International Journal of Advancements in Technology

Khan, Int J Adv Technol 2017, 8:1
DOI: 10.4172/0976-4860.1000177

Review Article Open Access

Software-Defined Networking Reviewed Model

Muhammad Faisal Imran Khan*

Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh

***Corresponding author:** Muhammad Faisal Imran Khan, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh, Tel: + 8801941417305; E-mail: buetcse110@gmail.com

Received date: November 14, 2016; **Accepted date:** November 29, 2016; **Publication date:** January 01, 2017

Abstract

Software-defined Networking (SDN) is one new promising development in between Cloud Computing and Networking industries. As, SDN is quite new, there are many challenges in implementing it in traditional networking environment. In this paper, my aim is to describe current SDN implementation challenges in detail and proposing a reviewed model for SDN, where many of these challenges are solved.

Keywords: OpenFlow; Software-defined networking; Network security; Reliable networking; Scalability

Introduction

Software-defined networking (SDN) is a networking architecture purporting to be dynamic, manageable, cost-effective and adaptable, seeking to be suitable for the high-bandwidth, dynamic nature of today's applications [1].

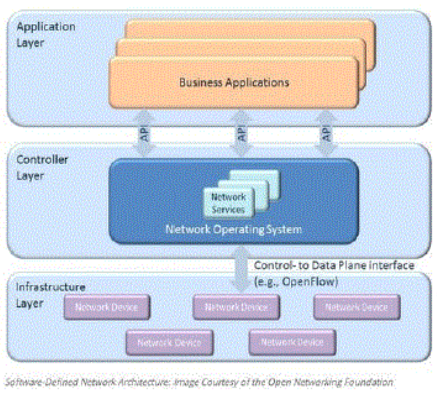

Figure 1: Software-defined networking structure [2].

The principal of SDN is to separate the control plane from the data plane and thus centralize network intelligence. In this model, a networking device does not have intelligence on its own but depends on a separate software which will have control over similar many physical devices (Figure 1).

SDN is an agile model of networking because abstracting control from data plane lets administrators dynamically adjust network-wide traffic flow to meet changing needs. It is also flexible because SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs [1].

One framework for SDN is OpenFlow (Figure 2). In OpenFlow framework, the central part is a software named Controller [3]. It is where intelligence of routing decisions, routing table generation and packet traffic control are generated. It manages the hardware based OpenFlow switches via OpenFlow protocol. OpenFlow switch uses the routing table generated by Controller to determine where the data packets should be sent to. This routing table is called Flow Table. Though it is generated by Controller, it stays in the switch. Controller sends control message to switch and switch enters a rule in the Flow Table based on this message. Initially the Flow Table is empty. When a new packet is received by a switch, it is matched with the existing flow entries in the Flow Table. If no match, it is sent to Controller to be processed. Then controller makes decision on how to handle the packet like – should it be dropped or make a new rule in the Flow Table to deal with this packet and similar packets received in the future.

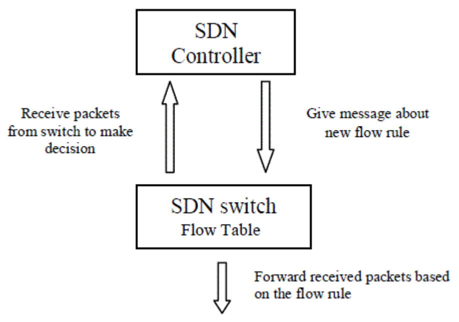
SDN Controller
Receive packets from switch to make decision
Give message about new flow rule
SDN switch
Flow Table
Forward received packets based on the flow rule

Figure 2: OpenFlow data traffic model.

Though SDN promise a simplified network management by centralized control, SDN need research in issues like – network security, compatibility, scalability, reliability, Controller placement, wireless integration, Flow Table management, performance under latency constraints, difficulties regarding language of the software in SDN etc. These issues can limit the network usage significantly when it is implemented in large scale network like in a supercomputing data center across the country or continents. There are solutions for few of these issues and others are still open question to researchers. These complicated research issues of Software-defined Networking (SDN) are discussed, an illustration for proposed model of SDN based on recommendations is presented then the conclusion.

SDN Research Issues

Security

In Software-defined Networking (SDN) model, the control plane is separated from the data plane, and the increased gap between two planes has made the network more attack prone compared to traditional networks. The existing security technologies for the traditional hop by hop networks also cannot be integrated with the OpenFlow model. Due to simplicity in SDN architecture, Controller cannot poke around every packet to detect presence of malicious programs [4,5].

One of the well known risks of SDN platform, Controller is prone to DOS attacks, which can be devastating for the entire network (Figure 3). By creating a huge number of new and unnecessary flows, third party can overwhelm the Controller, as Controller cannot decide quickly about how to handle these enormous flows. Such saturated Controller cannot control the original data traffic [4].

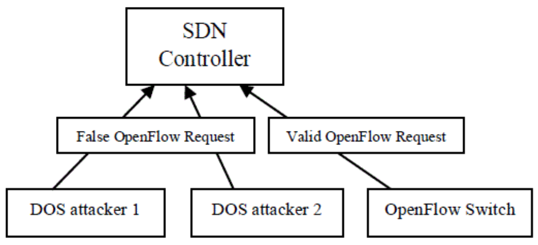

Figure 3: SDN is prone to DOS attacks.

Switch software compatibility

OpenFlow switches use embedded software from vendors that are mainly needed to process messages sent by the Controller and configure the Flow Table accordingly [4]. This piece of software needs to be complaint with the OpenFlow specification. But this specification is still ambiguous and can have several interpretations to different vendors. This could make producing compatible switch program from different vendors very difficult (Figure 4).

Scalability

When the network scales up in the number of switches and the number of end hosts, the SDN Controller can become less effective to control data traffic [5]. As more requests will be queued to the Controller, it may not be able to handle them all.

As the flow entries are generated by Controller and stored in switch, the performance of the SDN data traffic depends on switch resources (CPU, memory, etc.) and Controller (software) performance. When new message from Controller come to switch and it is updating the switch forwarding information base (FIB) by creating a new rule, a delay can occur in handling other subsequent packets received in the same switch.

In one solution, DevoFlow [5], micro-flows are managed in data plane and more massive flows in the Controller, meaning that load on Controller will decrease, so it can handle relatively larger network and thus the problem of scalability can be minimized.

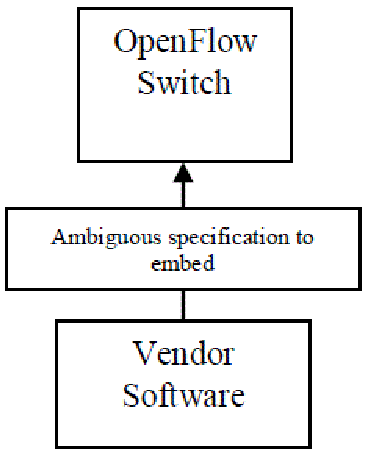

Figure 4: Non compatible for commercial use.

Controller compatibility and controller placement

Multiple Controllers may be used to control the same domain. It is important to ensure compatibility among Controllers to enable cooperation [3]. The compatibility is needed for enabling various fundamental services like inter-domain routing to enable communication between hosts in different domains (Figure 5).

If multiple Controllers compete for the same channel there may not be full utilization of Controller resources. Optimized placement of Controller is needed for maximum utilization of Controller resources.

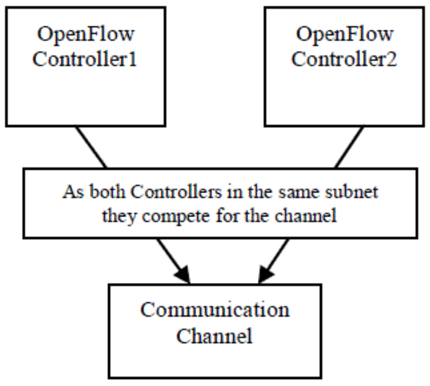
OpenFlow Controller1
OpenFlow Controller2
As both Controllers in the same subnet they compete for the channel
Communication Channel

Figure 5: Controller placement in same subnet problem.

Software language

Although SDN simplifies network management with simple interfaces to determine high level network policies, the underlying

SDN framework need to translate these policies into low-level switch configurations [5,6]. If a programmer tries to implement a new policy atomically to a particular switch, in current SDN frame work, it will cause the disruption of the whole network [3]. A suitable low level language in SDN framework is required to make this model more programmer-friendly (Figure 6).

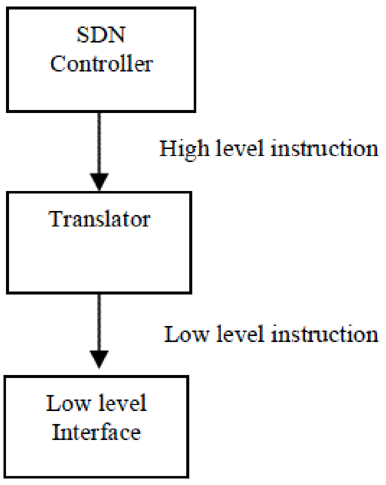

Figure 6: SDN programmers needs low level language to reach switch interface directly.

Reliability

The reliability concerns emerge in SDN in two situations – first, when a specific path/link fails and second when the Controller fails [5-7].

1. In case of path/link failure, the SDN Controller should have the ability to support fast traffic rerouting into active links.
2. In SDN, only central Controller is in charge of the whole network. If this Controller fails, the whole network may collapse. So in case of Controller failure, it is important that the Controller can enable clustering of two or more SDN Controllers in an active standby mode (Figure 7). Memory synchronization between active and stand by Controller should be maintained.

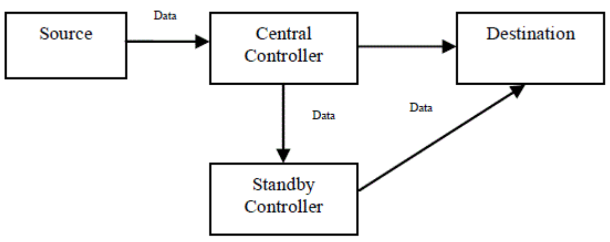

Figure 7: A reliable controller approach in SDN.

A similar solution is SiBF [5], which consists of an army of rack managers (RMs), one per rack, acting as a Controller. When the master Controller fails, flow requests are handled by another stand-by Controller (RM) until the master Controller comes back up. In case of link failure, SiBF installs new mappings (new back-up flow entries) in the switches for each active entry. The packets in the switches will be rerouted to their destinations on the alternative paths indicated by the back-up entries.

Performance under latency constraints

Performance of SDN is measured by two metrics, 1) flow setup time and 2) number of flow per second that the Controller can handle. There are two ways to setup a flow: proactive and reactive [5].

In proactive model, flow setup takes place before packet arrival at the switch, and therefore, when a packet arrives, the switch already knows how to deal with it. This mode has negligible delay and removes the limits on the number of flows per second that can be handled by the Controller [8-10].

Pro-active Flow	Re-active Flow
Flow setup before packet arrival	Flow setup after packet arrival
Negligible delay	Additional time for Controller process and update switch

Table 1: Different flow setup in SDN controller.

In reactive model, flow setup is performed when a packet arriving at the switch does not match any of the switch flow entries. Then the Controller will create flow rule to decide how to process that packet and this rule will be stored in the switch (Table 1). The reactive flow setup time is the sum of the processing time of the Controller and the time for updating the switch. Therefore flow initiation adds overhead that introduces reactive flow-setup delay.

To overcome performance limitation, the key factors that affect flow setup time should be considered, these key factors are processing speed of Controller and I/O performance of the network switch.

Flow table management

The SDN switches can always report forwarding results to the higher layer – OpenFlow Controller [3]. The results could be simple success or failure for a data forwarding operation, or some error messages or other forwarding status data. In the current OpenFlow model (Figure 8), there are no specifications on how to handle these feedback data.

If Controller stays busy with dealing redundant feedback from switch, it can slow down overall data forwarding operations during heavy data traffic in switch. It is important to perform filtering on the feedback messages from switch based on the pattern analysis in the Controller, so that it can filter redundant packets during heavy data traffic.

Robust Wireless Integration: Although OpenFlow has been well developed in wired network, there are very few studies on its performance in wireless networks [3]. Two issues are not covered in OpenFlow, which are necessary for wireless networks (Figure 9).

1. In a wireless networks the OpenFlow data panel must perform efficient channel sensing/access. However, the existing OpenFlow standard only defines data forwarding functions in the hardware.

2. Unlike wired OpenFlow model than can use cables to easily achieve control/data packet communication among nodes, wireless network uses unreliable wireless links for both control plane communications and data plane packet forwarding. The control plane demands a high-quality channel for loss-free delivery.

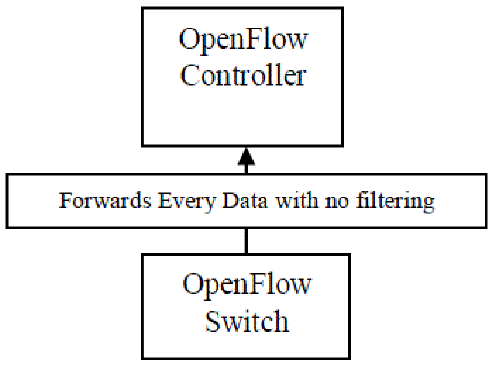

Figure 8: Non intelligent flow table.

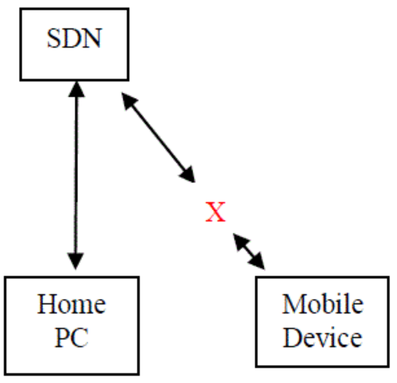

Figure 9: Limitation of SDN to wireless connection.

Summary of the Findings

I have combined recommendations from all the previously discussed issues in the Figure 10 model below:

In this reviewed model

1. SDN Controller now has a low level language instead of previous high level one, in order to make it more programmer-friendly.
2. Controller also has carrier sensing ability to help working in robust wireless environment.
3. Now there are standby Controllers to enable reliable networking with SDN.
4. There is one layer of security with the Controller. This layer can detect difference between ordinary reply from switch and malicious third party replies, thus lessening the threat of DOS attacks in Controller.
5. In switch level, clear specifications have been installed to lessen ambiguity in vendor software embedding.
6. The flow set up time is minimized by maximizing pro-active flow rate in Controller.
7. According to "DevoFlow" discussed in section II(c), the micro data flow is managed in the data plane and more massive data flow is managed in the control plane, thus reducing the traffic in Controller when SDN connections grow tremendously huge.
8. Now in Controller, only filtered data can reach from the switch. For this regard, pattern recognition ability has been included in Controller.
9. To reduce conflict, Controller of SDN is placed in an optimized way so it doesn't compete for channel in another SDN Controller's subnet.

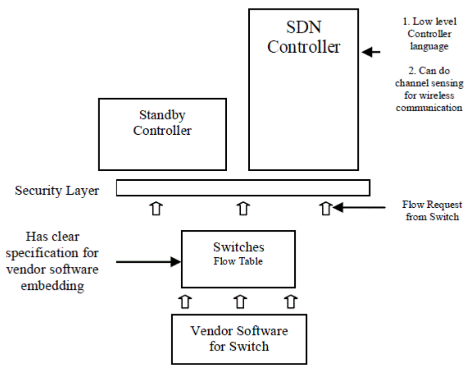
SDN Controller
1. Low level Controller language
2. Can do channel sensing for wireless communication
Standby Controller
Security Layer
Flow Request from Switch
Has clear specification for vendor software embedding
Switches Flow Table
Vendor Software for Switch

Figure 10: Reviewed model for SDN.

Conclusion

In this paper I have comprehensively surveyed the research issues for Software-defined Networking (SDN) and proposed a reviewed model of SDN. In this reviewed model, I have added recommendations on network security, compatibility, scalability, reliability, Controller placement, wireless integration, Flow Table management, performance under latency constraints, difficulties regarding language of the software in SDN etc. I believe that this reviewed model can help SDN engineers to implement moderate version of SDN so that, it can work as the proper replacement of traditional hop by hop network.

Acknowledgment

Thanks to Dr Mohammad Shohrab Hossain for technical help with building the contents in each issue. Special Thanks to Md. Iftekharul Islam Sakhib for helping with resources.

References

1. http://www.infotechlead.com/mobility/interop-2014-avaya-showcase-automated-campus-part-sdn-initiative-21223
2. http://www.networkcomputing.com/cloud-infrastructure/7-essentials-software-defined-networking/1672824201?image_number=1

3. Hu JIF, Hao Q, Bao K (2014) A survey on Software-defined Networking (SDN) and OpenFlow: from concept to implementation. IEEE 16: 4.
4. Jarraya Y, Madi T, Debbai M (2014) A survey and a layered taxonomy of Software-defined Networking. IEEE 16: 4.
5. Jammala M, Singha T, Shamia A, Asalb R, Lic Y (2014) Software-defined Networking: State of the art and research challenges. Networking and Internet Architecture.
6. Abhinav K, Kumar A, Nalini C, Venkatesan KGS (2015) QOS - guaranteed neighbour selection and distributed packet scheduling algorithm by using MANET wireless networks. IJIRCCE.
7. Sontakke PD, Dhote CA (2015) Spoofing attacks detection and localizing multiple adversaries in wireless networks. IJIRCCE 3: 5.
8. Alam M M, Hamida EB (2015) Wearable Wireless networks for internet of humans: trends and challenges. J Telecommun Syst Manage 4: 1.
9. Tamilarasi K, Bharathi R (2014) Optimum service interval and service period: MBR in wireless networks. IJIRCCE 4: 1.
10. Karthika RV, Karuppusamy MN (2014) TSROD: Time synchronization by reducing oceans delay in underwater wireless networks. IJIRCCE 2: 1.