Software Defined Networks: A Survey

Shruthi S Manvachar

***Abstract*— Internet has led to the creation of a digital society, where everything is connected and is accessible from anywhere. However, despite their widespread usage, traditional IP networks are complex and very hard to manage. It is both difficult to configure the network according to defined policies, and to reconfigure it to respond to faults, loads and changes. To make matters even more difficult, current networks are also vertically integrated i.e. the control and data planes are combined. With the increasing demand, usage of bandwidth and frequency spectrum resources is beyond expectations. The frequency spectrum and network information have considerable relevance. Thus the spectrum utilization and channel flow interactions should be simultaneously considered. Software Defined Networking (SDN) has evolved to change this state of affairs, by breaking this vertical integration.**

***Index Terms*— Control plane; Data plane; OpenFlow; Software Defined Networking**

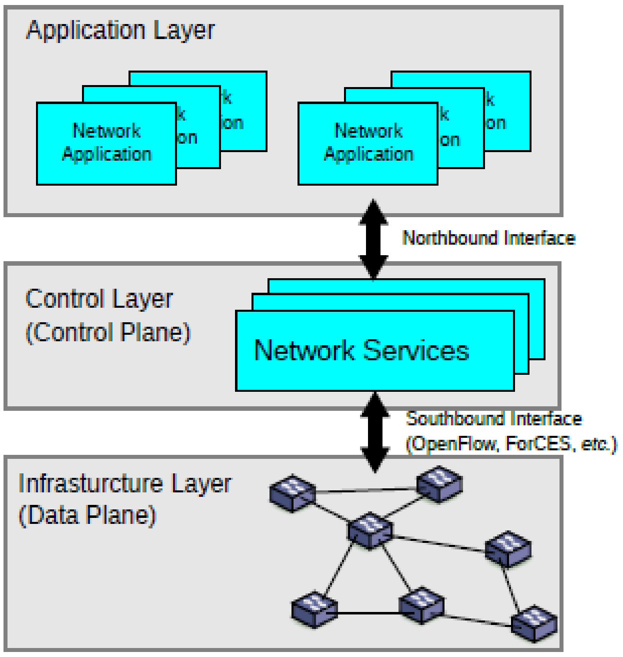
I. Introduction

Software defined networking – SDN is gaining a lot of attention in recent years as it addresses the lack of programmability in existing networking architectures by enabling easier and faster network innovation. SDN separates the data plane from the control plane and facilitates software implementations of complex networking applications on top. It supports for less specific and cheaper hardware that can be controlled by software applications through standardized interfaces. There is an expectation for more flexibility by dynamically adding new features to the network in the form of networking applications. This is a known concept from mobile phone operating systems, such as Apple's iOS and Google's Android, where "apps" can dynamically be added to the system [6].

According to the Open Networking Foundation (ONF), the SDN is:
• Directly programmable: Network control is directly programmable because it is decoupled from forwarding functions.
• Agile: Control is abstracted from forwarding which lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
• Centrally managed: Network intelligence is centralized (logically) in software based SDN controllers that maintain a global view of the network which appears to applications and policy engines as a single logical switch.
• Programmatically configured: SDN allows network managers to manage, configure, secure, and optimize the network resources very quickly through dynamic, automated SDN programs, which can be written by them as programs do not depend on proprietary software.
• Open standards and vendor-neutral: Implemented through open standards, SDN simplifies network design and operations as instructions are provided by SDN controllers instead of multiple vendor-specific devices and protocols [2].

Manuscript received January 03, 2015.
Shruthi S Manvachar, Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bangalore, India

II. SDN Architecture


Application Layer
Network Application
Network Application
Northbound Interface
Control Layer (Control Plane)
Network Services
Southbound Interface (OpenFlow, ForCES, etc.)
Infrasturcture Layer (Data Plane)

A three-layer Software-Defined Networking (SDN) Architecture [6]

- The control and data planes are decoupled.
- Network intelligence and state are logically centralized
- The underlying network infrastructure is abstracted from the applications. [1]

SDN consists of four key features:

- It separates the control plane from the data plane.
- A centralized controller and view of the network.
- Open interfaces between the devices in the control plane (controllers) and those in the data plane.
- Programmability of the network by external applications [1].

The networks that employ programmable routers to facilitate the routing mechanism are known as Software-Defined Networks.

- The SDN framework consists of three layers. The bottom layer is the infrastructure layer which is also called as the data plane. It consists of forwarding network elements. The tasks of the forwarding plane are mainly data forwarding, monitoring local information and gathering statistics [6].

- The middle is the control layer also called as the control plane. It is responsible for programming and management of the forwarding plane. It makes use of the information provided by the forwarding plane and defines network operations and routing. It consists of one or more software controllers that communicate with the forwarding network elements through standardized interfaces. OpenFlow mainly considers switches, while other SDN approaches consider other network elements such as routers. [6]

- The application layer contains network applications that can introduce new network features, such as security and manageability, forwarding schemes or assists the control layer in the network configuration.

- The application layer can receive an abstracted and global view of the network from the controllers and use that information to provide appropriate guidance to the control layer. The interface between the application layer and the control layer is referred to as the northbound interface. For the latter, no standardized API exists today, and in practice, the control software provides its own API to applications. [6][23]

III. SDN Development tools

A. Emulation and Simulation Tools

Mininet is one which allows an entire OpenFlow network to be emulated on a single machine thus simplifying the initial development and deployment process. The ns-3 network simulator supports OpenFlow switches within its environment, though the current version only implements OpenFlow v0.89

B. Available Software Switch Platforms

There are currently several SDN software switches available that can be used to run an SDN testbed or when developing services over SDN. For example Open vSwitch, Indigo.

C. Native SDN Switches

Commercial switches available are NetIron CES 2000 Series, RackSwitch G8264 etc.

D. Available Contoller Platforms

Flowvisor is a transparent proxy between OpenFlow switches and multiple OpenFlow controllers. It creates network slices and can delegate control of each slice to a different controller, also promoting isolation between slices. RouteFlow is an open source project that provides a virtualized IP routing over OpenFlow capable hardware.

E. Code Verification and Debugging

Verification and debugging tools are vital resources for SDN also as for traditional software development. For portable network apps to be successful, network behaviour must be thoroughly tested and verified.

NICE is an automated testing tool used to uncover bugs in OpenFlow programs through model checking and symbolic execution.

Anteater checks network invariants that exist in the data plane such as connectivity or consistency .It is protocol-agnostic and also catches errors that result from faulty switch firmware or inconsistencies with the control plane communication.

VeriFlow proposes a real-time verification tool that resides between the controller and the forwarding elements. This adds the potential benefit of being able to halt rules that will cause anomalous behaviour before they reach the network [4].

IV. Open Flow

The OpenFlow protocol has been accepted as the interface between the control and data planes. OpenFlow provides per flow statistics collection primitives at the controller. The controller can poll a switch to collect statistics on the active flows. It can request a switch to push flow statistics (upon flow timeout) at a specific frequency. The controller has a global view of the network. An effective monitoring solution can be developed using these capabilities of an OpenFlow Controller. A network management application for SDN would be a part of the control plane, rather than being independent of it. This is due to the heterogeneity in the controller technologies, and the absence of a uniform abstract view of the network resources [10].

OpenFlow switches runs embedded software needed to process control messages sent by the controller and configures flow tables accordingly. This software needs to be compliant with the OpenFlow specification. The specifications may be ambiguous and may have several interpretations, which may give implementation freedom to vendors which lead to implementations that exhibit compatibility and interoperability. In SDN deployment scenarios, the infrastructure is constituted of OpenFlow switches from multiple vendors. Thus, this kind of problems can easily occur at the forwarding infrastructure level. Systematic OpenFlow Testing (SOFT) is an exhaustive approach and tool for automated switch interoperability testing using symbolic execution and the constraint solver (having as input formulas over the theory of bitvectors and arrays that captures most expressions from languages like C, C++, Java, Verilog etc). This approach allows leverage multiple OpenFlow implementations at the development stage.
[11][24][25][26][27]

OpenFlow controllers

1) *Maestro:* Maestro is a multi-threaded OpenFlow controller. OpenFlow packets are received from the sockets by the main thread and put in the shared raw-packet queue.

Packet batching is used but the number of bytes to be read is not static and depends on the present workload. Maestro is the only publicly available controller which uses task batching so that worker threads pulls a batch of tasks to process multiple flow-requests in a single execution. Output batching technique is used to send packets out in which packets belonging to the same destination are grouped together and sent using a single socket system call [12].

2) *NOX-MT:* NOX-MT has a multi-threaded architectural designwhich uses the Boost::Asio libraries for network and low-level I/O programming. Boost::Asio acts as a switch partition and is responsible for distributing the connecting OpenFlow switches to worker threads statically. Static packet batching technique is used to reduce frequent read system calls. The incoming packets are processed one by one and then batched together in case of high control traffic before they are sent out. The static input batching helps NOX-MT to achieve a high throughput performance but affects its latency [12].

DISCO, a DIstributed SDN Control plane for WAN and constrained overlay networks, relies on a per domain organization, where each controller is in charge of an SDN domain, and provides a lightweight and highly manageable inter-controller channel. DISCO dynamically adapts to heterogeneous network topologies while being resilient enough to survive to disruptions [13]. DISCO is able to support in the distributed and heterogeneous nature of modern mission-critical networks [14].

Software Defined Networking and OpenFlow allows for better network control and flexibility in the pursuit of operating networks as efficiently as possible. OpenFlow provides interfaces to implement fine-grained Traffic Engineering (TE) [17].

The re-configurability is the ability to safely update the current forwarding states at the data plane to the new states without interrupting the data flows. When updating, one or multiple controllers write to the forwarding states of their switches via the OpenFlow (OF) protocol. The forwarding states control different switch resources to decide the output behaviors of the incoming packets at the switch level.
Example includes the forwarding rules in the flow table, the rate limiting on the input queues as well as up and down statuses of the ports [18].

CrossRoads:

Being a single point of failure and a performance bottleneck, a single OpenFlow controller across multiple data centers is not practical since direct wire or hub based layer 2 connectivity is required between the controller and all the network elements. CrossRoads works across multiple data centers, each with their own one or more independent OpenFlow controllers. CrossRoads is a network fabric that provides a layer agnostic and seamless online and offline VM mobility across multiple data centers. CrossRoads extends to location independent based on pseudo addresses to work with control plane overlay of OpenFlow network controllers in several data centers [19]

Palette: Distributing Tables

Palette is a framework which decomposes and distributes SDN tables across the network. Palette is important as switch table sizes can become a bottleneck in scaling SDNs. It facilitates handling the heterogeneity of switches in the network and the changes of equipment [1].

V. Scalability of SDN

SDN moves the control function out of data plane. This enables both the planes to evolve independently and has many advantages like high flexibility, being vendor agnostic, programmability, and the possibility of realizing a centralized network view. But there have always been concerns about performance and scalability.

It is nontrivial to define a standard API between the two planes as this API should handle the needs of several architectures and provide independent evolution of both the planes. All or most of switch vendors should be able to adopt to a standardized API for it to be useful or else networks will be in control of specific vendors, which leads to proprietary layers, which prevents rapid change and innovation in networks.

OpenFlow API can be extended in order to better handle the concept of “content” and to support new content-related methods, such as key management, caching and routing-by-name, and so on. OpenFlow architecture and protocols can be modified and enhanced to support Information Centric Networks (ICN) [20].

Transferring of traditional local control functionalities to a remote controller results in new bottlenecks like signalling overheads that can be significant depending on the type of network and associated applications.

Controller Scalability: SDN pushes all the control functionality to a centralized controller providing a complete network wide view, developing control applications and enforcing policies. But controllers can become bottleneck in the network operation when the size of network grows; more events and requests are sent to the controller which fails to handle all the incoming requests [22].

VI. SDN APPLICATIONS

Software-defined networking has applications in a wide variety of networked environments. The following are different environments for which SDN solutions have been proposed or implemented.

Data Centers

Data Centers have evolved at an amazing pace in recent years, constantly attempting to meet rapidly changing demand. Enforcing policy and traffic management is critical when operating in large-scale, as any service disruption or delay may lead to massive loss. An increasingly important consideration is energy consumption, which has a non-trivial cost in large-scale data centre. A power manager that utilizes SDN to find minimum power network subset satisfies current traffic conditions and turns off switches that are not needed. As a result, they show energy savings between 25-62% under varying traffic conditions.

Enterprise Networks

Enterprises often run large networks, while also have strict security and performance requirements. In addition, different enterprise environment have different requirements, characteristics, and user population, for example, University networks can be considered a special case of enterprise networks. In this environment, many of the connecting devices are temporary and not controlled by the University, challenging security and resource allocation. Universities must often provide support for research test-beds and experimental protocols. Adequate management is critically important in Enterprise environments, and SDN can be used to programmatically enforce and adjust network policies as well as help monitor network activity and tune network performance [4].

VII. CONCLUSION

Software defined Networking is an innovative trend in communication networks. The main idea behind SDN is to decouple the control and data planes in a router. A router in a typical SDN environment shall only forward data, while all routing decisions shall be made by an external device known as controller. The number of controllers and their placements in the network depends on a variety of factors. However, for medium sized networks, one or two controllers may be sufficient.

Software-Defined Networking (SDN) can provide an enhanced interaction between networks and applications, and a more dynamic and demand-based allocation of network resources to heterogeneous applications. This paper highlights SDN architecture and OpenFlow

ACKNOWLEDGEMENT

I consider it is a privilege to express my gratitude and respect to all those who guided me in completion of technical paper. The research presented in this paper is supported by management of M. S. Ramaiah Institute of Technology. It’s a great privilege to place on record my deep sense of gratitude to our HOD Dr. K G Srinivas, Computer Science & Engineering, M. S. Ramaiah Institute of Technology. I am grateful to thank Dr. S.Y Kulkarni, Principal, M. S. Ramiah Institute of Technology. I am grateful to thank Dr. Monica R Mundada, Associate Professor, Computer science department, M. S. Ramaiah Institute of Technology, for her invaluable support and guidance.

I would like to thank the reviewers for their time and expertise, constructive comments and valuable insights.

REFERENCES

[1] Sakir Sezer, Sandra Scott-Hayward, and Pushpinder Kaur Chouhan, “Are We Ready for SDN? Implementation Challenges for Software-Defined Networks”, DOI:10.1109/MCOM.2013.6553676, July 2013, Communication Magazine, IEEE

[2] Open Networking Foundation https://www.opennetworking.org

[3] S.F. Hasan, Emerging Trends in Communication Networks, Springer Briefs in Electrical and Computer Engineering, DOI: 10.1007/978-3-319-07389-7_3

[4] Nunes, B.A.A. INRIA, Sophia-Antipolis, France Mendonca, M. ; Xuan-Nam Nguyen ; Obraczka, K. ; Turletti, T.

“A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks”, Vol. 16, Issue. 3, pp. 1617 – 1634, DOI:10.1109/SURV.2014.012214.00180, Communications Surveys & Tutorials, IEEE , 13 February 2014

[5] Zhijing Qi, Grit Denker, Carlo Giannelli, Paolo Bellavista, Nalini Venkatasubramanian

“A Software Defined Networking Architecture for the Internet-of-Things”, pp. 1-9, DOI:10.1109/NOMS.2014.6838365, 5-9 May 2014, Network Operations and Management Symposium (NOMS), 2014 IEEE, Krakow

[6] Wolfgang Braun and Michael Menth , “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices”, Future Internet 2014, 6, 302-336; doi:10.3390/fi6020302

[7] Bernardos, C.J. Telecommun. Eng., Univ. Carlos III de Madrid, Leganés, Spain de la Oliva, A. ; Serrano, P. ; Banchs, A. ; Contreras, L.M. ; Hao Jin ; Zúniga, J.C.,

“An architecture for software defined wireless networking”, Vol. 21, Issue: 3, pp. 52-61, DOI:10.1109/MWC.2014.6845049, June 2014, Wireless Communications, IEEE

[8] Dan Li, Yunfei Shang ; Congjie Chen, “Software defined green data center network with exclusive routing”, pp. 1743 – 1751, DOI: 10.1109/INFOCOM.2014.6848112, INFOCOM, 2014 Proceedings IEEE, April 27 2014-May 2 2014, Toronto, ON

[9] H. Kim and N. Feamster, “Improving network management with software defined networking,” Communications Magazine, IEEE, vol. 51, no. 2, pp. 114–119, 2013

[10] Shihabur Rahman Chowdhury, Md. Faizul Bari, Reaz Ahmed, Raouf BoutabaDavidR,“PayLess:A low cost network monitoring framework for Software Defined Networks”, pp. 1-9, DOI: 10.1109/NMOS.2014.6838227, ,Network OperationsandManagement Symposium (NOMS), IEEE ,5-9 May 2014, Krakow

[11] Yosr Jarraya, Taous Madi, Mourad Debbabi, “A Survey and a Layered Taxonomy of Software-DefinedNetworking”, pp.1955-1980, Vol.16, Issue:4,DOI: 10.1109/COMST.2014.2320094 , Communications Surveys & Tutorials, IEEE , 2014

[12] Syed Abdullah Shah, Jannet Faiz, Maham Farooq, Aamir Shafi, Syed Akbar Mehdi,“An Architectural Evaluation of SDN Controllers”, pp. 3504–3508,ISSN:1550-3607, DOI:10.1109/ICC.2013.6655093, Communications (ICC), IEEE International Conference on, 9-13 June 2013

[13] Kevin.Phemius, Mathieu.Bouet, Jeremie.Legua,“DISCO: Distributed Multi-domainSDNControllers”,pp.1-4, DOI:10.1109/NOMS.2014.6838330, Network Operations and Management Symposium (NOMS), IEEE, 5-9 May 2014, Krakow

[14] Mathieu Bouet, Kevin Phemius, and Jeremie Leguay,“Distributed SDN for Mission-critical Networks”, pp.942–948,

DOI: 10.1109/MILCOM.2014.162,

Military Communications Conference (MILCOM), IEEE, 6-8 Oct. 2014, Baltimore, MD, USA

[15] Yossi Kanizo, David Hay, Isaac Keslassy,“Palette: Distributing Tables in Software-Defined Networks”,

pp.545-549,DOI:10.1109/INFCOM.2013.6566832,INFOCOM Proceedings IEEE, 14-19 April 2013, Turin

[16] Gringeri S., Bitar, N. , Xia T.J., “Extending Software Defined Network Principles to Include Optical Transport”, Vol. 51, Issue. 3 , pp. 32-40, DOI: 10.1109/MCOM.2013.6476863, Communications Magazine, IEEE , March 2013

[17] Niels L. M. van Adrichem, Christian Doerr and Fernando A. Kuipers, “OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks”,pp.1-8, DOI:10.1109/NOMS.2014.6838228, Network Operations and Management Symposium (NOMS) IEEE, 5-9 May 2014

[18] Boyang Zhou Coll. of Comput. Sci., Zhejiang Univ., Hangzhou, China
Chunming Wu ; Xiaoyan Hong ; Ming Jiang, “Programming Network via Distributed Control in Software-Defined Networks”, pp.3051 – 3057, DOI:10.1109/ICC.2014.6883789, 10-14 June 2014,

Communications (ICC),IEEE International Conference on, Sydney, NSW

[19] Vijay Mann, Anilkumar Vishnoi, Kalapriya Kannan and Shivkumar Kalyanaraman,"CrossRoads: Seamless VM Mobility Across Data Centers through Software Defined Networking", pp. 88 – 96, DOI:10.1109/NOMS.2012.6211886, Network Operations and Management Symposium (NOMS), IEEE, 16-20 April 2012, Maui, HI

[20] Luca Veltri, Giacomo Morabito, Stefano Salsano, Nicola Blefari-Melazzi, Andrea Detti, "Supporting Information-Centric Functionality in Software Defined Networks", pp. 6645 – 6650, DOI: 10.1109/ICC.2012.6364916, Communications (ICC), IEEE International Conference on, 10-15 June 2012, Ottawa, ON

[21] Thomas Zinner, Michael Jarschel, Andreas Blenk, Florian Wamser, Wolfgang Kellerer, "Dynamic Application-Aware Resource Management Using Software-Defined Networking: Implementation Prospects and Challenges",pp.1 –6, DOI:10.1109/NOMS.2014.6838404, Network Operations and Management Symposium (NOMS), IEEE, 5-9 May 2014, Krakow

[22] "On Scalability of Software-Defined Networking", Volume:51 , Issue: 2, pp. 136 – 141, DOI:10.1109/MCOM.2013.6461198, Communications Magazine, IEEE

[23] Pinheiro, R.S. ; Comput. Sci. Post-graduation Program (PPGCC), Brazil ; Pinheiro, B.A. ; Esteves, R.P. ; Abelem, A.J.G.,"RepoSDN: An repository organization and coordination method of software defined networks applications", pp. 1-4, DOI: 10.1109/NOMS.2014.6838380, 5-9 May 2014, Network Operations and Management Symposium (NOMS), 2014 IEEE, Krakow

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson,J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation incampus networks," ACM SIGCOMM Computer Communication Review,vol. 38, no. 2, pp. 69–74, 2008.

[25] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrixestimator for openflow networks," in Passive and Active Measurement. Springer, 2010, pp. 201–210.

[26] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement withopensketch," in Proceedings 10th USENIX Symposium on NetworkedSystems Design and Implementation, NSDI, vol. 13, 2013.

[27] P. Van Mieghem and F. A. Kuipers, "Concepts of exact QoS routing algorithms," Networking, IEEE/ACM Transactions on, vol. 12, no. 5, pp. 851–864, 2004.

Shruthi S Manvachar, Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bangalore, India.