Investigating the Security Aspect of Software Defined Networking (SDN)

Aliyu Lawal Aliyu, Peter Bull & Ali Abdallah
School of Computing, Telecommunications and Networks
Faculty of Computing, Engineering and the Built Environment,
Birmingham City University

E-mail: aliyu.lawalaliyu@mail.bcu.ac.uk

Abstract: *The complexity associated with current networks is quite restrictive. New network applications cannot be deployed without major network disruption and associate cost. Policy implementation and management of large networks becomes a tedious task with heterogeneous devices that are physically distributed. And to keep in pace with recent advances in network technology a lot is spent in capital expenditure and operational expenditures. These make the network evolve more slowly. A new network paradigm called software defined networking (SDN) is set to address most of these issues by separating the vertical integration between the control logic and the forwarding function, thereby making networks programmable to suit application needs and foster network evolution.*

SDN reduces network complexity and make networks more flexible, providing a logically centralised controller that manages physically distributed systems across the network. SDN emerged from projects on control to data plane separation and programmable networks. But the separation of the control logic from forwarding functions introduced new threats, not present in traditional networks. There is a need to mitigate these new threats and ensure availability and dependability, in order to fasten the transition of traditional network to SDN. This paper gives a brief history of projects that led to realisation of SDN and a layered security aspect of SDN is presented focusing on vulnerabilities that exploit the communication channel between logically centralised controller and underlying forwarding element.

Keywords: Software Define Networks, Architecture , Security and Programmable networks

Introduction

The pace at which networking technologies evolve is claimed by Jarraya *et al.* (2014) and Ghodsi *et al.* (2011) to be slow as compared to other communication technologies. Similarly Kreutz *et al.* (2014a) and Huang *et al.* (2013a) state that the slow evolution stems from vendors producing proprietary firmware and protocols that communicate only with their products. This leads to a more closed communication environment which hinders innovation and creativity in networking technologies. Huang *et al* (2014b) and Benson *et al* (2009) lament that the architectural choices made by

vendors make it difficult for new network applications and services to be integrated in to the existing network infrastructure without additional capital, operational and management cost.

Due to these restrictive behaviours Latifi *et al.* (2014) and Hamadi *et al.* (2014) state that researchers and network operators have been looking for a solution that will give them optimum flexibility and control of their network independent of vendor specifications and limitations by protocols. Several authors, including Huang *et al.* (2014b) and Ahmed and Boutaba (2014) pinpoint that the reason behind the limitations of the current networks lies within the vertical integration of the three logical planes in networking devices which are management, control and data as shown in Figure 1. This vertical integration is constricting, because the control plane that makes the decision on how to handle network traffic is tightly coupled with the data plane which form the (infrastructural layer), and this layer works based on the decision made by the control plane. The management plane include programs and utilities like Simple Network Management Protocol (SNMP) and Secure Shell (SSH) that are used remotely to configure and monitor networks.

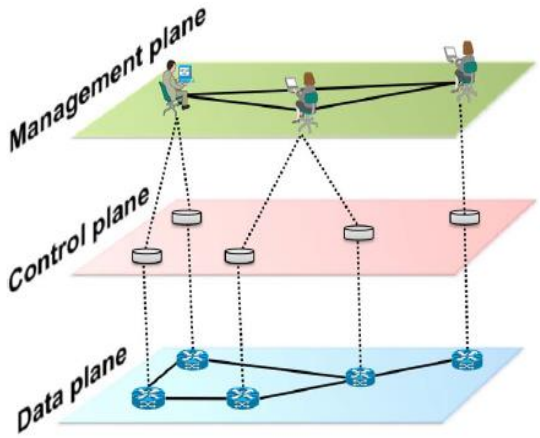

Figure1: Conventional network (Kreutz *et al*., 2014a)

A promising solution to these problems shown by several authors, including Huang *et al.* (2013a); Kreutz *et al.* (2014a); Shin and Gu (2013) and Hamadi *et al.* (2014), is a new network paradigm called Software Defined Networking (SDN) which revisits the logical architecture of networking device and decouples the vertical integration that bundled the control and data plane together thereby providing scalability, flexibility, manageability, and reduce operational and capital expenditure cost. As described by Jain *et al.* (2013), the vertical integration which bundled the network intelligence and semantics to the networking devices in the data plane is now separated, as shown in Figure 2, and the intelligence is handled by a logically centralised controller in the control layer.

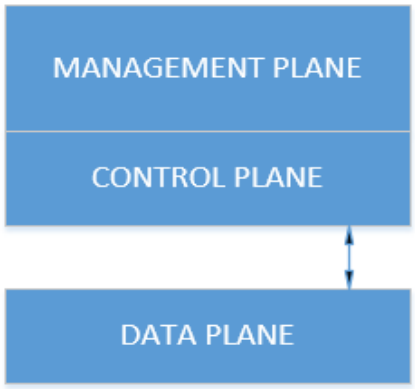

Figure 2: SDN Network breaking vertical integration

As stated by Ahmed and Boutaba (2014) the full architecture of SDN comprises additional sub-interfaces between the major layers, called the southbound interface and northbound interface. The southbound interfaces sits between the control and infrastructural layer while the northbound interface lies between the control and application layers. Jarraya *et al* (2014) note that for high availability and scalability implementation, westbound and

eastbound Application Programming Interfaces (APIs) exist that allow multi-domain network control. Figure 4 shows the complete architecture of SDN.

According to Mckeown *et al.* (2008) and Feamster *et al.* (2013), the controller programs the data plane devices in the infrastructural layers through the southbound interface. Many protocols are available in the south band interface that encapsulate the messaging signal between the control layer and infrastructural layer. Kreutz *et al.* (2014) described these protocols, examples are Openflow, Revised Openflow Library (ROFL), Opflex, Openstate, Forwarding and Control Element Separation (FORCES), Protocol Oblivious Forwarding (POF), Path Computation Element Protocol (PCEP) and Open Virtual Switch Database (OVSDB). Figure 3 shows some of the supported protocols in the southbound interface.

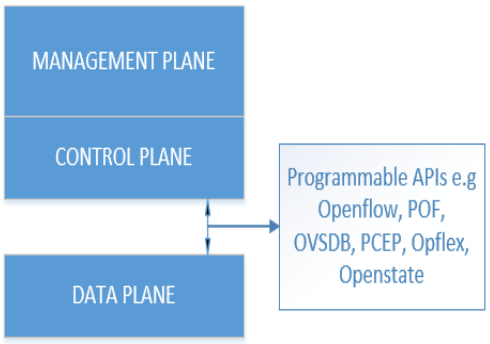

Figure 3: Southbound Interface protocols

Of the many available protocols, the *de facto* and industry standard protocol used for network programmability, as mentioned by Kloti *et al.* (2013) and Scott-Hayward *et al* (2013) is the *openflow*. The protocol has a large community support base of network researchers, academia, enterprise and individuals that contribute toward its standardisation and application in network control and operations. On the other hand the northbound API does not have a standardised interface for network orchestration but makes use of the REST API to push network application requirements like routing, load balancing and firewall (Fw), Intrusion Detection and Prevention system (IDPS) through the controller. The controller translates those requirements into commands to the forwarding elements.

Figure 4 presents the SDN architecture.

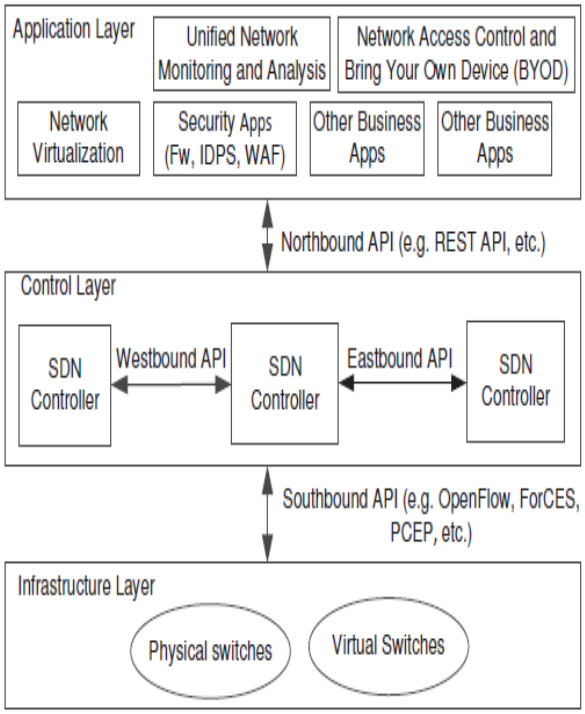

Figure 4: SDN Architecture Jarraya *et al.* (2014).

The separation and programmablity which is new to communication networks introduced some vulnerabilities which were not present in traditional networks as explained by Shin and Gu (2013) and Kloti *et al* (2013). Furthermore, Kreutz *et al.* (2013b) note some new threats that arise due to introduction of new entities in the network by SDN. These threats impede the transition of enterprise, organisations, academia etc. to migrate fully into SDN.

The International Data Corporation (IDC) shows that thte SDN global market will rise from $960 million in 2014 to $8billion in 2016, however the critical factor to reach this target lies within the maturity of SDN which include *security*, interoperability, reliability and advancement in design and implementation of various SDN components.

History of Software Defined Networking (SDN)

The idea of SDN stems from two main projects which are *control to data plane separation* and *programmable networks* (Feamster *et al.* 2014, Jarraya *et al.* 2014). Most of the work in the various projects was independent but complementary, because the main driver is to have a network that provides optimum control while still maintaining efficiency, performance and scalability. The project came about because of the constricting nature of conventional networks where orchestration, network flexibility, maintenance and operation are costly, cumbersome, tedious and difficult to attain.

Control to Data Plane Separation Projects

According to Kreutz *et al.* (2014a), the earliest projects that advocated control-data plane separation are Network Control Point (NCP), Routing Control Platform (RCP), Secure Architecture for Network Enterprise (SANE) and Ethane. These will be discussed separately.

Network Control Point (NCP)

NCP is one of the first initiative to separate control from the data plane, as mentioned by Sheinbein and Weber (1982). It was introduced by the telecommunication company AT&T in the early 1980s. NCP helps reduce network complexity and ease management by providing a global view of the network. Likewise Feamster *et al* (2004) add that NCP supports rapid introduction of new services.

Routing Control Platform (RCP)

Caesar *et al.* (2005) mentioned that RCP provides a logically centralised framework through which optimal route selection is carried out on behalf of routers and selected routes are forwarded to the routers. This ability helps in reachability exchange between multiple domains to enable scalability and evolution of routing architecture. However, Feamster *et al.* (2004) point out that RCP uses Border Gateway Protocol (BGP) to install routing paths in routers, which allows immediate deployment and serves as a logically centralised proxy for route injections in routers only. But Greenberg *et al.* (2005) claim that RCP considers only BGP which is a single protocol that is part of the data plane which has other protocols that forward traffic through the network.

Secure Architecture for Network Enterprise (SANE)

SANE is an architecture that deals with security aspect of a separate control and forwarding framework as claimed by Scott-Hayward *et al.* (2013). SANE enforces security policies like Intrusion detection, firewall, access control etc. in network through a logically centralised server situated in a single protection layer (Jarraya *et al.*, 2014). Moreover Feamster *et al* (2013) state that SANE produce a logically centralised flow-level solution for access control in an enterprise network. However Casado *et al.* (2006) point out that, at the time of its proposal, SANE was considered as an extreme approach to enterprise because a logical centralised

controller is responsible for policy enforcement and host authentication.

ETHANE

ETHANE built on SANE adds two components the first of which is the controller that has a global view of the network topology with the corresponding network policy and the second component comprises of simple Ethane switches that receive flow information through a secure channel from the controller (Feamster *et al.,* 2013). According to Mckeown *et al*. (2008), the deployment of Ethane in Stanford University set the ground for creation of *Openflow*, which is the key enabler of SDN functionality. But Scott-Hayward *et al* (2013) highlight the drawback of Ethane where application traffic can compromise network policy.

Summary

SDN leverages from most of these projects like the global view of the network provided by SDN is an initiative of NCP and Ethane. The logical centralisation of controller and network security policy for flow rule insertion was inherited by SDN from Ethane, RCP and SANE. However Feamster *et al.* (2013) pinpoint that all the efforts of control-data plane separation projects depend solely on existing routing protocols. Much of the needed functionalities required for flexibility like dropping, flooding or modifying of packets are absent and they do not allow for matching of the header field. These imposed limitations on the side of programmable controllers to support range of network applications. SDN differs in that it provides a protocol independent forwarding, and is not centred toward solving problems of a specific network architecture, which is the case in many control–data separation projects like NCP and RCP. It is a holistic approach that can be used to solve network problem across all network architectures including cloud architectures, mobile telecommunication networks and broadcast networks.

Programmable Network Projects

The principal idea of programmable networks is to realise a dynamic, flexible and customisable network Van der Merwe *et al.,* 1998). The main programmable network projects are *active networks* from computer networks and *opensignalling* (OpenSig) from telecommunication networks (Jarraya *et al.,* 2014). The idea of programmability complements separation of the two planes because separation decouples the control logic from data while programmability allows the control logic to define network requirement globally in terms of instructional codes and send them down to forwarding elements in other to control packet flows and network operation.

Active Networks

The idea of active networks emerged in 1994 from the United States Defence Advanced Research Projects Agency (DARPA) in their quest to determine the future direction of networking systems. Lazar *et al.* (1996) described active networks as a novel approach to network architecture where networking devices carry out customised computations on the traffic flowing through them. After execution, the behaviour of the networking device changed and provided different level of fine-grained control. In addition, active networks allow for new network service deployment at run time which gives it a high level of dynamism. The implementation of active networks is seen in the project of Tennenhouse (1997) where nodes can compute or modify content of a packet. The approach uses *programmable switches* and *capsules.* The

former do not affect or change existing packets and support switching devices that are cable of accepting programmes and with corresponding instructions on how packets should be processed. The latter suggest that tiny programs should replace packets which are encapsulated in transmission frames and executed at each node along their path.

Open signalling (OpenSig)
OpenSig proposed to control networks through a set of well-defined programmable network interfaces and distributed programming environments (Campbell *et al.* 1998). The principle behind it is to make networks programmable like personal computers (PCs), thereby allowing flexible deployment of new network services (e.g. mobility management, routing, handover etc.). This paved the way for third-party software providers to enter the market. However Jarraya *et al* (2013) justify why OpenSig suffers in the hands of vendors who are not willing to expose their interfaces for third party software programmability.

Summary
Limitations were introduced by Active networks and OpenSig complexity, such as isolation, performance and security issues. They require nodes to process each packet separately which bring performance bottlenecks and also execution of code to be performed at infrastructure level where the network devices are not designed to operate in that fashion unless they undergo a major upgrade. The upgrades require current network devices to add supported functionalities necessary for programmability and to provide high processing in CPU which is costly, disruptive and brings more power dissipation in network devices. It also requires redesigning of Application Specific Integrated Circuit (ASIC) chipsets which will complement the processes needed by fine grained programmability. It also depends on vendors being willing to to expose their internal architectural setting through some well-defined API and drift away from a closed networking system. This idea was rejected by major vendors in the market which consequently impedes industrial adoption and research to go in that direction. SDN differs in the sense that it does not require vendors to expose their internal settings to provide programmability. What is required from the networking device is to provide support for an SDN agent (e.g openflow) and the network will be customisable to suit operator needs. SDN does not require per device computation which brings additional overhead. Instead, the functionality is delegated to an external controller and the networking devices are just passive hardware gears with forwarding capability based on instruction sets defined by the centralised controller.

Security Issues in SDN
Threat vectors and security weakness are identified in SDN architecture (Kloti *et al.* 2013; Wasserman and Hartman, 2013). However Shin and Gu (2013) argue that only threat targeting control-data plane communication are SDN specific; other threat vectors only affect the conventional networks. Kreutz *et al.* (2014) mentioned that other threats are independent of technology or protocol (e.g FORCES, PCEP and OpenFlow) because they can be found at different levels of SDN architecture. As shown in Figure 5 there are seven identified threat vectors. The first indicates the possibility that bogus or fake traffic can be generated to overwhelm data plane device and controller by intruder. The second exploits the

vulnerabilities in a networking device and a lunch offensive attack against the network. The third is SDN specific and exploits the open communication channel between controller and forwarding elements to eavesdrop and monitor link communication. The fourth is significant and SDN specific because it exploits the vulnerabilities in controllers to take over the control function. A compromised controller leads to the whole network being compromised because the network only responds to instruction sets defined globally by the controller. The fifth is SDN specific, which is malicious applications that are developed and deployed on SDN controllers. The sixth arises due to compromise of the management station on which the controller runs, this indirectly compromises controller operation. The last threat is lack of remediation and forensics in SDN networks which make it difficult to recover after breach and trace intruders. Table 1 provides the summary of the different threat vectors.

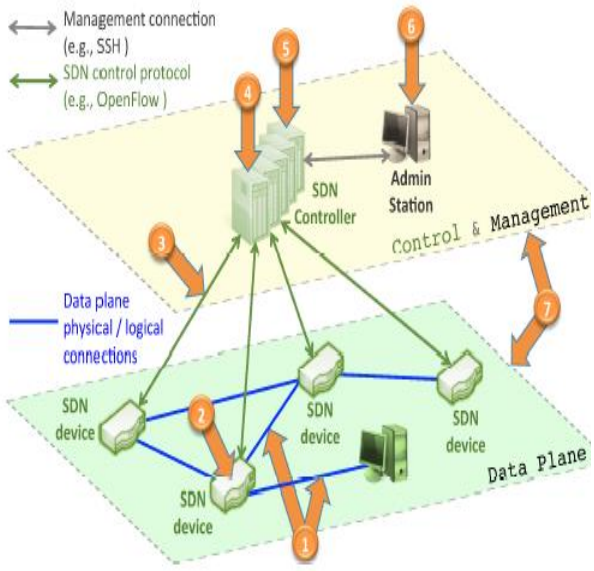

Figure 5: Threat Vectors of SDN Architecture (Kreutz *et al.*, 2013b)

Threat vectors	Specific to SDN?	Consequences in software-defined networks
Vector 1	no	Open door for DDoS attacks.
Vector 2	no	Potential attack inflation.
Vector 3	yes	Exploiting logically centralized controllers.
Vector 4	yes	Compromised controller may compromise the entire network.
Vector 5	yes	Development and deployment of malicious applications on controllers.
Vector 6	no	Potential attack inflation.
Vector 7	no	Negative impact on fast recovery and fault diagnosis.

Table 1: Threat Vectors (Kreutz *et al.*, 2013b)

Since SDN is a new network paradigm at an early stage with the potential of becoming the next generation communication network, much is expected on the side of availability and dependability (Ros and Ruiz, 2014). Scott-Hayward *et al.* (2013) mentioned that every secure communication network should guarantee confidentiality, integrity, availability, authentication and non-repudiation. This cannot be achieved without having concrete threat mitigation techniques in SDN. We now discuss the security issues in details with respect to different layers of SDN architecture.

Infrastructural Layer

Denial of service attack seems to be most feared at this layer (Kreutz *et al.*, 2013b) because an attacker (adversary) can take over control of a device and use it to send large amount of new packet flows that require the attention of the controller. This will stop the controller from responding to legitimate new flow requests. Braga *et al* (2010) state that most DoS attacks are difficult to detect due to their similarities with legitimate network traffic; hence they provide a mechanism of traffic detection using Self Organising Maps (SOM). SOM is an artificial intelligence method that

uses traffic rate limiting mechanisms like average of packets per flow and average of bytes per flow to detect whether certain traffic is fake or legitimate. However the limiting factor of SOM is that it has to be trained and be familiar with network traffic.

According to Scott-Hayward *et al.* (2013), another threat that resides within the interface data and control layer can lead to Man-in-the-Middle-Attack or eavesdropping as shown in Figure 5. However Benton *et al* (2013) claim that the threat can be mitigated using Transport layer Security (TLS) which installs a root certificate that would establish a secure protocol handshake between controller and switch.

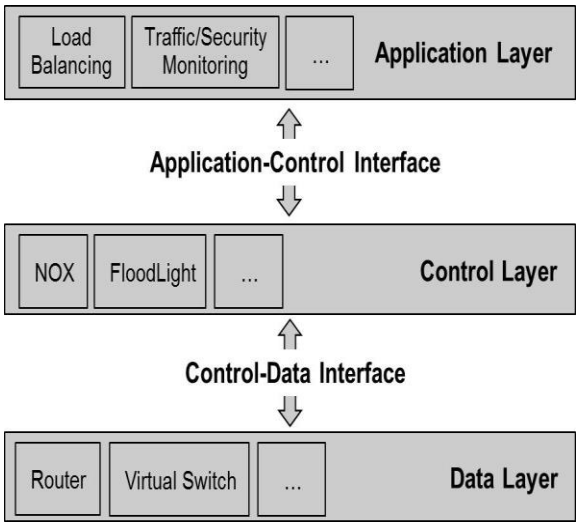

Figure 6: SDN Architecture with interface between layers (Scott-Hayward *et al.*, 2013)

Control Layer

One of the well-known vulnerabilities at this layer is compromising the controller through a denial of service (DoS) attack, which can have a catastrophic effect on the entire network. To mitigate this issue Shin *et al* (2013) proposed AVANT-GUARD, which is a mechanism that stops control plane saturation from the effect of TCP-SYNC flood and network mapping. But the framework failed to account for other protocols such as User Datagram Protocol (UDP) or Internet Control Message Protocol (ICMP). However Kreutz, *et al.* (2013b) suggest the replication of the controller to allow for fault tolerance in case the main controller is compromised. This would increase availability and at the same time provide security. Likewise to maintain trust between controller and switches, Benton *et al.* (2013) proposed Flowvisor, which serves as a proxy between the layers and rewrites flow rules.

Application Layer

As identified by Jarraya *et al*. (2014), the vulnerability in this layer is the trust between the controller and the application. SDN operate policies that allow business applications to apply changes in the network and there are no formal verification technique or semantics to assess the trust of these applications. Malicious applications can exploit this vulnerability and disrupt network operation. However, Porras *et al*. (2012) proposed a security enforcement kernel called FortNOX that addresses the issue of trust by implementing a role-based authentication mechanism and setting severities between applications to restrict privileges. In a different way Sonkoly *et al.* (2012) and Sherwood *et al.* (2010) view network virtualisation isolation as a slice in the application layer as another mechanism that contain threats in a single slice and prevent security breaches from propagating to other slices. The isolation aids in threat containment, however containment does not guarantee mitigation - it only reduces the size of the affected network slice.

Kloti *et al.* (2013) and Shin and Gu (2013) identified that Openflow networks have various dependability and security issues

like information disclosure, denial of service and elevation of privileges. However Bentont *et al.* (2013) and Porras *et al* (2012) emphasised that the absence of mitigation mechanisms like access control, intrusion detection systems, isolation and security recommendations is a major cause of these vulnerabilities.

Conclusion

The research aims at providing a secure control layer communication with data plane devices and network applications. There is a lack of counter security measures in SDN architecture that can stop denial of service (DoS) attack between the control layer and the infrastructural layer, and the architecture lacks mechanisms to authenticate and dictate how applications should behave in the network, because a malicious application can take down the entire network operation if allowed to run without any form of access control. All these problems arise due to the separation of control from the data plane which, consequently, introduced new threats. Several authors make suggestions on how the security issues should be dealt with but most of their contributions lack implementation and consideration; and the SDN architecture still remains insecure. From the literature, this research will try to cover up the noticeable gap from previous surveys and make the solutions practically realisable to attain a more secure SDN architecture.

References

Ahmed, R. & Boutaba, R. (2014) 'Design Considerations for Managing Wide Area Software Defined Networks', *Communication Magazine IEEE*. Vol. 52, No. 7, pp. 116–123.

Benton, K., Camp, L. J. & Small, C. (2013) 'OpenFlow vulnerability Assessment', *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. New York, NY: ACM pp. 151–152.

Benson, T., Akella, A., & Maltz, D. (2009) 'Unravelling the Complexity of Network Management,' *Proceedings of the 6th USENIX symposium on Networked systems design and implementation (NSDI'09)*. Berkeley, CA: ACM pp. 335-348.

Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A. & van der Merwe, J. (2005) 'Design and implementation of a routing control platform', *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA: ACM. Vol. 2, pp. 15–28.

Casado, M., Garfinkel, T., Akella, A., Freedman, M. J., Boneh, D. McKeown, N. & Shenker, S. (2006) 'SANE: A protection architecture for enterprise networks', *Proceedings of the 15th conference on USENIX Security Symposium* – Vol. 15, No. 10. Berkeley, CA, USA: ACM pp. 1-15

Campbell, A. T. (1998) 'Open Signaling for ATM, Internet and Mobile Networks (OPENSIG'98)' *SIGCOMM Computer Communication Review*, Vol. 29, No.1, pp. 97 – 108.

DARPA (1997) 'Active network program' http://www.sds.lcs.mit.edu/darpa-activenet/ Accessed: 30th December 2014

Feamster, N., Rexford, J. & Zegura, E. (2014) 'The Road to SDN', *SIGCOMM Computer Communication Review* Vol. 44 No. 2, New York, NY, USA: ACM. pp. 87-98

Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Rexford, J., Xie, G., Yan, H., Zhang, J. & Zhang, H. (2005) 'A Clean Slate 4D Approach to Network Control and Management', *SIGCOMM Computer Communication Review* Vol. 35, No. 5, pp. 41–54.

Ghodsi, A., Berkeley, K. T. H. U. C., Shenker, S., Berkeley, I. U. C., Singla, A. & Wilcox, J. (2011) 'Intelligent Design Enables Architectural Evolution', *Proceedings of the 10th ACM Workshop on Hot Topics in Networks.* No. 3. New York, NY: ACM pp. 1–6

Huang, W. Y., Liu, T. L.., Chou, T.-Y. & Hu, J. W. (2014) 'Automatical End to End Topology Discovery and Flow Viewer on SDN', *28th IEEE International Conference on Advanced Information Networking and Applications Workshops*, Victorian: IEEE pp.910–915.

Huang, D. Y., Yocum, K., & Snoeren, A. C. (2013) 'High-fidelity switch models for software-defined network emulation', *Proceedings of the Second SIGCOMM Workshop on Hot Topics in Software Defined Networking.* New York, NY: ACM, pp. 43-48.

Hamadi, S., Snaiki, I., & Cherkaoui, O. (2014) 'Fast path acceleration for open vSwitch in overlay networks', *Global Information Infrastructure and Networking Symposium (GIIS)*. Montreal, QC: IEEE, pp.1–5.

IDC Corporate (2014) 'Growing SDN Momentum Presents Fresh Opportunities for Data Center Networks' http://www.idc.com/getdoc.jsp?containerId=prUS25052314 [Accessed 30th December 2014].

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Holzle, U., Stuart, S., & Vahdat, A., (2013) B4: experience with a globally-deployed software defined wan, In Proceedings of the SIGCOMM conference, New York, NY, USA: ACM, pp. 3–14.

Jarraya, Y., Madi, T., & Debbabi, M. (2014). A Survey and a Layered Taxonomy of Software-Defined Networking. *Communications Surveys & Tutorials, IEEE* vol. 16, no. 4, pp. 1955-1980

Kreautz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S. (2014) Software-Defined Networking: A Comprehensive Survey. *Proceedings IEEE* Vol. 103, no. 1.pp 14-76

Kreutz, D., Ramos, F. M. & Verissimo, P. (2013) 'Towards secure and dependable software-defined networks;, *Proceedings of the second SIGCOMM workshop on hot topics in software defined networking*, New York, NY: ACM, pp. 55–60.

Kloti, R., Kotronis, V. & Smith, P. (2013) 'OpenFlow: A security analysis; *International Conference on Network Protocols (ICNP)*. Goettingen, Germany: IEEE, pp. 1 – 6

Lantz, B., Heller, B., & McKeown, N. (2010) 'A network in a laptop: rapid prototyping for software-defined networks', *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, No. 19. New York, NY: ACM, pp.1–6.

Latifi, S. & Durresi, A. (2014) 'Emulating Enterprise Network Environments for Fast Transition to Software-Defined Networking', *Mediterranean Conference on Embedded Computing (MECO),* Budva: IEEE pp.294–297.

Lazar, A., Lim, K. S. & Marconcini, F. (1996) 'Realizing a foundation for programmability of ATM networks with the binding architecture, *Journal on Selected Areas in Communications*, Vol. 14, No. 7, pp. 1214–1227.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. & Turner, J. (2008) 'OpenFlow: enabling innovation in campus networks', *SIGCOMM Computer Communication Revolution*, Vol. 38, No. 2, New York, NY: ACM pp. 69–74.

Porras, P. Shin, S., Yegneswaran, V., Fong, M., Tyson, M. & Gu, G. (2012) 'A security enforcement kernel for OpenFlow network', *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. New York, NY: ACM, pp. 121–126.

Ros, F. J. & Ruiz, P.M. (2014) 'Five nines of southbound reliability in software-defined networks', *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. New York, NY: ACM, pp. 31–36.

Sherwood, R., Chan, M., Covington, A., Gibb, G., Flaislik, M. Handigol, N., Huang, T. Y. Kazemian, P., Kobayashi, M., Naous, J. Seetharaman, S., Underhill, D., Yabe, T., Yap, K.-K., Ylakoumis. Y. Zeg, H., Appenzeller, G., Johari, R., McKeown, N., Parulkar, G. & Kobayashi, M. (2010) 'Carving research slices out of your production networks with OpenFlow', *SIGCOMM Computer Communication Review*, Vol. 40, No. 1, New York, NY: ACM, pp.129-130.

Scott-Hayward, S., O'Callaghan, G., & Sezer, S. (2013) 'SDN Security: A Survey, *Conference of SDN for Future Networks and Services (SDN4FNS*), New York, NY: IEEE pp.1–7.

Sheinbein, D. & Weber, R. P. (1982) '800 service using SPC network capability', *The Bell System Technical Journal, Alcatel-Lucent* Vol. 61, No. 7. pp. 1745-1757.

Shin, S. & Gu, G. (2013) 'Attacking software-defined networks: A first feasibility study', *ACM Proceedings of the second workshop on Hot topics in software defined networks*. New York, NY: ACM, pp. 1–2.

Shin, S., Yegneswaran, V., Porras, P. & Gu. G. (2013) 'AVANTGUARD: Scalable and Vigilant Switch Flow Management in Software-defined Networks', *Proceedings of the 2013 SIGSAC conference on Computer & communications security.* New York, NY: ACM, pp. 413–424

Song, H. (2013) 'Protocol-oblivious Forwarding: Unleash the power of SDN through a future-proof forwarding plane', *Proceedings of the Second SIGCOMM Workshop on Hot Topics in Software Defined Networking*. New York, NY: ACM, pp. 127–132.

Sonkoly, B., Gulyas, A., Nemeth, F., Czentye, J., Kurucz, K., Novak, B., & Vaszkun, G. (2012) 'OpenFlow Virtualization Framework with Advanced Capabilities,' *Proceeding of the European Workshop on Software Defined Networking*. Darmstadt: IEEE pp. 18–23.

Tennenhouse, D. Smith, J. Sincoskie, W. Wetherall, D. & Minden, G. (1997) 'A survey of active network research', *Communications Magazine*, Vol. 35, No. 1, pp. 80–86.

Van der Merwe, J., Rooney, S., Leslie, I., & Crosby, S. (1998) 'The tempest practical framework for network programmability', *Network*, Vol. 12, No. 3, pp. 20–28.

Wasserman, M. & Hartman, S. (2013) 'Security analysis of the open networking foundation (onf) OpenFlow switch specification', *Internet Engineering Task Force*, April. [Online]. Available at: https://datatracker.ietf.org/doc/draft-mrw-sdnsec-openflow-analysis/ [Accessed 10th December 2014].