Global Journal of Computer Science and Technology: E Network, Web & Security
Volume 17 Issue 1 Version 1.0 Year 2017
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

The Components that can Build Flexible & Efficient Software Defined Network

By Deepak Kumar & Manu Sood
Himachal Pradesh University

Abstract- SDN (Software Defined Network) is a new networking approach towards current networking industry. S.D.N has attarcted the researchers attention, because there is wide scope of innovation and research. The main concept behind the SDN networks is the separation of controller from data plane. This natural feature makes SDN adaptive of being flexible and scalable. We are mentioning some of the important components those are needed to make current SDN networks even better and efficient that can be managed easily and updated whenever needed, without any interruption of services. Also we have discussed how we can manage the data plane, control plane and how we can identify where fault has occurred.

Keywords: *SDN, hypervisor, virtualization, openflow, programmable data plane.*

GJCST-E Classification: *C.2.1, C.2.2*

Strictly as per the compliance and regulations of:

The Components that can Build Flexible & Efficient Software Defined Network

Deepak Kumar [α] & Manu Sood [σ]

Abstract- SDN (Software Defined Network) is a new networking approach towards current networking industry. S.D.N has attarcted the researchers attention, because there is wide scope of innovation and research. The main concept behind the SDN networks is the separation of controller from data plane. This natural feature makes SDN adaptive of being flexible and scalable. We are mentioning some of the important components those are needed to make current SDN networks even better and efficient that can be managed easily and updated whenever needed, without any interruption of services. Also we have discussed how we can manage the data plane, control plane and how we can identify where fault has occurred.

Keywords: *SDN, hypervisor, virtualization, openflow, programmable data plane.*

I. Introduction

In a Software-Defined Networks (SDN) the controller resides in the control plane that controls the heterogeneous forwarding devices. The main concept behind the SDN is the Data Plane and Control Plane separation, virtualization and programmatic control. Controller can change the functionality of the forwarding devices through command by changing the rules and policies. The main purpose of the SDN is to satisfy the changing needs of enterprises and users. In SDN net work administrator can change the flow of packets through centralized controller without configuring the forwarding devices (switches, routers) manually. Whenever packet came across switch (in data plane) the rules and policies installed in the firmware guide the switch where to forward the packet. The communication between the controller and data plane takes place through south bound interface usually known as Open Flow. The architecture of SDN is as shown in Figure1. There are three layers; the 1st layer is called as application layer (management plane). The 2nd layer is called as control layer (control plane) where controller resides. The controller can be any of the NOX [1], POX [2], FLOODLIGHT [3], BEACON [4] etc. The 3rd layer is known as the infrastructure layer (data plane).

Open low [5] is a protocol that actually enables the separation of control plane from data plane. To be more specific it is not the controller that controls the data plane, it the application that uses the controller to manage the switches in data plane. SDN is much flexible compared to the traditional networks the only risk is that it can be failed any time. The recent techniques are not that much sufficient to tell about how network would behave when controller will fail. There must be a network management service that can manage various network management applications to run independently, while monitoring and maintaining the performance as well as network safety. Various aspects of the network are captured by network state like which link is active and how switches are forwarding traffic. Different views can be seen through network state. Observed state that maintains the updated view of the actual state of the network, applications can read this state and changes in propose state are based on their own goals.

Also there is a need of system that can consistently update the network and dynamically schedules these updates based on the runtime difference in the update speed of various switches in Software Defined Networks (S.D.N). With the advent of S.D.N that provides the excellent opportunity to developers for developing basic abstractions for the management of network updates. Instability in networks are generally due to changes in the configuration that leads to unavailability of the network, performance problems and security issues. Sometimes intermediate configuration also behaves incorrectly during the update process even if the initial and final configurations are correct. S.D.N programs must be updated consistently as we update software, whether the reason is to migrate to new controller, bugs repairing and address performance issues.

Operators of S.D.N performs network updates by stopping the old controller and starting the new controller, this process cleanup the preinstalled entries of flow table that can creates problems including loss of packet, or increase in latency etc. There must be a mechanism that ensures to maintained the well defined behaviour of the network even if the change of configuration took place. The interaction between the today's datacenter and application running on them takes place in a complex way, making network operators to run various traffic management services to maintain the working of network. Also solution regarding traffic management are often limited because of the divide between the network and hosts. The network devices only deals with knowledge regarding layer of networks where as the hosts have the view how applications interacts with the network.

Author α σ: *Department of Computer Science Himachal Pradesh University, Summer Hill, Shimla India.*
e-mails: deepak.cs339@gmail.com, soodm_67@yahoo.com

So there must be a system that may have unified view of the both host and network so that maintenance of both takes place in easy way. Another important thing we need to be considered is packet tracing. If there is any problem regarding handling of packet, we should trace back the packet to find out the root cause. This helps in debugging the networks, testing of network performances etc. Earlier mechanism were required of modification of switches that

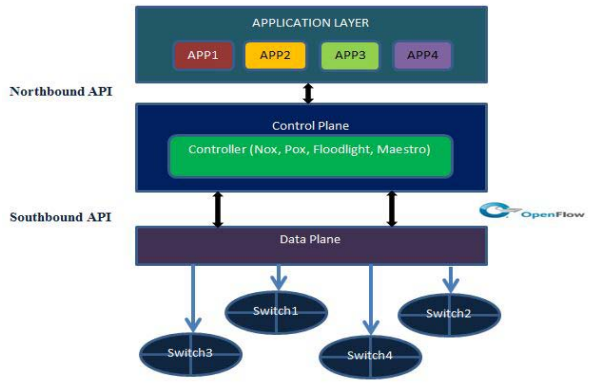

Figure 1: SDN Architecture

results in more overhead. S.D.N makes this happen to calculate the transformations that leads us to packet observations. In order to measure the flow of traffic across network paths is difficult for many management services including traffic engineering (TE) [6], diagnosing network congestion. There must be a query based language for the traffic monitoring. Also there is a need of protocol independent programming language. In next sections we are discussing few components that can make SDN much robust and efficient, like data plane performance monitoring, network performance diagnosis, hypervisor for efficient network, protocol independent language for switches, packet trace back, To find the shortest path for the forwarding of packets between switches.

II. Data Performance Monitoring

Data plane is generally local to each of the hard ware devices like switches, routers, or the card on the router, and arrival packet speed determines how to operate them. Data plane is made up of various hard ware devices of network that provides connectivity. These hardware devices are routers, Ethernet switches and firewalls. The configuration to hardware devices are provided by control plane through control interface (Open Flow) and the configuration across these devices can be updated whenever needed. In order to optimize the network configuration request is made by hardware devices to the controller (control plane). As many applications moving to the cloud day by day, so cloud operators need to diagnose performance problems consistently.

Till now Offline processing of logs is very slow and inefficient. We need a system to analyze TCP performance in terms of real time across the end-host working over hypervisor or connected to NIC [7], switch etc. It should determines whether the connection is affected at the sender's end or due to the congestion across network or problem is at the receivers end because of limited buffer capacity. With the increase of edge devices that offers adjustable processing of packet at high speed on hardware devices in data plane, that makes possible to monitor TCP performance.

P4 [8] which is a protocol independent language that help us in management of the traffic. In order to minimize the state requirements of the data-plane, there is a need of detection of all connections, after that all connections are diagnosed in order to find fault across them. In Figure 2 as shown there is a need of inbuilt diagnose or trouble-shooter in the controller so that it can consistently look for problems across the network elements and manage them as soon as possible in order provide the robust and flexible network. Red arrows showing programs written in protocol independent language i.e. P4 can be implemented in data plane through controller by programmatic control. Whereas blue arrows showing the TCP traffic across the hardware devices like switches can be monitored (TCP traffic information can be sent to control plane through data plane). Here switch1 and switch2 are the edge devices Here switch1 and switch2 are the edge devices which can be monitored through controller to captures the TCP traffic.

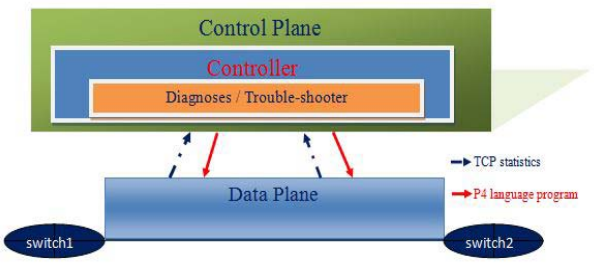

Figure 2: TCP Statistics gathering from edge devices of data plane by control plane

Diagnosis and troubleshooting will also helps to identify where the actual problem occurs: is it across sender or it is at receivers end or it is due to the network congestion. In order to make this happen, there is a need of protocol independent language like P4 through which we can write programs and be implemented through controller. can make performance of the network even more better, if we use the network elements (switches, router etc.) that supports the protocol independent languages.

III. Network Performance Diagnosis

Control plane or the controller provides the global view of the network, enabling the network administrator to update the rule, policies or protocol across the hardware devices lying in data plane at any time whenever need to be updated. S.D.N platform

provides controller the capability to intelligently control the network elements; like we can change the topology across network device if any intruder try to interrupt the flow of packet, in that case controller can intelligently sense that someone across the hardware device trying to steal information or interrupting the service e.g. in figure 3 as shown, when intruder (yellow triangle) tries to access across switch3 then switch3 report to controller through data plane. Controller than change the topology of underlying switches, as initially flow of packet takes place from switch1 to switch5 through path switch1-switch3-switch4-switch5 (blue dotted line) but due to intruder interruption across switch3, controller update the new topology across the switches, so now flow of packet between switch1 and switch5 is takes place through path switch1-switch2-switch4-switch5 (green dotted lines). This functionality of handling hardware resources through programmatic control makes S.D.N suitable choice for current networking environment.

In order to make S.D.N more efficient there is need of handling many things like, what if controller fails, in that case the whole network will suffer.

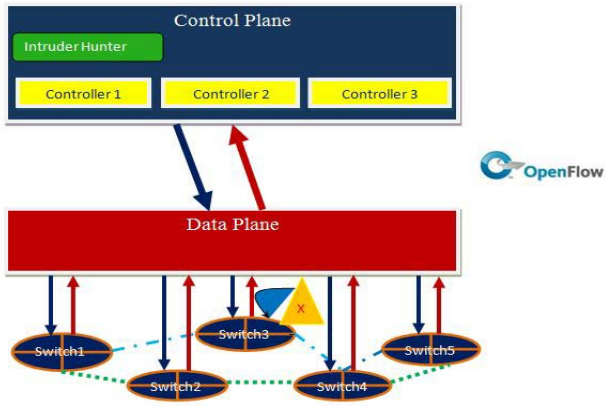

Figure 3: Intruder handling through control plane

The solution of this problem is that, there should be more than one controller in the control plane. So that if one fails other controller will control the flow of packet through programmatic control across switches. By doing so network will behave normally as there was no problem. Having more than one controllers also have other advantages, like while upgrading the controller, during that time if any fault occur in data plane then other controller will handles all the faults or provide services to the network elements, only limitation of having more than controller the cost. For an efficient network the switch should be intelligent, so that they may be able to configure the shortest path to reach the destination.

IV. Hypervisor for Efficient Network

A hypervisor commonly also known as virtually machine monitor (VMM) is a software program that is part of virtualization technology. Hypervisor [9] mainly is lates controllers (network operating system) or vari- ous business applications from the underlying hardware devices in data plane. As we have discussed in section 3.

A centralized controller in S.D.N react to network condition those are changed by upgrading the rules and policies across the hardware devices in the data plane. Every software need upgrades to fix errors, to add new features. Similarly for upgrading the controller, it need to be stopped, while during this transition, network will fail. So the idea of multiple controllers came. This idea helped to manage the network even when the one of controller fails, because other controllers are capable enough to handle any interrupt or fault along any hardware resources.

One another important point came, if controller1 installed the rule and policies across hardware devices and got failed, in that case will other controller like controller2 and controller3 will support the polices or rules installed by controller1. For this thing to happen all of controllers (as in our example: controller1,controller 2 and controller3) must linked or coordinate with each other.

Figure 4: Role of hypervisor

The thing that help the controller to coordinate with each other is called hypervisor. As hypervisor is a natural platform to support multiple operating system providing hardware devices the illusion of having only the one controller and is providing services to the individual hardware device (whether it is router, switch or access-point).

V. Protocol Independent High Level Language

The heading One of the high level language suitable for the programmed packet handler which is protocol independent is P4 [10], One of the high level language suitable for the programmed packet handler which is protocol independent is P4, P4 stands for Programming Protocol-Independent Packet Processors. P4 works in collaboration with the Open Flow protocol.

Open Flow is the protocol which is responsible for the decoupling of control plane from data plane, enabling us to write the program in P4 and implement it in data plane through programmatic control by centralized intelligent controller. The advantage of having the protocol independent language is that hardware devices are not specific to the particular network protocols. Also this provides programmers with capability to describe the packet processing functionality that is independent of the type of underlying hardware devices.

VI. Packet Trace Back

The main goal of the paper trace back is to determine how the packet has reached to its current location and also the path through which it has reached. Packet trace back [11]has the many of the advantages like; to determine the security of the network, performance monitoring and debugging of the network. DDOS attack might be first detected, and then we can trace it back and shut off the link through which it is entering. One more example is; if network administrator identifies that some flow have poor performance, through packet trace back can depict which nodes needs to be examined for congestion. Also the path followed by packet helps in debugging for errors. Figure 5 shows that inflow of packet takes place across switch D and all packet are outflow through switch A e.g. Suppose a packet-P whose first bit of the source IP address is 1, leaving switch A through port 1 and the aim is to trace back its path through the network system. Packet arriving on switch D at port id 3 is forward to switch B

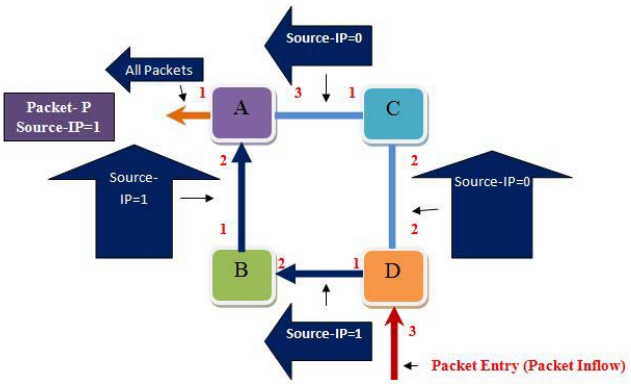

Figure 5: Packet Tracing

only if the first bit of the source IP is 1,otherwise forwarded to the switch C. As this switches B and C also forwards the packet to switch A, e.g. if switch C receives a packet with IP whose first bit is 1, then that packet would be dropped. Therefore by doing so we can determine that packet-P have not followed the path through switch C but have traversed the path through switch D-switch B- switch A.

VII. Conclusion and Future Scope

Till now we have discussed various factors that can help us to build flexible and robust network. So all of these are the approaches that we have to be considered. By considering these we can overcome and handle various faults. The switches must be intelligent enough to decide where to forward the packet in the case when controller is not responding. The main purpose of doing is that the traffic must remains in the data plane. The use of multiple controller is prime factor for making S.D.N networks much more flexible. The only portion where the S.D.N networks lacks is the security. There are various other approaches needs which can make current network even much secure.

Also if we use of the Big Data concept, that can help S.D.N to be more scalable. As this is a new trend in Networking technology so the chances of research are much more, because S.D.N in itself is very broad concept.

References Références Referencias

1. N.Gude, T.Koponen, J.Pettit, B. Pfaff, M.Casado, N. Mckeown, S. Shenker, NOX: Towards an Operating System for Networks. SIGCOMM Comput. Commun. Rev., 38*: 105-110, July 2008*.
2. POX [online] Article available at link: http://www.noxrepo.org/pox/about-pox/
3. Floodlight [online] Article available at link http://floodlight.openflowhub.org
4. D. Erickson, The Beacon Open Flow controller Proc. *In HotSDN* 2013.
5. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson,J. Rexford, S. Shenker, and J. Turner. Open flow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev., 38 (2) 69–74, 2008.
6. Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, C. Yang. Traffic engineering in software defined net working: measurement and management.
7. T. Tofigh and N. Viljoen. Dynamic analytics for programmable NIC's utilizing P4- identification and custom tagging of elastic telecoms traffic.http:// p4.org/wp-content/uploads/2016/06/P4-Poster- Net ro nome- ATT.pdf
8. M. Shahbaz, S.Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, J. Rexford. PISCES: A Programmable, Protocol-Independent Software Switch. http://pisces.cs.princeton.edu
9. X. Jin, J. Gossels, J. Rexford, D. Walker. CoVisor: A Compositional Hypervisor for software Defined Networks, Princeton University.
10. P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. talayco, A. vahdat, G. Varghese, D. Walker. P4: Programming Protocol-Independent Packet Processors.
11. H. Zhang, J. Reich, J. Rexford. Paper Traceback for Software Defined Networks, Princeton University