Software Defined Network: Architecture and Programming Language Survey

Article *in* International Journal of Pure and Applied Mathematics · January 2018

CITATIONS
4

READS
3,765

3 authors:

Mustafa Hasan Albowarab
Technical University of Malaysia Malacca
6 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Nurul Azma Zakaria
UniversitiTeknikal Malaysia Melaka
52 PUBLICATIONS **125** CITATIONS
SEE PROFILE

Zaheera Zainal Abidin
Technical University of Malaysia Malacca
58 PUBLICATIONS **317** CITATIONS
SEE PROFILE

International Journal of Pure and Applied Mathematics
Volume 119 No. 18 2018, 561-572
ISSN: 1314-3395 (on-line version)
url: http://www.acadpubl.eu/hub/
Special Issue

http://www.acadpubl.eu/hub/

Software Defined Network: Architecture and Programming Language Survey

[1]Mustafa Hasan Albowarab, [2]Nurul Azma Zakaria and [3]Zaheera Zainal Abidin

[1,2,3]Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia.

Abstract

Nowadays, most of the things are connected and from anywhere can be accessible because of the wide-speared of the internet over the entire world. This growth of the internet led to complex of control and manages the IP networks. Further, the networks are challenging to establish the network according to rearrange policies. Software defined networking (SDN) can thereby stabilize these issues by giving new functions of controlling and managing to the complete network systems. We present in this paper a review on SDN. The first phase of this paper is the introducing and architecture for SDN, describe its core theories and in what way it differs from other common Networking. Further we explain about SDN routs, and the standardization. In the second phase we explain the significant structure layers of SDN construction and the layers of SDN method. Finally, SDN review is concluded in the last section of this paper.

Key Words: Software defined networking (SDN), SDN Architecture, SDN programming language.

1. Introduction

Networking becomes more complex, with the help of cloud computing, massive datacenters and virtualization. Improved methods and better networks to manage and switch them are searching by the administrator. Throughout the control plane of the network has been separated by the software-defined networks (SDNs) from the plane of data. APIs which are in SDNs allow administrators to manage the resources of networks through services and applications.SDN is a manageable, cost-effective, adaptable and dynamic emerging architecture. This has made it perfect for the dynamic nature and high-bandwidth of recent applications.

According to the architecture of SDN, the intelligence and state of network has been centralized rationally, data and control planes have been decoupled and the basic infrastructure has been withdrawn from the application.
There are four pillars that define the SDN network architecture based on:

1. The plane of control has been decoupled from the plane of data as stated above.
2. Packet forwarding is made based on flows in contrast to the decision based forwarding in the traditional networks. Counters; Match fields and a set of commands are contained in a flow entry .
3. The controller of SDN is in authority for the centralized control logic. Existing Popular controllers of SDN contain OpenDaylight1, ONOS2, Floodlight3 and POX4.
4. Networks of SDN are programmable. Therefore, the possibility of developing software via an interface to administer data plane is high.

The particular architectural features of SDN (Ghodsi et al., 2011),(Jarraya, Madi and Debbabi, 2014) have been reviewed in a few papers nowadays. There is general idea of OpenFlow and a brief background in (Ghodsi et al., 2011) and. A quite justified 3 blocks collected of high-level network controller, services and the switch interface are presented in these OpenFlow-oriented reviews(Jarraya, Madi and Debbabi, 2014). Nevertheless, in the same way to the earlier works, the survey does not give a demanding treatment of SDN important aspects. Essentially, present surveys do not have detailed discussion of the SDN important building blocks. Take for example, programming languages, interfaces and the network operating systems (NOSs). They also lack of study of cross-layer problems such as dependability, scalability and reliability. There is no complete current research efforts, tasks, and related standardization activities overview. First and foremost, the context is explained, the motivation for SDN is introduced, the new paradigm concept is explained and by what method it be different from the old networking. In the beginning of the study, it is focus on explaining SDN is not as different as a advance technological. Certainly, its presence is fixed at the joint of the traditional concepts, technology drivers, and recent and future requirements. The consequence of the high interest of industry and the possibility to adjust the

networking status quo from numerous views, many standardization efforts around SDN are continuing.

SDN Architecture Planes

In this section of the paper the SDN architecture planes are explained, it is allocated into three major layers, as illustrated in Figure 1.

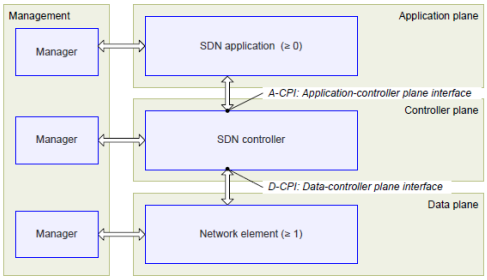

Figure 1: SDN Architecture Planes

SDN Application Layer: The SDN Application Layer The programs in the application layer or plane is applications of SDN, which insist on business applications or services of network and describes a service-aware performance of network resources for manner of programmatic. These programs connect their requirements and response of network by relating with the SDN controller plane through the Northbound Interface (NBI). Therefore, plane of controller can modify the network resources activities automatically. The global abstract network view of the network resources are accessed by the programs in the application plane, for their purposes of internal decision-making, given by the controller plane of SDN by using models of data displayed via the Northbound Interface.

By exposing an info model instances for use by other applications, a plane of application might act as a model server of data. Officially, the further applications are clients that connect to the application server agent of SDN. An entity of application plane will turn as an information model client by functioning on an info model instance displayed by an entity of server. This entity of server might be a controller of SDN. An entity of application plane may turn in both parts at the same time. Take for example; a path computation engine (PCE) may count on controller of SDN for virtual network topology info, while offering the controller of SDN a PCE (Paper, 2012).

SDN Control Layer: it also called the control plane, which is located under the application layer. It is in charge of programming and managing the forwarding plane. Then, it accesses the information given by the forwarding plane and the network routing and operation are defined. One or more controller of software that connects with the forwarding network elements through standardized interfaces is comprised.

Data plane: The data plane comprises a set of one or more network elements, each of which contains a set of traffic forwarding or traffic processing resources. Resources are always abstractions of underlying physical capabilities or entities.

Load Balance System Design

Tables of routing contain data of destination network and routers can only compute the information of next-hop network without the global view of network routing. Nevertheless, the capability to display the network global view at the early stage of network construction is obtained by the controller of SDN. Entirely paths between each resource intersection to every destination intersection are updated by the controller of SDN in order to update global network topology information. Every global path load condition can be assessed by using this global network superiority in SDN. Figure 2 shows the network construction for suggested load balance system of SDN.

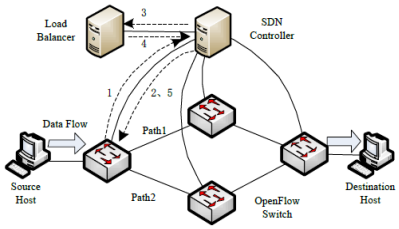

Figure 2: Network Architecture of SDN Load Balance System

Building Blocks

The switch of SDN, the controller of SDN, interfaces of southbound and northbound and SDN applications are the fundamental building blocks of the SDN architecture. The following paragraphs provide details for each of the above components.

Switch: The talented way for data centers and infrastructure of virtualized network is software switches (Weissberger, 2013). Examples of software based OpenFlow switch implementations consists of Switch Light (Floodlight Controller|Big Switch Networks, Inc., 2018), ofsoftswitch13, Open vSwitch(Open vSwitch,2018), OpenFlow Reference, Pica8, Pantou, and XorPlus (Open vSwitch,2018).The quantity of ports of virtual access is bigger than ports of physical access on data centers is shown in current reports. Open vSwitch which consisted in software switches are used to move network functions to the edge, as a result allowing network virtualization (Koponen et al., 2014).

Controller: The SDN Controller is a rational object that obtains requirements and instructions from the application layer of SDN and transmits them to the components of networking. The controller is also in charge of taking out information about the network from the hardware devices and connecting it back to the applications of SDN. The flow entries can be added, updated,

deleted by the controller while using the southband API. The control plane of SDN is represented by the controller of SDN

The architecture of SDN does not require the internal design of a controller of SDN. It could be a set of identical procedures organized to share load or protect one another from failures. Components of controller can be executed on computer platforms. They may also execute on distributed resources such as virtual machines in data centers. It can say that the controller of SDN is assumed to have global scope. Besides, its components are assumed to share their state and information with the controller. Numerous components of controller may have joint write access to resources of network. In this situation, they must be constructed to control disjoint sets of resources or actions and synchronized with each other thus they never issue conflicting commands.

Southbound Interface: Southbound APIs assist effective control over the network and allow the controller of SDN to make changes based on the real-time demands and requirements. OpenFlow is the first and perhaps most famous southbound interface. It is an industry standard that describes the way the controller of SDN should work together with the forwarding plane to control the network(Mckeown et al., 2008). Extensible Markup Language (XML) is used by NetConf to connect with the routers and switches to set up and construct changes.

OpenFlow: The protocol of OpenFlow can be noticed as one potential implementation of controller-switch (southbound) interactions, as the interaction between the controllers of network and switching hardware is described. An abstraction for applications of business is provided to use facility which is given by the control layer (Kobayashi et al., 2013). Even though one of the basic concepts of SDN is to avoid vendor locking, dependence on one protocol does not serve this purpose.

The OpenFlow switch is the elementary forwarding element, which is accessible through the protocol and interface of OpenFlow. Construction of Flow-based SDN such as OpenFlow might need extra forwarding table entries, buffer space, and statistical counters, although this structure, would appear to simplify the switching hardware at the first glance , as shown in Figure 3 (Lara et al., 2014). An OpenFlow switch contains a flow table, which makes packet lookup and forwarding. Each flow table in the switch holds a set of flow entries that contain header fields or match fields and counters.

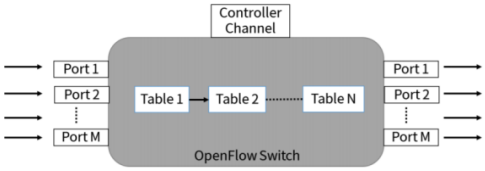

Figure 3: OpenFlow Switch Architecture

Programming Languages in SDNs

Languages of programming are increased rapidly for years. Both industry and academia have languages of programming. The progressions toward more convenient and reusable code are driven an important shift on the industry of computer(ACM and 2008, no date), (Farooq et al., no date).

Respectively, networks programmability is beginning to transfer from low-level machine languages to languages of high-level programming(Mckeown et al., 2008),(Foster et al., no date),(Monsanto et al., 2012),(Voellmy and Hudak, 2011),(Monsanto et al., no date),(Hinrichs et al., 2009),(Voellmy et al., no date),. Assembly-like machine languages, fundamentally impersonator the behavior of forwarding devices, forcing developers to use too much time on low-level details rather than on the problem solving. Programs of Raw OpenFlow have agreement with detailed behavior of hardware such as the importance collection of rules, overlapping rules and data plane inconsistencies that arise from in-flight packets whose flow rules are under installation (workshop and 2013, no date), (Song and Haoyu, 2013),(Li et al., 2017). (Monsanto et al., 2012), (Mckeown et al., 2008),

A high-programming languages provides abstractions can considerably help address many of the challenges of these lower level instruction sets (Farooq et al., no date), (Voellmy and Hudak, 2011),(Voellmy et al., no date), (Pentikousis et al., 2013),(Monsanto et al., no date),. In SDNs, high-level programming languages can be deliberate and used to:

1. Abstractionof high level is generated for make simpler the task of programming forwarding devices;
2. The productive and problem-focused environments are allowed for programmers of network software, speedy up innovation and growth;
3. reusability of code and modularization of software are promoted in the network control plane of network;
4. the development of network virtualization is fostered.

Numerous tasks can be lectured by languages of programming in SDNs. Take for example, it is tough to make sure that numerous tasks of an application do not affect each other in pure OpenFlowbased SDNs. Rules created for a challenges should not override the functionality of another challenges (Ferguson, Liang and Fonseca, 2012), (Nelson et al., 2014). Another example is when numerous applications run on a single controller (Katta, Rexford and Walker, no date),(Shin et al., 2013),(Son and Porras, 2013),(Yu and Wundsam, 2014),(Chua, 2012). Normally, rules are generated by each application according to its own requirements and rules. As a result, inconsistent policies can be created and set up in devices of forwarding, which can make difficulties for network operation. Languages of programming and systems of runtime can assist to resolve these issues. Significant techniques of software design such as reusability and modularity of code are difficult to accomplish using low-level models of programming(Fellow, no date).

The competence of writing and generating programs for virtual network topologies('Towards software-friendly networks', no date), (Gutz and Foster, no date). is other interesting feature which languages abstraction of programming provided. This idea is same as the object-oriented programming, where specific data and function for application creators are abstracted, simplify it to emphasis on explaining a specific issues. Take for example, in context of SDN, one can think of generating simplified virtual network topologies which be the whole network instead of installing and creating policies in every forwarding devices. Therefore, the developer of application ought to be able to abstract the network as a huge switch, rather than a grouping of some basic physical devices.

The languages of programming have to be in charge of creating and setting up the lower level directions necessary at each forwarding device to enforce the policy of users via the network. Developing a routing application turn into a direct procedure while with this type of abstraction.In the same way, a set of virtual switches represents a physical switch, every single of them belonging to a dissimilar virtual network. It is much harder for these two examples of abstract network topologies to implement with low level instruction set. On the other hand, abstraction for topologies of virtual network is provided by a language of programming.

High-Level Programming Languages of SDN: Language of high-level programming will become more influential tools which is for giving and implementing concepts for dissimilar significant properties and purposes of SDN. Low-level instruction sets suffer from numerous issues. Higher level languages of programming have been suggested, with varied goals in order to address some of these tasks, such as:

- concepts that allow dissimilar challenges of management to be achieved through network policies are provided;
- concepts that allow dissimilar challenges of management to be achieved through network policies are provided;
- several challenges are decoupled;
- interfaces of higher level programming are implemented to avoid low-level instruction sets;
- forwarding policies issues are solved;

2. Discussion

Since year 2011, the review discovered that the need for study for the SDN topic combined with tasks and effects increased. There is no relevant article found in 2010 and one relevant article has been found in 2011. There is an increase in year 2012 which are seven relevant articles have been found.

A general idea of the known tasks and SDN effects is shown in Figure 1. The implementation is addressed by most of the papers as a challenges. The high

risk of changing outdated constructions of network are counted in this category, it has been studied and discussed and most often. The category of demand the second highest. The know-how topic is explained in the third category. This category subsumed the controlling and administrating software defined networks with the existing staff.

The software unique features shown in Figure 1 defined networking are discussed the most in the articles studied. Fundamentals like decoupling hardware from the software and the entire network construction global view, take place in this category. The second category which is management is a significant part while explaining effects of software defined networks compared with outdated networks. The network maintenance and programmability are included in this key category. The economic factors are discussed in the last category. These trends tell us that recent study is more emphasis on consideration of scientific and technical.

3. Conclusion

Conventional networks are complicated and inflexible to accomplish. SDN generated a chance for resolving these problems. Particular of the significantconcepts of SDN are the overview of dynamic programmability in forwarding devices concluded open,the decoupling of the control southbound interfaces, and data plane. In spite of latest and interesting tries to review this new section in the history of networks(Nunes et al., 2014), the literature was still absent, to the best of our knowledge, a single extensive and comprehensive overview of the building blocks, concepts, and challenges of SDNs. this paper used a layered approach to systematically dissect the state of the art in terms of concepts, ideas, and components of the SDN.

References

[1] ACM, M. G.-C. of the and 2008, undefined ‘Education Paving the way for computational thinking’, dl.acm.org. Available at: https://dl.acm.org/citation.cfm?id=1378713 (Accessed: 2 June 2018).

[2] Chua, R. (2012) ‘OpenFlow northbound API: A new olympic sport’. Available at: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Chua%2C+R.%2C+2012.+OpenFlow+northbound+API%3A+A+new+olympic+sport.&btnG= (Accessed: 2 June 2018).

[3] Computer, S. O.- and 2013, undefined ‘Software-defined networking: On the verge of a breakthrough?’, pdfs.semanticscholar.org. Available at: https://pdfs.semanticscholar.org/1fe2/657ac876d2b4f87f2472cb9b62393f83be2d.pdf (Accessed: 4 June 2018).

[4] Farooq, M. et al. 'An evaluation framework and comparative analysis of the widely used first programming languages', journals.plos.org. Available at: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0088941 (Accessed: 2 June 2018).

[5] Fellow, C. I.'Modular SDN Programming with Pyretic', pp. 40–47.

[6] Ferguson, A. D., Liang, C. and Fonseca, R. (2012) 'Hierarchical Policies for Software Defined Networks', pp. 37–42.

[7] Floodlight Controller | Big Switch Networks, Inc. Available at: https://www.bigswitch.com/tags/floodlight-controller (Accessed: 7 June 2018).

[8] Foster, N. et al. 'Frenetic: A network programming language', dl.acm.org. Available at: https://dl.acm.org/citation.cfm?id=2034812 (Accessed: 2 June 2018).

[9] Ghodsi, A. et al. (2011) 'Intelligent Design Enables Architectural Evolution'.

[10] Gutz, S. and Foster, N. 'Splendid Isolation : A Slice Abstraction for Software-Defined Networks'.

[11] Hinrichs, T. L. et al. (2009) 'Practical Declarative Network Management'.

[12] Jarraya, Y., Madi, T. and Debbabi, M. (2014) 'A Survey and a Layered Taxonomy of Software-Defined Networking', IEEE Communications Surveys & Tutorials, 16(4), pp. 1955–1980. doi: 10.1109/COMST.2014.2320094.

[13] Katta, N. P., Rexford, J. and Walker, D. 'Logic Programming for Software-Defined Networks', (1), pp. 2–4.

[14] Kobayashi, M. et al. (2013) 'Maturing of OpenFlow and Software-defined Networking through deployments', COMPUTER NETWORKS. Elsevier B.V. doi: 10.1016/j.bjp.2013.10.011.

[15] Koponen, T. et al. 'Network Virtualization in Multi-tenant Datacenters.', usenix.org. Available at:

https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-koponen.pdf (Accessed: 4 June 2018).

[16] Lara, A. et al. (2014) 'Network Innovation using OpenFlow : A Survey Network Innovation using OpenFlow : A Survey'. doi: 10.1109/SURV.2013.081313.00105.

[17] Li, S. et al. (2017) 'Protocol Oblivious Forwarding (POF): Software-Defined Networking with Enhanced Programmability',

IEEE Network, 31(2), pp. 58–66. doi: 10.1109/MNET.2017.1600030NM.

[18] Mckeown, N. et al. (2008) 'OpenFlow : Enabling Innovation in Campus Networks'.

[19] Monsanto, C. et al. (2012) 'A Compiler and Run-time System for Network Programming Languages'.

[20] Monsanto, C. et al. (no date) 'Composing Software Defined Networks.', usenix.org. Available at: https://www.usenix.org/system/files/tech-schedule/nsdi13-proceedings.pdf#page=10 (Accessed: 4 June 2018).

[21] Nelson, T. et al. (2014) 'Tierless Programming and Reasoning for Software-Defined Networks This paper is included in the Proceedings of the Tierless Programming and Reasoning for Software-Defined Networks'.

[22] Nunes, B. A. A. et al. (2014) 'A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks', pp. 1–18.

[23] Open vSwitch (no date). Available at: https://www.openvswitch.org/ (Accessed: 7 June 2018).

[24] Paper, O. N. F. W. (2012) 'Software-Defined Networking : The New Norm for Networks'.

[25] Pentikousis, K. et al. (2013) 'MobileFlow : Toward Software-Defined Mobile Networks', (July), pp. 44–53.

[26] Shin, S. et al. (2013) 'FRESCO : Modular Composable Security Services for Software-Defined Networks', 2(February).

[27] Software-Defined Networking (SDN) Definition - Open Networking Foundation. https://www.opennetworking.org/sdn-definition/ (Accessed: 4 June 2018).

[28] Son, S. and Porras, P. (2013) 'Model Checking Invariant Security Properties in OpenFlow', pp. 1974–1979.

[29] Song, H. and Haoyu (2013) 'Protocol-oblivious forwarding', in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13. New York, New York, USA: ACM Press.

[30] 'Towards software-friendly networks' dl.acm.org. Available at: https://dl.acm.org/citation.cfm?id=1851288 (Accessed: 2 June 2018).

[31] Turull, D. et al. 'Evaluating OpenFlow in libnetvirt', diva-portal.org.

http://www.diva-portal.org/smash/record.jsf?pid=diva2:564963 (Accessed: 2 June 2018).

[32] Voellmy, A. et al. 'Procera: a language for high-level reactive network control', dl.acm.org.

https://dl.acm.org/citation.cfm?id=2342451.

[33] Voellmy, A. and Hudak, P. (2011) 'Nettle : Taking the Sting Out of Programming Network Routers', pp. 235–236.

[34] Weissberger, A. (2013) 'VMware's Network Virtualization Poses Huge Threat to Data Center Switch Fabric Vendors'. Available at: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=VMware's+network+virtualization+poses+huge+threat+to+data+center+switch+fabric+vendors&btnG= (Accessed: 3 June 2018).

[35] workshop, H. S.-P. of the second A. S. and 2013, undefined 'Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane', dl.acm.org. Available at: https://dl.acm.org/citation.cfm?id=2491190.

[36] Yu, M. and Wundsam, A. (2014) 'NOSIX : A Lightweight Portability Layer for the SDN OS', 44(2), pp. 29–35.