Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 7, No. 4, November 2016), Pages: 27-39
www.jacr.iausari.ac.ir

Abstracted, Uniform and Centralized Global View of the Network State and Configuration

Mahdi Arghavani[1,2], Mahmood Ahmadi[3]✉

1) Department of Information Technology, Collage of Engineering, Kermanshah Branch, Islamic Azad University, Kermanshah, Iran.

2) Department of Information Technology, Collage of Engineering, Kermanshah Science and Research Branch, Islamic Azad University, Kermanshah, Iran.

3) Department of Computer Engineering, Razi University, Kermanshah, Iran

mahdiarghavani@iauksh.ac.ir; m.ahmadi@razi.ac.ir

Received: 2016/04/09; Accepted: 2016/06/13

Abstract

The proliferation of Information and Communications Technology (ICT) along with the incremental growth in the use of Cloud services, enterprise network and data center technologies in recent years, have caused unprecedented expansion of computer networks. However, the existing network architecture was not well designed to meet the requirements of today's needs. One of the most important problems in the existing large networks is the lack of abstracted, uniform and centralized global view of the network state and configuration. The efforts to achieve this goal lead us to construct a novel SDN based system to give network visibility to operators and management applications. In this paper, we have a new look to the importance of the logically centralized control plane, abstraction and network's global view. To prepare this network-wide visibility for the use of applications in management plane, we have designed and implemented a new northbound interface and a key-enable control module called CTRLmod based on SDN in POX platform. To show how the system works, GVpresent is developed to graphically display real-time network topology, state and configuration.

Keywords: *SDN, Network-Wide View, Network Abstraction, Centralized Control*

1. Introduction

Computer networks have had a profound impact on human life and societies [1]. It has been growing explosively [2] but there are some problems that are inherited from its traditional architecture [3], [4], [5]. In the early stage of computer networks, the requirements were simple and limited to things like email or file transfer. At that time, experts who had the honor to design this kind of fantastic relationship never had considered the complexities and requirements of today's large networks such as the need for having network-wide visibility. With the widespread use of this technology and the emergence of new services, requirements continuously increase and the need for changing in the architecture was unavoidable. But because of issues such as significant capital investment and compatibility issues, the redesign and redeployment of new network architecture progress slowly.

In traditional architecture control mechanisms are integrated with forwarding elements and are distributed across networks. So there is not a centralized control platform armed with a global view to supervise the entire network. In this architecture,

to express the desired high-level network policies, operators have to manually configure devices individually with low-level and vendor specific commands.

Network conditions continually change over time and operators under the pool of various services and standards do not have any comprehensive view of the network and they are obliged to rely on rudimentary tools (e.g., ping, traceroute) besides with their own vague intuition [5]. The lack of proper abstraction, network-wide view and centralized control platform, make the network management more complicated and it will be challenging, error-prone, repetitive and tedious tasks [4].

It is clear that computer networks as well as any other systems have their best functionality and usability when they well controlled and managed. But a large network cannot be put together and managed by human effort alone. Shenker [6] has pointed out that era of ad-hoc control mechanisms is over and we need a more systematic design. That is why the researchers are looking to make it more programmable because the programmability facilitates automation of configuration, policy enforcement, and management [7]. As we will see later, abstracted centralized global view is an important part of this systematic design and the operators by the use of it can be aware of network circumstance to create innovative solutions and program the network like a robot.

Software-Defined Networks (SDN) is a new paradigm in network management that make network programmable through decoupling control plane from data plane and centralized it in a software abstraction (Figure 1). With SDN, the flexible, simple and vendor-agnostic switches offer full line-rate forwarding performance (Gbps) for any scale of deployment. The forwarding decisions are flow-based and dictate to data plane from control plane via well-defined Application Programming Interfaces (APIs) such as OpenFlow [8]. In this architecture, control logic (e.g., routing algorithms like BGP) is implemented in a centralized controller, which is a software stack running on a commodity hardware and manages flow control to enable intelligent networking. The control plane also communicates and offers services to the management plane via open northbound APIs such as one which has been designed in this paper.

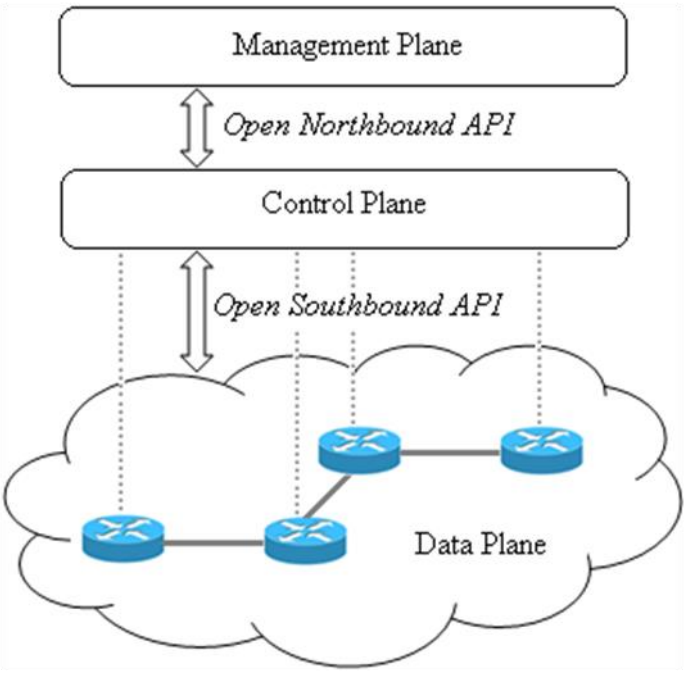

Figure 1. Simplified view of an SDN architecture [3].

In this paper, we will design and implement a novel approach to present network-wide view. Our contributions are summarized as follows.

- Proposal of a key-enabled control module called CTRLmod that provides a network-wide view.
- Proposal of a novel northbound interface to bring more abstraction and facility into network management.
- Proposal of an SDN-based application called GVpresent to display network state, topology and configuration in a graphical real time manner.

In this paper, the importance and benefits of network-wide view and its building blocks is discussed in section 2. Design and implementation of our contribution are described in section 3. Related works are expressed in section 4. Conclusion and future works are presented in section 5.

2. Network-Wide View

To have a more effective and efficient network management, we require a network-level understanding of the entire state and configuration. Armed with a single unified comprehensive view of these conditions which is frequently updated, allows the controlling system to have better perception, analysis and ultimately make optimal decisions [9].

The global view maintains a timely, accurate and coherent snapshot of the state of each network component (e.g., port up/down status, link weight, ACLs, BGP import policy and routing information). It helps operators to have network verification, validation and assurance. This outlook improves innovative solution, reduces cost and increases system performance [4], [9].

There is no any abstracted network-wide view in traditional networks and each new function (e.g., ACL or accounting) have to rely on its own incomplete view [19]. This has been a cause of emerging unrestrained and disproportionate set of various protocols that have been developed to conquer many architectural defects. In such environments, decisions are made based on limited and imperfect local information of the network global state (e.g., RIP or even OSPF).

In SDN, control plane is run on one or more servers and handles state distribution via northbound and southbound APIs in a bidirectional communication. The purpose of this handling is to collect information from switches, giving them proper control states, coordinate servers about state and services to management plane's applications. So with SDN, network-wide view and its basic primitives are implemented once in a centralized abstraction point and reused across many applications.

This view can be implemented in a data structure beside other control plane modules, or in a database on one of the control plane servers. The use of this structure, make the logic of network applications creation very easy. For example a traffic engineering application can refer to the traffic matrix and observes volume of traffic from each ingress point to each egress point, or a routing application can apply dijkstra's algorithm over the structure representing the topology and find shortest path between two points, or another application may notify operators to change an interface card, because of increasing CRC errors. The amount and attribute of detail that the network-wide view will provide rely on what the control program wants to do and what is the system's limitation. Security issue, QoS or even routing algorithm would all require different information and different amounts of detail on the underlying network [9], [18].

Two major building blocks of network-wide view are Centralized Control and Abstraction:

2.1 Centralized Control

In aircrafts, phone systems, electric power grids and any other distributed systems which precise control is necessary, a cabin or control room is required. In these place operators, observe the situation, analyze the data and then steer remote equipment. This mechanism increases reliability and provides the possibility of tuning and troubleshooting. By this centralized control platform, they can react more comprehensive to dynamic network conditions and can lead it in accordance with the current status of operation.

Before SDN, control and data plane separation had introduced in NCP [11], RCP [12], 4D [13] and Tempest [14]. For example in the 4D one of design principles is to have network-level objectives which means targets should be explained in the form of network-level terms based on entire network state. To this end, 4D separates the decision logic from the underlying protocols [13], [15].

Compared with traditional network, the separation benefits include:

a) Each part can be developed independently.
b) It can use more factors into solutions (e.g., including current bandwidth loads). When there is centralized control, if we want to change policy, it's enough to change functions and parameters. In other words it's not necessary to change the mechanism.
c) Capacity of memory and processor available on the controller server is much stronger than that of network devices. So control platform is able to analyze faster with higher quality on their decisions. In addition, control program designers do not worry about device overloading and can use resources easily.
d) Access to network-wide view and feasibility of powerful logical analysis. If the control plane does not separate from forwarding plane, then integrated network devices will not be aware of global state and can not do the best. For example in IS-IS, the network is divided into areas and in OSPF, it is divided into some domains in which each router can access only to information of its domain or area.
e) The control plane can access to secondary memory, but this feature is very expensive for network devices. In traditional approaches, storage and processing limitations can create a more unpredictable environment.
f) There is no real-time awareness of network state (e.g., traffic load, device configuration) in the traditional networks. Even some useful information such as maximum bandwidth, security policies and statistical information is not typically shared. So there is a need to separate and centralize control plane to collect real-time data to make proper decisions required at any time [16].

It should be noted that aforementioned separation has some challenges such as scalability, reliability and security issues.

2.2 Abstraction

In computer science abstraction is a technique for managing complexity of computer systems. Abstraction is freedom from hardware dependency, implementation details and cumbersome limitations. By the abstraction only the functionality is shown.

Distributed large scale and complex computer systems are made by the use of abstraction because it simplifies the interfaces between various components [7]. For example, programming languages in the early days did not have any abstraction and machine language which is a complex and confusing method, was the only way to interact with the hardware.

Because of the abstractions are key to extracting simplicity [17], experts began to create virtual memory, library functions, OOP, garbage collection and so on. Based on these concepts and by the provided flexibility, creative programmers can implement efficient software.

In traditional networks the lack of abstraction, in which can explain requests in human term, or an operating system that undertake interaction with devices, causes to be network management very difficult. Fortunately, the idea of separating control plane from data plane and centralize it in a software layer, provides a nice abstraction. In this way OpenFlow is the best. With a flow-based abstraction, this general forwarding model receives the control logic from a software level and injects rules in the simple switches which are distributed in the network.

SDN in a comprehensive point of view, have different levels of abstraction in its three layer architecture. It introduces Northbound and Southbound interfaces and provides communication between layers and exposes the appropriate levels of detail for complex network functions. The existence of these interfaces simplifying the management of large-scale complex networks.

By implementation of abstraction in APIs, SDN makes it feasible to program network and facilitates automation of provisioning, configuration, and management. In this condition the applications which are running in management plane gather information and express desired behavior without being involved in details (e.g., how much traffic is there between Node1 and Node2 or do not communicate Node1 with Node2 for two hours).

Requests arrive in the control plane via northbound interface and cause to call some procedures with various parameters. If it is necessary to refer to the data plane, these procedures send some messages to the switches through southbound interface and make changes in their flowtables or collect some information about the network state.

As explained before, high-level requests of management plane arrive to the control plane. Each request is broken into several instructions, then if there is a need they will be more fine-grained and disseminate to data plane. This flow is bidirectional in which the fine-grained information in data plane arrives to the control plane and then if needed, some processes will be done on it. Then the processed data will be sent to the higher level in the coarse-grained form. Compared to the traditional approach, desired behaviors in SDN architecture express faster in higher level (e.g., declarative language) and are easier to understand and debug.

Interfaces provide abstraction and together bring modularity, portability, assurance and efficiency. As Barbara Liskov said, Modularity based on abstraction is the way things get done [17]! With having modularity the network programming would be systematic and algorithmic. Not only by automated tools which are made in this way, network management is not more time-consuming, dull, error-prone and costly but also, by preventing inconsistencies and misconfigurations, the network would be more stable and uptime.

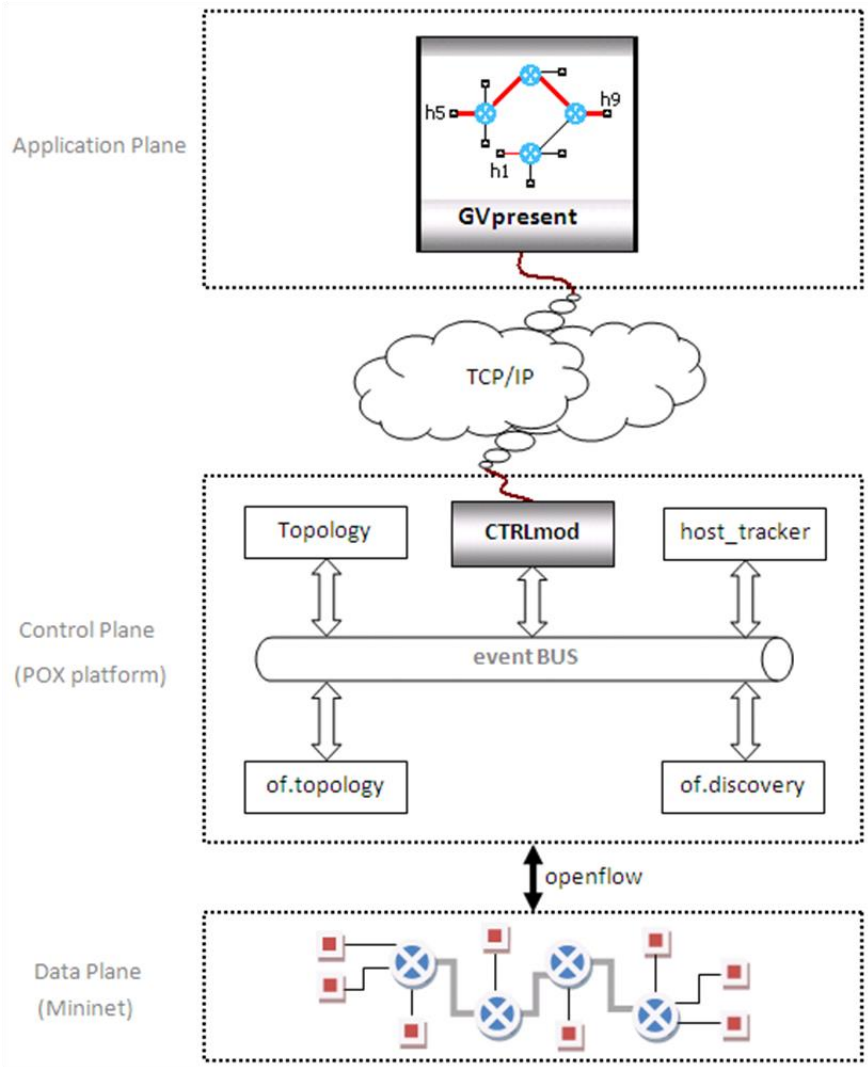

Figure 2. Components and their relationships.

3. Design and Implementation

As discussed in pervious sections, SDN proposes the network-wide view by separating control from data plane and centralized it in a software abstraction level. In this contribution, we design and implement CTRLmod, GVpresent and a new northbound interface between them based on SDN architecture. The first in control plane, take routing decisions and supply network-wide view and the next in the management plane, display network state and configuration (Figure 2).

We use POX [21], a rapid prototype and development platform as the basic control plane and Mininet [22], a network emulation environment as the data plane for our work. Both of them are open source and are written in Python. They are geared to support research and education as a learning and prototyping tools in the topic of software defined networks.

POX provides an abstracted and event-oriented interface to OpenFlow protocol. It is responsible to discover SDN switches, enforce policies to and collect statistic from them. A module developed on top of that can be a producer-consumer of events. Each of these modules can produce events and register them with core in a regular format to notify those who subscribed for. They can also subscribe to events registered by other modules and be aware of their occurrence [23].

Mininet uses Linux lightweight virtualization to rapidly prototyping large programmable networks on the limited resources of a single PC. Any Mininet host or switch behaves just like a real one because they all are realistic virtual machines which run on a single Linux kernel. So Mininet networks can run real code, applications and protocol stacks without modification [24].

In control layer, there are many modules with various tasks but only those which are directly collaborated with CTRLmod are shown in figure 2. For example of.discovery discovers the connectivity between OpenFlow switches by sending out LLDP packets [25]. This module also raises LinkEvent to notify changes of links (up/down) to modules which have subscribed for it. The host_tracker module tracks hosts and raises HostEvent to notify subscribed module when hosts join, leave, or move within the network. It is important to mention that the host_tracker is not able to detect an existing host before its first interaction with the network.

<xml><switches><item dpid="1" capabilities="135" /><ports name="s1-eth1" port_no="1" hwAddr="f6:27:2b:e7:88:17"></ports><ports name="s1-eth2" port_no="2" hwAddr="7a:e2:67:a7:97:17" connecteddpid="2"></ports><ports name="dp0" port_no="65534" hwAddr="00:23:20:92:10:98"></ports><item dpid="2" capabilities="135" /><ports name="s2-eth1" port_no="1" hwAddr="2a:aa:04:e5:9a:49"></ports><ports name="s2-eth2" port_no="2" hwAddr="ae:e9:9d:ed:99:74" connecteddpid="1"></ports><ports name="dp1" port_no="65534" hwAddr="00:23:20:f8:64:c6"></ports></switches><links><item fromdpid="2" frommac="00-00-00-00-00-02" fromport="2" todpid="1" tomac="00-00-00-00-00-01" toport="2" /><item fromdpid="1" frommac="00-00-00-00-00-01" fromport="2" todpid="2" tomac="00-00-00-00-00-02" toport="2"/></links><hosts></hosts></xml>

Figure 3. The CTRLmod's sample response, when it gets the "Topo" message.

3.1 CTRLmod module

The CTRLmod is written in python and plays an important and focal role in the system. It uses multithreading technique and has two significant tasks. One is routing packets by "layer 2 learning switch" technique and installing rules in switches, and the other one is providing network-wide view for applications such as GVpresent.

Currently there is no any accepted standard for northbound interactions. This lack of northbound standards is considered a challenging deficiency in SDN and some research groups are developing proposals to standardize it [16]. We have developed an innovative way to do that by exchanging XML message between CTRLmod and SDN applications (here GVpresent).

There may be more than one SDN application running on top of a single controller. For this purpose, we have used client-server architecture to connect management plane with CTRLmod module. In this architecture which is based on the request-response

pattern, each application can request information about network in a high-level abstract term. For example when GVpresent requests topology information from control plane by sending “Topo” message, the CTRLmod responses a proper message in XML format and gives network topology information back to it (Figure 3). As is shown, relatively digested and human-readable information are inserted in XML tags with a certain predefined format.

But about presenting the fine-grained information which maintain in switches, CTRLmod needs to communicate with data plane. In this case CTRLmod sends a request via OpenFlow to underlay and returns its answer to the client application usually in a more coarse-grained abstract form.

3.2 GVpresent Application

As it mentioned we have implemented Gvpresent in the management plane. This SDN application communicates with CTRLmod in control plane through TCP/IP and sends requests to the module to learn about network’s state and configuration (Figure 2). Gvpresent shows a graphical and real-time global view over the network in data plane. This view includes information on configuration of switches and ports, network’s topology, links status, flow entries and so on.

After running the system, if user sends a request to view network topology, a graph same as figure 4 will be appeared. In this example there are four switches and identified eight hosts. In the displayed image black lines indicate links with no traffic, thin red lines indicate links with light traffic and thick red lines indicate links with rather heavy traffic. Note that each link has two counters in switches which connected to them, so the load of links is calculated by this fine-grain information. In the figure 4, the thick red line indicates rather heavy traffic between H5 and H9. This figure is updated every few seconds according to what the user specifies. If there would be a change in network topology or links status, the GVpresent considers it in the displayed image. Figure 5 shows a larger network topology.

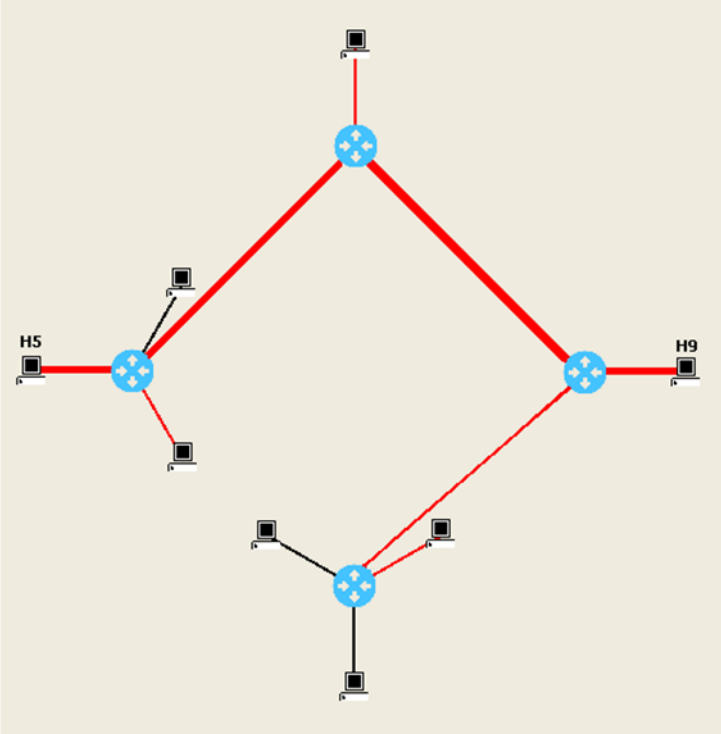

Figure 4. Network topology and its real time traffic state.

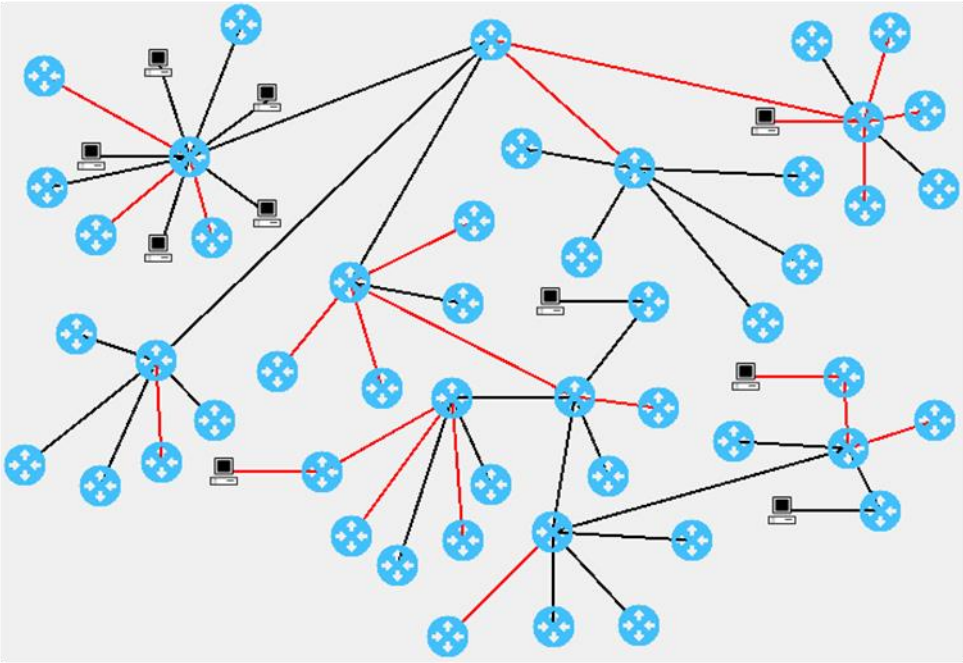

Figure 5. A larger topology.

In terms of scalability and consistency, the information of this view must be updated slowly. So some fine-grained information may stay in lower layer (e.g., packet state in switches) and only to be fetching when needed.

If we click on any of the switches in the displayed topology, we're taken into a page wherein information on that switch is available. As we shown in figure 4, switch number 1 (the left one) contains three hosts and is adjacent to only one other switch. MAC address and its capabilities are also displayed (Figure 6).

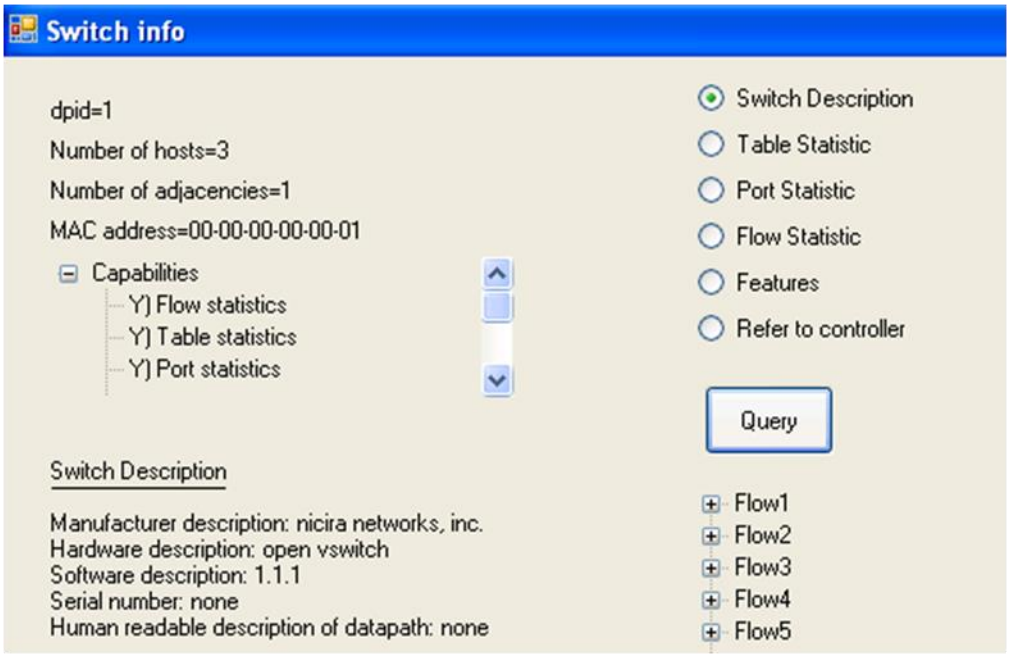

Figure 6. Switch state and configuration.

There are some selection buttons on the right side of the panel to run queries about the state information and configuration of the switch. For example figure 7 shows flowtable entries in hierarchical order.

Figure 7. A flowtable entry.

4. Related Work

Ours is not the first attempt to develop a network-wide view. However, it is easy to understand, interactive and requires no complex bindings. The 4D project was the first one which used the idea of giving control mechanisms a global view of the network [18]. Later, SDN considered this view on its architecture. One of the most prominent is Onix [10]. In this distributed control platform, global view is include physical and logical entities organized in a meaningful structure called Network Information Base (NIB). Applications can query network state through NIB and write its computation back to change the network behavior. In yanc [20] which is a controller platform for SDN, network configuration and state are exposed as a file system. In this model information about hosts, switches, flows and so on stored in a form of meaningful hierarchical directory and file structure. Consequently users and management applications have to interact with this shared information through standard file I/O without high-level abstraction. Pantuza et al in [23] use graph approach in SDN to model network topological information and represent it by an offline visualizer in a non-interactive manner.

5. Conclusion

In this paper, after investigating the need of new network architecture, we described the benefits of having centralized control platform, abstraction and network-wide view. Then a novel control module and northbound interface in POX platform were designed and implemented to provide this view for applications of management plane. Finally, an innovative SDN application named GVpresent was created. This tool shows the system operation and can help network management team to have information dominance over the entire network.

Our deployments and operational experiences on many emulated networks in Mininet demonstrate the feasibility of proposed system but more evaluation on performance, reliability and scalability is required. As GVpresent is based on the software defined networking paradigm, it also suffers from the inherent delay caused by the communication of the separated planes. So the real time representation of the network state is limited to the updates that occurred every few seconds.

In the future, we try to add notification and logging mechanisms to the system in which the operators (or management applications) would be able to monitor and compare the current network state beside the old one. In current design, the access to CTRLmod services is without any limitation while it is necessary to develop an access right mechanism. It seems it is possible to design more feasible network debugging features and so that operators can apply their decisions to the network.

Video Availability

A video of the system in action is available at https://youtu.be/wg_rhq82ZKE.

References

[1] Kavanaugh, “The impact of computer networking on community: A social network analysis approach,” Telecommunications Policy Research Conference, 1999.

[2] Cisco, I. "Cisco visual networking index: Forecast and methodology, 2011–2016." CISCO White paper (2012): 2011-2016.

[3] D. Kreutz, F.M. Ramos, P. EstevesVerissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig, "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, 103(1), 14-76.

[4] ONF, "Software-defined networking: the new norm for networks," Mar. 13, 2012 [Online]. Available: https://www.opennetworking.org/images/stories/downloads/whitepapers/wp-sdn-newnorm.pdf.

[5] P. Kazemian, "Header Space Analysis," Doctoral dissertation, Stanford University, (2013).

[6] S. Shenker, "Network Management and Software-Defined Networking (SDN)," (EE122 Fall 2012). http://inst.eecs.berkeley.edu/~ee122

[7] S. Baucke, R. Ben Ali, J.Kempf, R. Mishra, F. Ferioli, A. Carossino, "Cloud Atlas: A Software Defined Networking Abstraction for Cloud to WAN Virtual Networking," In Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on (pp. 895-902). IEEE.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, J. Turner, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, 38(2), 69-74, (2008).

[9] P.A. Morreale, J.M. Anderson, "Software Defined Networking: Design and Deployment," CRC Press, (2014).

[10] M. Casado, N. Foster, A. Guha, "Abstractions for software-defined networks", Communications of the ACM, 57(10), 86-95, (2014).

[11] D. Sheinbein, R.P. Weber, "Stored program controlled network: 800 service using spc network capability," Bell System Technical Journal, 61(7), 1737-1744 ,(1982).

[12] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, J. van der Merwe, "Design and implementation of a routing control platform," In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2 (pp. 15-28). USENIX Association, (2005).

[13] Greenberg, G. Hjalmtysson, D.A. Maltz, A. Myers, J. Rexford, G. Xie, H. Zhang, "A clean slate 4D approach to network control and management", ACM SIGCOMM Computer Communication Review, 35(5), 41-54, (2005).

[14] J.E. Van der Merwe, S. Rooney, I. Leslie, S. Crosby, "The Tempest-a practical framework for network programmability", Network, IEEE, 12(3), 20-2, (1998).

[15] N. Feamster, J. Rexford, E. Zegura, "The road to SDN," Queue, 11(12), 20, (2013).

[16] P. Goransson, C. Black, "Software Defined Networks: A Comprehensive Approach", Elsevier, (2014).

[17] S. Shenker, M. Casado, T. Koponen, N. McKeown, "The future of networking, and the past of protocols," Open Networking Summit, 20, (2011).

[18] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, S. Shenker, "NOX: towards an operating system for networks," ACM SIGCOMM Computer Communication Review, 38(3), 105-110, (2008).

[19] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, S.Shenker, "Onix: A Distributed Control Platform for Large-scale Production Networks," In OSDI (Vol. 10, pp. 1-6), (2010, October).

[20] M. Monaco, O. Michel, E. Keller, "Applying operating system principles to SDN controller design", In Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks (p. 2). ACM, (2013, November).

[21] "The POX platform," http://www.noxrepo.org/pox/about-pox/, 2012, accessed on November, 2012.

[22] Lantz, B. Heller, N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (p. 19), ACM, (2010, October).

[23] G. Pantuza, F. Sampaio, L.F. Vieira, D. Guedes, M. Vieira, "Network management through graphs in Software Defined Networks," In Network and Service Management (CNSM), 2014 10th International Conference on (pp. 400-405). IEEE, (2014, November).

[24] Mininet, 2015. [online]. Available: Http://www.mininet.org

[25] The POX controller source code. Available in https://github.com/noxrepo/pox