# Software Defined Networking – an Overview

B. Susila
Dept. of Communication Systems
M.Kumarasamy College of engineering
Karur, Tamilnadu.

C. Veeralakshmi
Dept. of ECE
M.Kumarasamy College of engineering
Karur, Tamilnadu.

***Abstract*— Today's networks have grown leap and bounds, demand for bandwidth increase every day. Multimedia, streaming and public cloud infrastructure has increased the demand for bandwidth. Ever changing network demands needs change in internet architecture. Decoupling of control plane and data plane helps to have reliable network architecture. But still the number of protocols makes it very difficult to have a direct control over the network. Virtualization plays a major role in computing, storage and networking. Network overlays achieve the ever changing demands of today's bandwidth intensive application. Software Defined Networking (SDN) has changed the way we look at the network. SDN truly divides the control plane and data plane and paves the way to have direct control over the networks. The programmability of data plane using simple match and action condition has simplified the way the data path is controlled by control plane. Network Function Virtualization (NFV) is complementary to SDN and it helps to virtualize the middlebox functionalities. This paper discusses the underlying concepts of Software Defined Networking.**

***Index Terms*— *Software defined networking, OpenFlow, control plane, data plane, virtualization, Programmable networks.***

## I. INTRODUCTION

The internet has led to the creation of digital society where everything is connected and it can be accessible from anywhere. Despite their wide spread adoption, the traditional IP networks are hard to manage and configure them based on the predefined policies. Traditional IP networks are dynamic and more complex to reconfigure incase of any faults, load and changes. Computer networks can be divided into three planes of functionality, which includes data, control and management planes. Control plane functions, such as participating in routing protocols, run in the architectural control element. Control plane is used to populate forwarding tables. It will also perform service provisioning, reachability information exchange and connectivity management. The Management plane protocols like SNMP can be used to monitor the device operation, its performance, interface counters etc. The data plane will forward transit traffic. The network policies are defined in management plane, the control plane enforces the policies and the data plane executes the policies by forwarding the data based on the defined policies.

In traditional IP networks the control plane and the data planes are tightly coupled. Network includes numerous types of devices such as routers, switches, firewalls and numerous types of middleboxes to perform various types of operations. The network operators are responsible for configuring and managing them. The network misconfiguration related errors are common in today's network. Misconfiguration of a single device may create a bigger problem in the network. The vertical integration of today's networks reduces the flexibility and making difficult to the evolution and innovation for the networking infrastructure.

The idea of programmable networks has been proposed to facilitate the network evolutions. Software Defined Networking (SDN) is an emerging new paradigm where the vertical integration of control and data planes is decoupled, and the control plane is programmable. Central software program called controller will act as a brain of the network, will direct the entire network behavior. SDN dramatically simplify the network management and improve the network innovation and evolution. In SDN the network intelligence is logically centralized by programming control planes and the data plane will act as just a forwarding device. The open interfaces are used to program the network (eg. ForCES [1], OpenFlow [2] etc).

The field of Software Defined Network is quite recent and growing very fast and it is gaining wide acceptance in industries as well as academia. SDN started as a cleanslate project in Stanford University as academic research and it's becoming new evolution in the networking world. SDN makes it easier to create and introduce new abstractions in networking, simplifying network management and facilitating network evolution.

In this article we explore the SDN architecture and implementation, various programming languages used for Programming the controller, Benefits of SDN, Security and SDN applications.

## II. SDN ARCHITECTURE AND ITS IMPLEMENTATION

The Open Networking Foundation (ONF) [3] is the group that is most associated with the development and standardization of SDN. The architecture of SDN is shown in figure.1. According to the ONF, "Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow protocol is a foundational element for building SDN solutions." The standard SDN architecture

[4] consist of Southbound APIs, Controller and Northbound APIs.

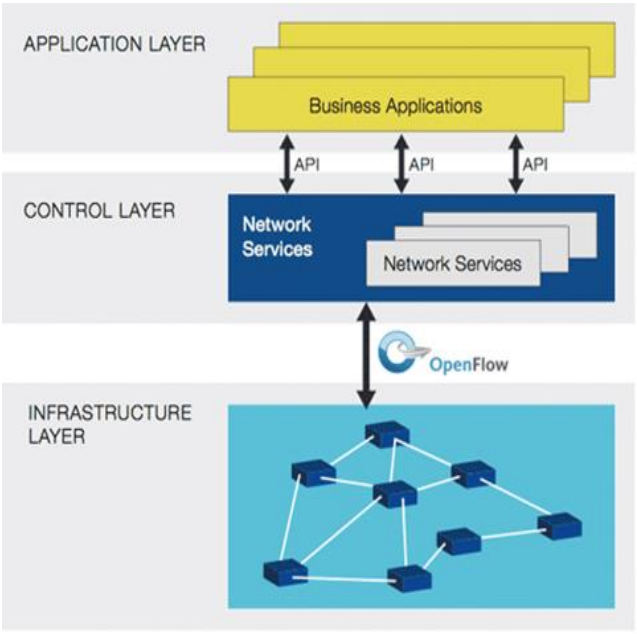


Fig.1 SDN Architecture

*Southbound APIs*: SDN protocols (OpenFlow, XMPP, BGP).
*Controller*: Considered as an operating system for networks, one that provides a centralized access to the entire network.
*Northbound APIs*: An interface for application developers to extract information about the network.

The SDN architecture has several advantages. The SDN architecture is

- Directly programmable.
- Agile.
- Centrally managed.
- Programmatically configured.
- Open standards-based and vendor-neutral.

Based on the SDN concept it can be defined by three levels of abstractions: (i) Forwarding (ii) Distribution and (iii) Specification.

*Forwarding abstraction* allows any forwarding behavior desired by the network applications. OpenFlow is a realization of one such abstraction.

*Distribution abstraction* making distributed control problem as centralized one. Network Operating System (NOS) is the realization of distributed abstraction.

*Specification abstraction* allows network application to express the desired behavior of the network. It can be achieved through the network virtualization and programming languages.

### *A. Infrastructure*

An SDN architecture consist of different layers each layer will perform specific functions. The infrastructure will discuss about different networking equipments. The SDN architecture is same as that of traditional networks composed of different networking equipments includes switches, routers and middlebox appliances. In SDN the logical intelligence is separated from the data plane and the controller is centralized. The configuration and communication between different control and data planes can be achieved by using the standard open interfaces like OpenFlow.

SDN architecture consist of two main elements the forwarding device and the controller. The forwarding device includes matching rules, matching packets and counters. The controller will define the handling of packets through a sequence of flow tables. A rule can be defined by combining different matching fields. The priority of rules will follow the sequence number.

OpenFlow enabled switches are available in the market have Ternary Content Addressable memory (TCAM) of up to 8K entries to store the forwarding rules from the controller. Software based Openflow switches includes OpenvSwitch, OFsoftSwitch, Pica8 to name a few.

### *B. OpenFlow*

With the OpenFlow protocol [5], a form of software-defined networking, a network can be managed as a whole rather than as a number of individual devices. OpenFlow moves the forwarding decision from individual switches to a controller, typically a server or workstation. The OpenFlow specification defines a protocol between the controller and the switches and a set of operations on the switches.

Each switch maintains a number of flow tables, with each table containing a list of flow entries. Table 1 shows the main components of a flow entry in a flow table.

TABLE 1: MAIN COMPONENTS OF A FLOW ENTRY IN A FLOW TABLE.

| Match Fields | Counter | Instructions |
|---|---|---|
| | | |

(1) *Match field*, to match against packets consist of the ingress port and packet headers, and optionally metadata specified by a previous table.
(2) *Counters* to update the matching packets such as reachability information, number of packet received and duration.
(3) *Set of instructions* used to handle the matching packets.

### *C. Data Plane Devices*

Hardware or software based forwarding devices will perform set of operations. The data plane devices are having well defined instruction sets used to perform operations based on the incoming packets. The operation includes drop or forward the packet, forward to the specific port etc. Generally the data plane devices are southbound API. Which includes OpenFlow [2] enabled switches, ForCES [1]. The forwarding devices can be connected by wireless channels and wired cables. The network consists of interconnected forwarding devices. The instruction set for forwarding devices are defined by southbound interfaces. It will also define the communication protocol for data plane devices.

The hybrid switches also used in current networks to perform the data plane operation. The operation includes (1)

Installing the forwarding rules, (2) Processing the forwarding rules. Installing the rules requires more memory. Memory is the main consideration in OpenFlow network. Normal open flow switches support only few hundred to few thousands of flow entries. TCAM memories are used to store the forwarding rules. TCAM is expensive and power consumable. Some proposals are applied to limit the memory issue of OpenFlow switches. DevoFlow is an extension to OpenFlow to address the memory issue. DIFANE [6], Palette [7] and One Big Switch [8] are some of the proposals to manage the forwarding rule space.

*D. Controller*

The controller is the brain of the network. The forwarding devices are programmed in control plane. Typically it is a northbound interface to abstract the low level instructions from southbound interfaces to program the forwarding devices. All control logic from applications and controllers will form the control plane.

The control plane can be programmed using high level languages such as java, python. The controller is also called as Network Operating System (NOS). Different SDN controller includes NOX [12], POX [16], Trema [17], floodlight [13], Open daylight [15] controller. The SDN controller typically contains number of pluggable modules that can perform different tasks. The controller can perform orchestration of new rules throughout the network and can perform analysis of running algorithms.

*E. Programmability of SDN Networks*

The software-defined networking (SDN) is developed to create an infrastructure that is much more agile and flexible. It should perform network automation and orchestration that better supports the dynamic changing demands of users, as well as the devices and data accessing the network. One of the ways SDN delivers this agility and flexibility is by making the network more programmable.

There are three use cases to defining what programmability means for SDN networks:

- **Adjusting the Flows** – Mainly focuses on protocols such as OpenFlow that enable SDN Controllers to interact with routers and switches in the forwarding plane to adjust the traffic flow in the SDN Networks. This helps networks respond to the dynamic changing.
- **Supporting the Applications** – Concerned with the coordination, automation, and exception handling of a network to, better align with the needs of the applications running on it. The languages such as JavaScript Object Notation (JSON) or Extensible Messaging and Presence Protocol (XMPP) can be used to support rapid deployment of new applications based on the automation configuration.
- **Automating SDN Networks** – This use case focuses on SDN networks doing what they are supposed to do without interference from a network administrator. When something changes, the network should figure out how to address the change automatically.

Much of the programmability of the network relies on the northbound and southbound open application programmable interfaces (APIs) communications between the SDN Controller and the applications and switches/routers, respectively. The programmability of the network enables better bandwidth utilization, performance, and operational efficiency.

## III. TOOLS AND PLATFORMS FOR SDN

SDN is currently used for new network evolution and innovation and it is gaining wide acceptance in the networking industry. This section will provide an overview of various open source SDN tools and platforms that enable research and experimentation with SDN technologies.

*A. Emulation and Simulation*

Emulation and Simulation is software used for testing and prototyping the topology without the need of expensive physical devices. Mininet [9] is the emulation tool which is commonly used in SDN. Mininet is the emulator which can generate larger network with limited resource. It provides a simple and inexpensive **network test bed** for developing OpenFlow applications. Mininet-HiFi is an evolution of mininet, it enhances the container based emulation to provide resource provisioning, performance isolation, and accurate performance monitoring.

The Simulation of OpenFlow devices can be achieved by NS-3 [10] simulation Software. SDN troubleshooting Simulator (STS) [11] is designed to apply and specify a number of test cases to examine the network.

*B. Programming Languages*

Programming languages usually referred as high level languages, each language has a syntax and unique set of keywords. The programmable networks provide the ability to program the network from machine level language to many high level languages like Java and Python. Different types of open controllers are used to program the network. Table 2 provides the list of open source controller with programming languages. The programmability in the networks move from low level language to high level language to promote the code reusability, simplifying the task and network virtualization.

TABLE.1 LIST OF OPEN SOURCE CONTROLLER WITH THE PROGRAMMING LANGUAGES

| Controller | Language | Description |
|---|---|---|
| POX [16] | Python | OpenFlow controller written in python runs under various network conditions. |
| NOX [12] | C++/Python | Openflow controller developed by Nicira networks, the first openflow controller. |
| Beacon [14] | Java | Java based openflow controller, cross-platform supports event based operations. |
| Floodlight [13] | Java | Java based controller based on beacon implementation. |
| Opendaylight [15] | Java | Hosted by Linux foundation it has no restrictions on the |

| | | |
|---|---|---|
| | | OS. |
| Trema [17] | C and Ruby | Developed by NEC, a framework for developing Openflow controller using Ruby and C. |
| Nodeflow [18] | Javascript | Independent developers. The controllers can be developed using Javascript. |

The programming languages can provide the specialized abstraction to survive other network management requirements such as combining the results, monitoring, and counter polling. The programming languages provide the portability to the developers so they do not need to re-implement the application for different controller platforms.

To avoid overlapping of rules and efficiently express the packet forwarding rules other SDN programming languages such as Pyretic, Frenetic and Netcore were used to perform simultaneous and parallel operations.

## IV. NETWORK FUNCTION VIRTUALIZATION

Network functions virtualization (NFV) offers a new way to design, deploy and manage networking services. NFV decouples the network functions, such as network address translation (NAT), firewalling, intrusion detection, domain name service (DNS), and caching, to name a few, from proprietary hardware appliances so they can run in software.

It's designed to consolidate and deliver the networking components needed to support a fully virtualized infrastructure including virtual servers, storage, and even other networks. It utilizes standard IT virtualization technologies that run on high-volume service, switch and storage hardware to virtualize, the network functions. It is applicable to any data plane processing or control plane function in both wired and wireless network infrastructures.

The growing adoption of software-defined networking (SDN), network functions virtualization (NFV), and network virtualization have begun the trend toward modernizing the IP Infrastructure that runs today's networks. SDN, NFV, and network virtualization, as well as white box switching, are all fundamental pieces of a new type of IP Infrastructure being deployed today. We define the key technologies for this next generation of IP Infrastructure as:

- **SDN** offers a centralized way to orchestration and control the network. A key component to SDN is an SDN Controller which has the ability to act as the brain of the network. The SDN Controller receives information and orchestrates traffic on the network to switches and routers via southbound APIs, and to the applications with northbound APIs. Openflow is the protocol which is mostly used by SDN networks.
- **NFV** is the virtualization of network services. NFV came to realization when service providers attempted to speed up deployment of new network services in order to bring up revenue and growth plans. Noticing that hardware-based appliances limited their ability to achieve these goals, they looked to standard IT virtualization technologies and discovered that NFV helped accelerate service innovation and provisioning. This led to the creation of ETSI NFV, which sets NFV basic requirements and architecture.
- **Network virtualization** (NV) creates a logical, virtual network, by decoupling network functions from the hardware that deliver them. All network functionality is separated from the underlying hardware and simulated as a "virtual instance" that can be loaded onto general, off-the-shelf platforms; a single hardware platform can be used to support multiple virtual network instances.
- **White box** networking are network devices, such as switches and routers, that are based on "generic" merchant silicon networking chip set available for anyone to buy, as opposed to proprietary silicon chips designed by and for a single networking vendor.

## V. SDN SECURITY

As software-defined networking (SDN) environments continue to grow, security is essential to protect the availability and privacy of all connected resources and information, but securing SDN can be difficult. We can predict different attack vectors on SDN system. The SDN security concern includes security at different layers of SDN architecture.

Data plane layer, the attacker can target the different networking elements within the network. An attacker can gain unauthorized access to the network and destabilize the network elements and this type attack is called fuzzing attack. To secure the data plane layer the organizations prefer to use Transport Layer Security (TLS) to authenticate the traffic between network element and the controller. The other protocols SNMPv3 and SSH can also be used to authenticate the network elements and encrypt the data between them.

The controller layer is the main target to the attacker. The controller is the central decision point, it should be secured tightly. The availability of the SDN controller should be maintained. The security should be deployed, maintained and controlled to develop an environment with more scalable, efficient and secure. The next generation environments created a new category called Software Defined Security (SDSec), which decouples the security control plane from the security processing and forwarding planes and it can be maintained as single logical system. The SDSec will function like Network Function Virtualization (NFV).

## VI. BENEFITS OF SOFTWARE-DEFINED NETWORKING

Following are the benefits offered by the Software Defined Networking.

**Automation** — SDN allows for automation of complex operational tasks that make networks faster, more efficient and easier to manage.

**Increased uptime** — SDN has proven effective in reducing deployment and configuration errors that can lead to service disruptions.

**Less drain on resources** — SDN gives administrators control over how their routers and switches will operate in a single and virtual workflow. This helps the key staff to focus on more important tasks.

**Better visibility** — With SDN, system administrator's gain improved visibility into overall network function, allowing them to allocate resources more effectively.

**Cost savings** — SDN can lead to significant overall costs savings. It also reduces the cost of the devices among the network infrastructure.

## VII. SDN APPLICATIONS

An SDN application is a software program designed to perform a task in a software defined networking (SDN) environment. SDN applications can replace and expand upon functions that are implemented through firmware in hardware devices in a conventional networking environment. SDN application definition can be extended to data plane, control plane, network functions and L4-L7 layer applications.

The Software Defined Networking can be deployed in any traditional networks from home network to enterprise networks. SDN has been proposed for new innovation in the networking environment that can provide more flexibility and extensibility in the network architecture. New features and services can be added to SDN based network by upgrading the control plane. Network features and services can be added dynamically in the form of network application as a management function. The application includes Traffic engineering, Server load balancing, Virtualization, and inter domain routing.

### *A. Traffic Engineering*

There are several traffic engineering methods that has been proposed which includes ALTO [19], Hedera [20], QoS framework [22], QNOX [21]. The main goal of this application includes minimizing power consumption, maximizing network utilization, optimized load balancing, and traffic optimization.

Load balancing was the main application envisioned in the OpenFlow based SDN. Several techniques and algorithms have been proposed for this purpose [23]. The main goal of this application is to provide scalability. The Wild card based rules is a technique to provide scalability and to perform proactive load balancing [23]. The wild card rules can be used in cloud based networks. In the context of data center synchronization and communication effective resource utilization can be achieved with SDN and OpenFlow. SDN eases the introduction of new protocol in the network. IP multicasting [22] can be implemented in the control plane using OpenFlow by handling multicast request in the control software. The control software will install the forwarding rules in the switches according to the multicast application.SDN can also used to provide fully automated system for controlling the configuration of network devices. It is useful in the scenario of virtual aggregation it allows the network operators to reduce the data replication in the routing table. Another interesting application for large scale service provider is Traffic optimization. The other applications that are provided by traffic engineering include application-aware video streaming, QoS Provisioning.

### *B. Wireless and Mobility*

The distributed control plane in the current wireless networks it is hard to manage limited spectrum, allocating the resources, handling the interference, and efficient load balancing between the cells. SDN based approach is an opportunity to ease and manage the different wireless networks like cellular networks [24] and WLAN [25]. The dynamic spectrum usage, load balancing, creation of virtual access points, seamless hand-over, inter-cell interference coordination,Qos and access control policies, deployment of new applications can be achieved through SDN. Realizing these features can be achieved by allowing programmability in the stack layer of wireless networks. A heterogeneous wireless network is also a target for SDN.

### *C. Optical Transport Networks*

SDN technology is largely used in Cloud and Large data center application. Now the research interest has turned to extend the programmability in the optical transport networks [26]. The advantages of SDN include optimizing the network resource usage, simplify the service creation and specify resource requirements. SDN must be pushed beyond the capability to handle the Ethernet network infrastructure and architectures of Optical networks.

The transport networks were built to support long distance communication such as voice and data traffic with high reliability. Transport networks are more complex, they vary in terms of architecture and the type of cross connects which act as switches between links. SONET- a ring based network offers more reliability due to their ability to switch the traffic from working path to protection path in case of any failure. Optical Cross connects terminate links and switch the data streams towards destination. Add-drop multiplexers add or drop the data from multi wavelength data stream and redirect it to the destination. Reconfigurable Optical ADMs provide more flexibility in the network.

A single source to destination path may cross multiple links. The SDN controller should recognize the capability and restriction of network architecture and equipment. The controller should optimize the resource requirements and the resource allocation. SDN can be applied to multilayer architecture in optical networks.

## VIII. CONCLUSION

Traditional IP networks are hard to manage. One of the reasons is that it is vendor specific and the control plane and data planes are tightly coupled. Each line of products has its own configuration and management interfaces. It will lead to long cycles for product update. Software Defined Networking (SDN) created an opportunity to solve this long-standing problem. It provides the dynamic programmability of network using open interfaces by decoupling the control and data planes and the global view of network through logically centralized network controller. Further research in Software defined networking environment works towards extending SDN capabilities to carrier transport network, cloud computing paradigm.

## ACKNOWLEDGMENT

I would like to thank authors, mentioned in the references which are citied below for their valuable research works which helped me to gain knowledge. And also I thank my guide for her precious guidance.

## REFERENCES

[1] A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong,R. Gopal, and J. Halpern. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810 (Proposed Standard), March 2010.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson,J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review,38(2):69–74, 2008.

[3] Open networking foundation https://www.opennetworking.org/about.

[4] ONF, "Software-Defined Networking: The New Norm for Networks,"Mar.13,2012,https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdnnewnorm.pdf.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[6] M. Yu, J. Rexford, M. J. Freedman, and J. Wang. Scalable Flow-Based Networking with DIFANE. In Proc. SIGCOMM, 2010.

[7] Yossi Kanizo, David Hay, and Isaac Keslassy. Palette: Distributing tables in software-defined networks. In INFOCOM, pages 545–549,2013.

[8] Nanxi Kang, Zhenming Liu, Jennifer Rexford, and David Walker. Optimizing the one big switch abstraction in software-defined networks.

[9] Mininet at https://github.com/mininet/mininet/wiki/Introduction to mininet.

[10] Ns-3 project, "ns-3: OpenFlow switch support," 2013. [Online]. Available: http://www.nsnam.org/docs/release/3.13/models/html/openflowswitch.html

[11] ucb-sts, "STS - SDN troubleshooting simulator," 2013. [Online]. Available: http://ucb-sts.github.io/sts/

[12] Gude, N.; Koponen, T.; Pettit, J.; Pfaff, B.; Casado, M.; McKeown, N.; Shenker, S. NOX:Towards an Operating System for Networks. ACM SIGCOMM Comput. Commun. Rev. 2008, 38, 105–110.

[13] Floodlight, an open sdn controller. http://floodlight.openflowhub.org/.

[14] Beacon. https://openflow.stanford.edu/display/Beacon/Home.

[15] http://www.opendaylight.org/. Opendaylight, 2013.

[16] Pox. http://www.noxrepo.org/pox/about-pox/.

[17] Trema openflow controller framework. https://github.com/trema/trema.

[18] The nodeflow openflow controller. http://garyberger.net/?p=537.

[19] M. Scharf, V. Gurbani, T. Voith, M. Stein, W. Roome, G. Soprovich,and V. Hilt, "Dynamic VPN optimization by ALTO guidance," in Software Defined Networks (EWSDN), 2013 Second European Workshop on, Oct 2013, pp. 13–18.

[20] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in Proceedings of the 7th USENIX conference on Networked systems design and implementation, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 19–19.

[21] K. Jeong, J. Kim, and Y.-T. Kim, "QoS-aware Network Operating System for software defined networking with Generalized OpenFlows," in Network Operations and Management Symposium (NOMS), 2012 IEEE, april 2012, pp. 1167 –1174.

[22] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable QoS control for network convergence," in Proceedings of the 2010 internet network management conference on Research on enterprise networking, ser. INM/WREN'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–1.

[23] N. Handigol, S. Seetharaman, M. Flajslik, A. Gember, N. McKeown, G. Parulkar, A. Akella, N. Feamster, R. Clark, A. Krishnamurthy, V. Brajkovic, and T. A. and, "Aster*x: Load-Balancing Web Traffic over Wide-Area Networks," 2009.

[24] V. Chandrasekhar, J. Andrews, and A. Gatherer, "Femtocell networks: a survey," Communications Magazine, IEEE, vol. 46, no. 9, pp. 59–67, September 2008.

[25] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," in Proceedings of the first workshop on Hot topics in software defined networks, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 115–120.

[26] Optical transport working group otwg. In Open Networking Foundation ONF, 2013.