International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Improving Network Management with Software Defined Networking

A. Neeraja (13951D5811)[1], Dr. N. Chandra Sekhar Reddy[2], Mukund[3]

[1]Student, M. Tech CSE Department, Institute of Aeronautical Engineering, Hyderabad-500043, Andhra Pradesh, India

[2]Professor, CSE Department, Institute of Aeronautical Engineering, Hyderabad -500043, Andhra Pradesh, India

[3]Professor, CSE Department, Institute of Aeronautical Engineering, Hyderabad -500043, Andhra Pradesh, India

Abstract: ***Network Management is very challenging. The Maintenance, operating and providing a secured communication network is very complex. It requires the network operators to grapple low-level vendor specific configurations to implement the high level network policies which are complex. The rigidity of underlying infrastructure presents few possibilities of improvement or innovation, since network devices are generally closed and vertically integrated. The solution to this problem is a new paradigm in networking; software defined networking (SDN), which advocates the separation of data plane and control plane. This separation makes the network switches in the data plane as simple packet forwarding devices and leaving a logically centralized software program to control the behavior of the entire network. We focus on three problems in network management. They are enabling frequent changes to network conditions and state, providing support for network configuration in high level language and providing better visibility and control over tasks for performing network trouble shooting.***

Keywords: SDN, Network Management, Data Plane, Control Plane and open flow

1. Introduction

To operate and maintain a computer Network is a complex task. The projection of High level network policies requires configuring each individual network device separately from a heterogeneous collection of switches, routers, middle boxes using the vendor specific and low level commands. Apart from configuration complexity, Networks are dynamic and operators have no mechanisms to automatically respond to network events. It is therefore complex to implement the required policies in such dynamic environment.

The separation of control plane and data plane lays the ground to the Software defined networking paradigm. The network switches tends to be the Simple forwarding devices and the control logic is implemented in a logically centralized controller. According to the principle it is physically distributed. In SDN, the controller indicates the network behavior. The logical Centralization of Control logic in a software module which runs in a standard server in the network operating systems offers many advantages.

It is very simple and less error prone to modify the Network policies using Software than using the low-level device configurations. Secondly, A control program can automatically react to Sudden changes of the network state and can maintain the high level policies in place. And at the last, the centralization of the control logic in a controller with global Knowledge of network state simplifies the development of more Sophisticated Network functions.

SDN provides new ways to solve the Age-old problems in Networking field. It also simultaneously enables the introduction of sophisticated Network policies such as security and dependability. The Main causes of Concerns lie in the benefits of SDN Which are Network programmability and Control logic Centralization.

2. Theoretical basis and Literature review

Very little research is done in the field of language of SDN because this is a relatively new field. OpenFlow is the Application Programming Interface used for the controller to talk to the switches below where as the SDN makes it possible to program the network, it does not make it easy. OpenFlow controllers in todays generation offer low-level APIs that mimic the underlying switch hardware. As such a new platform must be created to provide programmers to program with ease and not have to deal with the lower level switches.

There are four problems with the current OpenFlow program. First is that the programs do not compose because of the interaction between concurrent modules. There are rules for each program and when running these programs concurrently the rules will overlap, causing programs to break. Second problem is the low level programming interface that impedes programmers to abstract and create large and complex program for the network. Third problem is the Two-tiered system architecture that forces programmers to specify communication patterns between the controller and each individual switches to avoid tricky concurrency issues. The last challenge is network race conditions that arise directly from the two-tiered system.

3. Features of Software defined Networking

Software Defined Networking (SDN) provides very interesting features which will revolutionize the future networks like centralize control mechanism, cost efficiency, innovation, programmability, scalability, security, virtualization, cloud support, automation, reliability and efficient environment to support Big-Data. One of the main objective of SDN is to provide innovation over the internet or the network, this ensure very less vendor dependence.

Programmability by operators is feature of Software Defined Networking which helps to put innovation into practice. Security and reliability are the major issues in today's networks, SDN enhance these two aspects to the maximum. Simple operation and cost efficiency are key factors with openness to design and invent in SDN. Yet this openness with respect to APIs not yet fully explored but in study this aspect has been studied. SDN makes it easy to share resources in an efficient manner with the feature of network intelligence.

SDN is helpful in service aware networking too. With SDN, Network Management has improved vastly which also give it cutting edge on traditional networking architecture and important determinant in future works. Networks based on Software Defined Networking are being implemented both on testbed and production networks. Fault tolerance property is a key for the production networks and a most desirable & a must for SDN networks. It is noted that with respect to fault tolerance in SDN there are not much researches. In the paper CORONET, a SDN fault-tolerant system is discussed. A fault tolerance SDN architecture is stated which quickly recover from the occurred faults and can work on highly scalable networks. The architecture describes the recovery from multiple links failures. SDN based networks are more and more been deployed both on testbed and production networking environments.

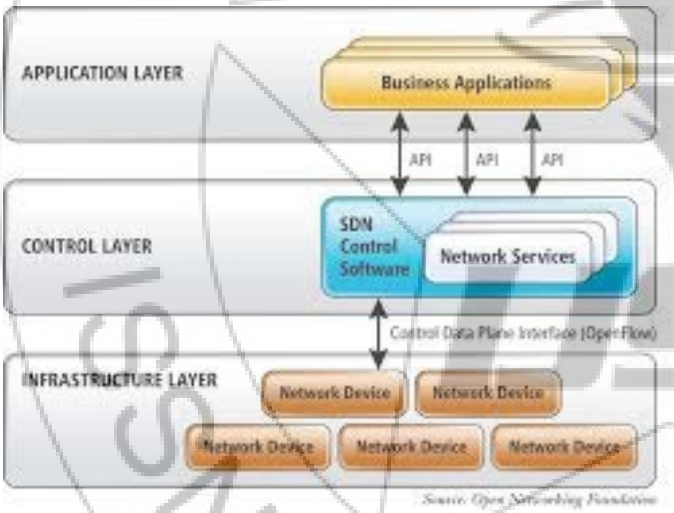

Figure 1: Architecture of SDN

4. Secure and Dependable Control Platform

Here, we present the general design of the secure and dependable SDN control platform we propose.

4.1 Replication

One of the most important techniques to improve the dependability of the system is replication. As can be seen in below figure our controller is replicated, with three instances in the example. Applications should be replicated as well. Besides replicated instances of the controller, in the figure we can observe application B also replicated in all controller instances. This mixed approach ensures tolerance of both hardware and software faults, accidental or malicious. Replication makes it possible to mask failures and to isolate malicious or faulty applications and/or controllers. Moreover, in case of a network partition, application B, with the proper consistency algorithms, will still be able to program all network switches, contrary to application A.

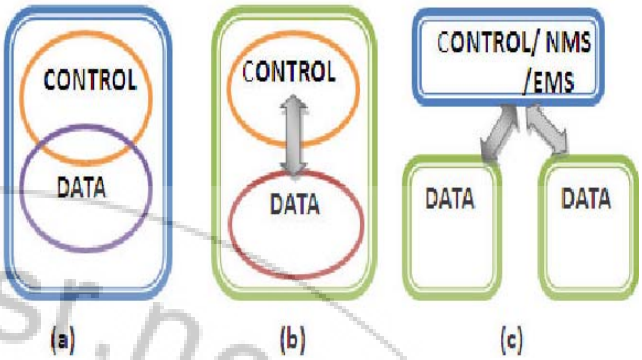

Figure 2: Secure & Dependable SDN

4.2 Diversity

Another relevant technique to improve the robustness of secure and dependable systems is diversity .Replication with diverse controllers is a good starting case. The basic principle behind this mechanism is to avoid common-mode faults (e.g., software bugs or vulnerabilities).For example, it is known that off-the-shelf operating systems, from different families, have few intersecting vulnerabilities which means that OS diversity constrains the overall effect of attacks on common vulnerabilities. In SDNs the same management application could run on different controllers. This can be simplified by defining a common abstraction for applications (a northbound API).

4.3 Self-healing mechanisms

Under persistent adversary circumstances, proactive and reactive recovery can bring the system back to a healthy state, replacing compromised components, and keep it working virtually forever. When re-placing components, it is important that the replacement be done with new and diverse versions of the components, whenever possible. In other words, we should explore diversity in the recovery process, strengthening the defense against attacks targeting specific vulnerabilities in a system.

4.4 Dynamic device association

If a switch is associated with a single controller, its control plane does not tolerate faults. Once the controller fails, the control operation of the switch fails and the switch will need to associate with another controller. For this reason, a switch should be able to dynamically associate with several controllers in a secure way A switch associated with different controllers would be able to automatically tolerate faults. Other advantages include increasing control plane throughput several controllers could be used for load balancing and reducing control delay by choosing the quickest-responding controller.

5. What Is OpenFlow?

OpenFlow is an open, standards-based communications protocol and an example of device-based SDN. OpenFlow

provides access to the forwarding plane of a network switch or router over the network, facilitating more sophisticated traffic management, especially for virtualized and cloud environments. The OpenFlow protocol is standardized and managed by the Open Networking Foundation (ONF), whose mission also includes the promotion of SDN technologies as a whole.

In a classical router or switch, the data plane and the control plane reside on the device. OpenFlow enables part of control plane operations to run on external servers called controllers. In practice, an OpenFlow API is generally a feature added to commercial network devices, whose hardware architecture and features remain crucial to network performance. The standard control plane of the device remains in place and performs traditional routing or switching. Today, most OpenFlow-enabled devices can also support both OpenFlow traffic and non-OpenFlow traffic, with mechanisms for determining to which pipeline each traffic flow should be routed.The real benefit of OpenFlow lies in the applications that it can enable. New forms of traditional control plane applications such as security or specialized QoS functions—and even entirely new applications—may be written to these controllers, as shown in Figure 1 below. This will enable cloud and hosting providers, in particular, to develop and market more truly differentiated services to their clients. Traditional enterprises can also benefit from this type of third-party network application development, for example, in developing capabilities that help meet the operational or regulatory requirements of their industry.

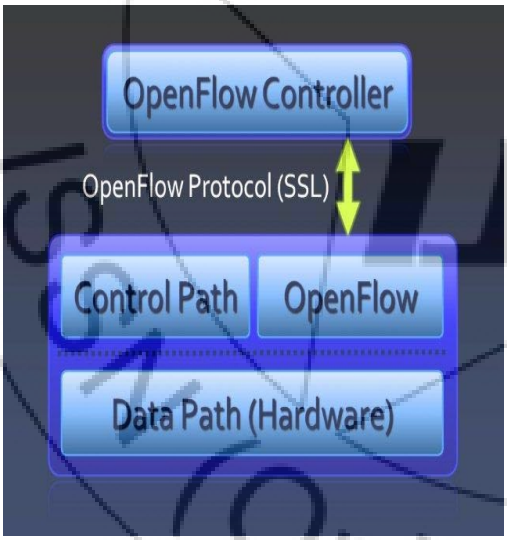

Figure 3: Controller Path

5.1 Use Cases for Control Plane Abstraction:

SDN will enable a wide variety of use cases as the technologies mature. In the near term, these are some of the most commonly envisioned scenarios:

- Service assurance through flow optimization in the Wide Area Network (WAN). Public cloud providers may wish to ensure their SLAs by maintaining visibility and control of traffic all the way to the client's network edge. This can be achieved by
- deploying OpenFlow-enabled devices both at the cloud provider edge and client ingress, with both devices communicating to the cloud provider OpenFlow controllers. OpenFlow can also help provide granular control of inter-data center traffic, including backup or disaster recovery operations.
- Service differentiation through rapid customization. As illustrated in Figure 1, the ability to develop new features quickly for highly specialized use cases is appealing to many, particularly in the cloud and hosting space, as it can provide opportunities for timely service differentiation and incremental monetization of the network. Such use cases might take the form of new security offerings, service levels, or bandwidth on demand.
- Service velocity through highly scalable and easily orchestrated network virtualization. By defining within the controller a set of policies that can be applied to any number of flows at need, the operator is able to truly divorce the service delivered
- to the client from the physical locations of the infrastructure supporting it.

5.2 Cloud SDN

Dynamic nature of cloud services requires server virtualization to be administered in real time utilizing network virtualization. Software Defined Networking is the new paradigm of networking, which uses a centralized controller to control the flow of packets in the data plane. This new approach makes network management easier and has ability to save costs for the organization. There has been significant advancement in cloud computing technologies, which has led to the development of cloud management tools like OpenStack (An Open source Infrastructure as a Service (IaaS) cloud computing project).

6. Future Work

The future of networking will rely more and more on software, which will accelerate the pace of innovation for networks as it has in the computing and storage domains. SDN promises to transform today's static networks into flexible, programmable platforms with the intelligence to allocate resources dynamically, the scale to support enormous data centers and the virtualization needed to support dynamic, highly automated, and secure cloud environments. With its many advantages and astonishing industry momentum, SDN is on the way to becoming the new norm for networks.

7. Conclusion

SDN is a compelling development for the public sector. It helps to simplify operations by automating and centralizing network management tasks. It makes the network more responsive to dynamic business and institutional needs by coupling applications with network control. Finally, SDN gives IT teams more agility, because they can quickly customize network behavior for emergent business needs. The increasing velocity of application development will continue to drive IT organizations to deploy technologies that allow them to scale and respond to rapidly changing demands.

References

[1] T. Koponen et al. \Onix: a distributed control platform for large-scale production networks". In:*OSDI*. 2010.
[2] N. Gude et al. \NOX: towards an operating system for networks". In: *Comp. Comm. Rev.* (2008).
[3] M. Caesar et al. \Design and implementation of a routing control platform". In: *NSDI*. 2005.
[4] M. Casado et al. \Rethinking Enterprise Network Control". In: *ACM Trans. on Networking* 17.4 (2009).
[5] P. Porras et al. \A security enforcement kernel for OpenFlow networks". In: *HotSDN*. ACM, 2012.
[6] http://readwrite.com/2013/04/23/software-Definedd-networking-dn#awesm=~omPg0fn3rysfHX
[7] Vijay K. Gurbani, M Scharf, T.V. Lakshman and V. Hilt. Bell Laboratories, Alcatel-Lucent"Abstracting network state in Software Defined Networks (SDN) for rendezvous services"." Communications (ICC), 2012 IEEE International Conference on Software Defined Networks. (2012).
[8] http://searchcloudprovider.techtarget.com/tip/Three-models-of-SDN-explained
[9] Fl´avio de Oliveira Silva, Jo˜ao Henrique de Souza Pereira, Pedro Frosi Rosa†, and Sergio Takeo Kofuji. "Enabling Future Internet Architecture Research and Experimentation by Using Software Defined Networking". EuropeanWorkshop on Software Defined Networking. 2012.
[10] Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, “A clean slate 4D approach to network control and management,” SIGCOMM Comput. Commun. Rev., vol. 35, no. 5, p. 41–54, Oct. 2005. [Online]. Available: http://doi.acm.org/10.1145/ 1096536.1096541 .
[11] N. McKeown, T. Anderson, H. Balakrishnan, G.Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, p. 69–74, Mar. 2008, ACM ID: 1355746.
[12] Dimitris Syrivelis, George Parisis, Dirk Trossen, Paris Flegkas, Vasilis Sourlas, Thanasis Korakis and Leandros Tassiulas. "Pursuing a Software Defined Information-Centric Network". European Workshop on Software Defined Networking, 2012.
[13] Myung-Ki Shin, Ki-Hyuk Nam; Hyoung-Jun Kim. "Software-Defined networking (SDN): A reference architecture and open APIs". ICTC. 2012.
[14] IBM. "IBM Systems and Technology Thought Leadership White Paper". 2012
[15] Open Networking Foundation. "Software-Defined Networking: The New Norm for Networks". ONF White Paper. April 13, 2012.
[16] OpenFlow switch specification Version 1.3. Open Networking Foundation. Avaible at:Http://www.opennetworking.org/. 2012.
[17] Open Networking Foundation. “OpenFlow /Software Defined-networking(SDN)”. http://www.opennetworking.org/.