But in real life scenario only " most significant" flows need to be managed. This issue may get resolved through persistent use of practical policies and wild card rules, but it may dent the ability of the controller to have the right granularity to effectively manage traffic and collect statistics. Their framework, DevoFlow, proposes to have some modest design modifications to keep flows in the data plane as much as possible while upholding enough visibility for effective flow management. This is achieved by transferring responsibility of most flows back to the switches and adding more efficient statistics collection mechanisms. Using this method, "Significant" flows (e.g. high throughput, long lived) are thus identified and managed by the controller. In a load-balancing simulation, on average, their solution had 10 - 53 times lesser flow table entries and 10 - 42 times lesser control messages over OpenFlow.

6.3. Optical Networks: By treating data traffic as flows, SDN and OpenFlow networks can support and integrate multiple network technologies. It then becomes possible for SDN to provide technology independent and unified control for optical transport networks. This will pave the way for communication between both packet and circuit switched networks. Optical Transport Working Group (OTWG) formed in 2013 by the Open Network Foundation (ONF), states that by applying SDN and the OpenFlow standard to optical transport networks the benefits obtained are: i) Improved control and management flexibility for optical transport network. ii) Allowing deployment of third party management and control systems.iii) Deployment of new services by leveraging virtualization and SDN [26].OpenFlow protocol has been used in many attempts and proposals to control both packet switched and circuit

switched networks. In [27] a NetFPGA [28] platform, using the OpenFlow protocol is employed in the proposal of a packet switching and circuit switched networks architectures based on Wavelength Selective Switching (WSS). A conceptual illustration of wavelength path control in transparent optical networks, based on OpenFlow is presented in [29]. Virtual Ethernet interfaces (veths) are introduced in this work. Physical interfaces of an optical node (e.g. photonic cross connect - PXC) are then mapped to the veths. An SDN controller (e.g. NOX controller) is enabled to control the optical light paths(e.g., by using OpenFlow protocol). Software Defined Optical Network (SDON) architecture is introduced in [30] and QoS for optical burst switching in OpenFlow based on SDON is developed. Results indicate that SDON provides a viable infrastructure to support unified control protocols for optimized network performance and improved network capacity.

7. Concluding Remarks: This paper, gives an overview of programmable networks. The emerging field of Software Defined Networking (SDN) is examined and the history of programmable networks, from initial ideas to recent developments is studied. SDN architecture as well as the OpenFlow [13] standard is illustrated in detail. Recent developments in SDN implementations and testing platforms are exemplified. Network applications and services that have been developed based on the SDN paradigm are studied.

References

[1] Thomas A. Limoncelli. Openflow: a radical new idea in networking. Commun. ACM, 55(8):42–47, August 2012.
[2] DevelopedcontrolofATMnetworks.http://www.cl.cam.ac.uk/research/srg/netos/old-projects/dcan/#pub.
[3] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, andG.J. Minden. A survey of active network research. IEEE Commun.Mag., 35(1):80–86, 1997.
[4] D.L. Tennenhouse and D.J. Wetherall. Towards an active networkarchitecture. In DARPA Active NEtworks Conf. and Exposition,Proc., pages 2–15. IEEE, 2002.
[5]. AT Campbell, I Katzela, K Miki, J Vicente -Open signaling for ATM, internet and mobile networks (OPENSIG'98) ,29(1):97-108, 1999.
[6] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown,and S. Shenker. Nox: towards an operating system for networks. ACM SIGCOMM Computer Commun. Review, 38(3):105–110, 2008.
[7] R. Enns. NETCONF Configuration Protocol. RFC 4741 (Proposed Standard), December 2006. Obsoleted by RFC 6241.
[8].M.Casado,MJ Freedman, J Pettit, J Luo, N McKeown,and S.Shenker Ethane: Taking control of the enterpriseACM SIGCOMM Computer communication review , Volume 37 Issue 4, 4-12,October 2007
[9] Zheng Cai Alan L. Cox T. S. Eugene Ng,Maestro: A System for Scalable OpenFlow Control, TSEN Maestro - Technical Report TR10-08, Rice University,2010
[10]. Beacon, https://openflow.stanford.edu/display/Beacon/Home
[11].SNAC-http://archive.openflow.org/wp/downloads
[12].Helios by nec .http://www.nec.com
[13] Open networking foundation https://www.opennetworking.org/about
[14]. Floodlight, An open SDN controllerhttp://floodlight.openflowhub.org/
[15] Christian Esteve Rothenberg, Marcelo Ribeiro Nascimento, Marcos Rogerio Salvador, Carlos Nilton Araujo Corrˆea, Sidney Cunha de Lucena, and Robert Raszuk. Revisiting routing control platforms with the eyes and muscles of software-defined networking. In Proc. 1st workshop on Hot topics in software defined networks, HotSDN '12, pages 13–18, New York, NY, USA, 2012. ACM.
[16] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford. A nice way to test openflow applications. NSDI, Apr, 2012.
[17] Haohui Mai, Ahmed Khurshid, Rachit Agarwal, Matthew Caesar, P. Brighten Godfrey, and Samuel Talmadge King. Debugging the data plane with anteater. In Proc. ACM SIGCOMM 2011 conf., SIGCOMM '11,41(4) pages 290–301, New York, NY, USA, 2011. ACM
[18] Ahmed Khurshid, Wenxuan Zhou, Matthew Caesar, and P. Brighten Godfrey. Veriflow: verifying network-wide invariants in real time.ACM SIGCOMM Computer Communication Review , HotSDN '12, pages 49–54, New York, NY, USA, 2012.ACM
[19] Andreas Wundsam, Dan Levin, Srini Seetharaman, and Anja Feldmann.Ofrewind: enabling record and replay troubleshooting for networks. In Proc. 2011 USENIX conf. on USENIX annu. technical conf., USENIXATC'11, pages 29–29, Berkeley, CA, USA, 2011.USENIX Association.

[20] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazi´eres, and Nick McKeown. Where is the debugger for my software- defined network? In Proceedings of the first workshop on Hot topics in software defined networks, HotSDN '12, pages 55–60, New York, NY, USA, 2012. ACM.
[21]. sdn trouble shooting simulator https://github.com/ucb-sts
[22] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari.Plug-n-serve: Load-balancing web traffic using openflow. ACM SIGCOMM Demo, 2009.
[23]R. Wang, D. Butnariu, and J. Rexford. Openflow-based server load balancing gone wild. In Workshop of HotICE, volume 11, 2011.
[24] A.K. Nayak, A. Reimers, N. Feamster, and R. Clark. Resonance:Dynamic access control for enterprise networks. In Proc. 1st ACMworkshop on Research on enterprise networking, 11–18. ACM,2009
[25] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula,Puneet Sharma, and Sujata Banerjee. Devoflow: scaling flowmanagement for high-performance networks. SIGCOMM ComputCommun. Rev., 41(4):254–265, August 2011
[26] Optical transport working group otwg. In Open Networking Foundation ONF, 2013.
[27] V. Gudla, S. Das, A. Shastri, G. Parulkar, N. McKeown, L. Kazovsky,and S. Yamashita. Experimental demonstration of openflow controlof packet and circuit switches. In Optical Fiber Communication(OFC), collocated National Fiber Optic Engineers Conference, Conference on (OFC/NFOEC), pages 1–3, 2010.
[28] Netfpga platform. http://netfpga.org
[29]A.N. Patel, P.N. Ji, and Ting Wang. Qos-aware optical burst switching in openflow based software-defined optical networks.n OpticalNetwork Design and Modeling (ONDM), 17th International Conference on, pages 275–280, 2013.
[30]Tremaopenflowcontrollerframework. https://github.com/trema/trema
[31] R. Sherwood, M. Chan, A. Covington, G. Gibb,M. Flajslik, N. Handigol, T.Y. Huang, P. Kazemian, M. Kobayashi, J. Naous,Carving research slices out of your production networks withopenflow ACMSIGCOMM Computer Communication Review, 40(1):129–130, 2010
[32]Marcelo R. Nascimento, Christian E. Rothenberg, Marcos R. Salvador,Carlos N. A. Correa, Sidney C. de Lucena, and Maur ˆ ´ıcio F. Magalhaes. ˜Virtual routers as a service: the routeflow approach leveraging SDNIn Proceedings of the 6th International Conferenceon Future Internet Technologies, CFI '11, pages 34–37, New York, NY,USA, 2011. ACM.
[33] Pox. http://www.noxrepo.org/pox/about-pox/.
[34]Open vswitch and ovs-controller. http://openvswitch.org/.
[35] Pantou: Openflow 1.0 for openwrt. http://www.openflow.org/wk/index.php/ Open- Flow 1.0 for OpenWRT.
[36] Ofsoftswitch13 - cpqd. https://github.com/CPqD/ofsoftswitch13.
[37] Indigo: Open source openflow switches. http://www.openflowhub.org/display/Indigo/.
[38] "EstiNet OpenFlow Network Simulator and Emulator" has been published in the IEEE Communications Magazine. September 2013