A Monitoring-based Load Balancing Scheme for Network Security Functions

Dongjin Hong*, Jinyong Kim*, Daeyoung Hyun†, and Jaehoon (Paul) Jeong‡
* Department of Electrical and Computer Engineering, Sungkyunkwan University, Republic of Korea
† Department of Software Platform, Sungkyunkwan University, Republic of Korea
‡ Department of Interaction Science, Sungkyunkwan University, Republic of Korea
Email: {dong.jin, timkim, dyhyun, pauljeong}@skku.edu

***Abstract*—This paper proposes an enhanced Interface to Network Security Functions (I2NSF) framework. To improve the whole packet throughput and manage resource of Network Security Functions (NSFs), the enhanced I2NSF framework monitors NSFs and distributes incoming packets to NSFs efficiently. Even if the legacy framework that provides security services using Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) has the similar NSFs, it is inefficient to be unable to distribute the packets to multiple NSFs. Based on the legacy I2NSF framework, therefore, we add two kinds of communication such as (i) communication between NSFs and security controller to monitor NSFs and (ii) communication between Security Function Forwarder (SFF) and security controller to perform the load balance for the packets to multiple NSFs. For the further communications between NSFs with security controller, we present a message format based on the information model proposed by Internet Engineering Task Force (IETF) I2NSF Working Group. We use capability data model proposed by IETF I2NSF WG, which describes the capability of an NSF. In order to show the feasibility of the proposed framework, we implemented the enhanced framework using IETF standards and open sources.**

Keywords*—*Software Defined Networking, Network Functions Virtualization, Monitoring, Load Balancing, Interface to Network Security Functions

I. Introduction

Computer networks are built with a complex combination of a lot of software including many protocols and hardware such as router, switch, hub, accelerator, and firewall. This complexity is not only difficult to find the event of an error, but also has a difficulty in management. According to Gartner, the number of Internet of Things (IoT) devices will be around 11 billion by 2018 [1]. It means that the number of smart devices for an individual or a household is increased and it brings a lot of traffic. As the number of smart devices are increased, the cycle will be shorter for institutions or companies that are replacing expensive network devices to accommodate unpredictable volume of traffic. It is also difficult to predict when they should be replaced. To resolve these problems, it should require a foundation for flexible network management and virtualization.

Consequently, Software Defined Networking (SDN) and Network Function Virtualization (NFV) are emerging to improve network management [2], [3]. The SDN, which can manage the network dynamically through software, has drawn significant attention from both industry and academia. The SDN is an effective solution for network traffic management by managing the network architecture. The industry and the academia are interested in the NFV that can be used by virtualizing network resources. Both technologies have created a synergy effect recently. Furthermore, both are important research and development challenges to be addressed. In particular, Internet Engineering Task Force (IETF) Interface to Network Security Functions (I2NSF) Working Group (WG) aims to define and to implement the standard interfaces. They provide network security services in network environments using SDN and NFV as the underlying infrastructure [4], [5]. Actually, the current network administrators create and manage high-level policies, which are usually made up of natural language that people who do not have expertise can easily understand, for responding to extensive network events and applications. Based on it, the administrators translate high-level policies into low-level policies for network devices and configure them for each network device and each vendor with very limited tools. The reason why the WG defines standard is to make it easy to manage interfaces for network security function.

The existing I2NSF framework provides security services by configuring policy to Network Security Function (NSF) such as Firewall, Intrusion Detection System, Intrusion Protection System, Anti-Distributed Denial of Service, and Anti-Virus based on policy creation [6]. However, we have to consider a limitations of the I2NSF framework. Even though the framework has other NSFs that have same function such as firewall and so on, it can be inefficient because it is impossible to distribute the incoming packets to multiple NSFs. This function is out of scope to I2NSF WG. And to the best of our knowledge I2NSF WG has not yet implemented monitoring. But the periodic and comprehensive monitoring NSFs plays an important role in terms of efficient use of the framework that provides security services. In this paper, we propose an enhanced framework to load balance the packets to NSFs and to monitor NSFs using NSF monitoring information model and data model for improving whole packet throughput and managing resource of NSFs [9], [10]. If the security controller monitors network traffic status and resource usage continuously among the monitoring information generated by the NSFs, the network traffic will be distributed to multiple NSFs. In other words, the traffic load for each NSF is reduce in the framework. It also leads to improve the throughput of framework. All of these functions can be added to the I2NSF framework without replacing or purchasing any other hardware and software.

The rest of this paper is organized as follows. Section II explains the I2NSF framework and analyzes the problem. Section III describes our proposed framework based on the problem of legacy framework. In Section IV, we describe the implementation of our proposed framework. We finally conclude this paper along with future work in Section V.

II. Legacy I2NSF Framework

This section describes I2NSF architecture and how security service is provided in the framework. A lot of researches [7]-[11] related to I2NSF framework have been reported. Our proposed framework also related to these researches and implemented based on them. Enns et al. [7] proposed the Network Configuration Protocol (NETCONF) to manage network devices for installing, manipulating, and deleting the configuration. The Extensible Markup Language (XML) based data encoding is used by NETCONF for configuration data as well as the protocol messages. Bjorklund [8] proposed the Yet Another Next Generation (YANG), which is a data modeling language for the NETCONF to model configuration and state data manipulated by NETCONF, NETCONF remote procedure calls, and NETCONF notifications. Our proposed framework also uses NETCONF and YANG when communicate NSFs with security controller and communicate Security Function Forwarder (SFF) with security controller.

Xia et al. [9] proposed an information model to monitor NSFs. And Hong et al. [10] proposed YANG data model to monitor NSFs. The monitoring information (e.g., events, alerts, alarms, logs, and counters) is used when the NSF communicates with security controller. Monitoring NSFs is important to the overall framework, if it is done in a timely and comprehensive way. It based on monitoring information generated by NSF. Our proposed framework performs load balancing based on monitoring. Thus, we constructed the payload of the packet based on these drafts.

Hares et al. [11] proposed Capability YANG data model for NSF, which can be used by NSF facing interface between the security controller and NSF, to inform the capabilities of NSF. It also utilized by I2NSF user to provide security controller with a list of the capabilities, which can be controlled by security controller. Our proposed framework uses this data model partly when SFF communicates with security controller.

Lopez et al. [12] proposed I2NSF framework to define standard interface for providing network security service. Our proposed framework is based on this framework.

A. I2NSF Architecture

This subsection describes the entities of I2NSF architecture. Fig. 1 illustrates I2NSF architecture, including three main interfaces. Developer's Management System (labeled as Developers Mgmt System in Fig. 1), which is utilized by a service provider, installs a security service package via registration interface. The security controller translates high-level to low-level policy. The SFF forwards packets from SDN network to appropriate NSF. The NSF provides security functions based on cloud environment and physical equipment. The consumer facing interface which is between I2NSF user and security controller allows the client to communicate with the security controller by using the RESTCONF [13] protocol. The NSF Facing Interface which is between security controller and NSF makes rule for specific NSF and allows the security controller to communicate with the NSF using the NETCONF protocol and YANG. The Registration Interface which is between security controller and Developer's Management in the framework allows the vendor to add NSF on the framework.

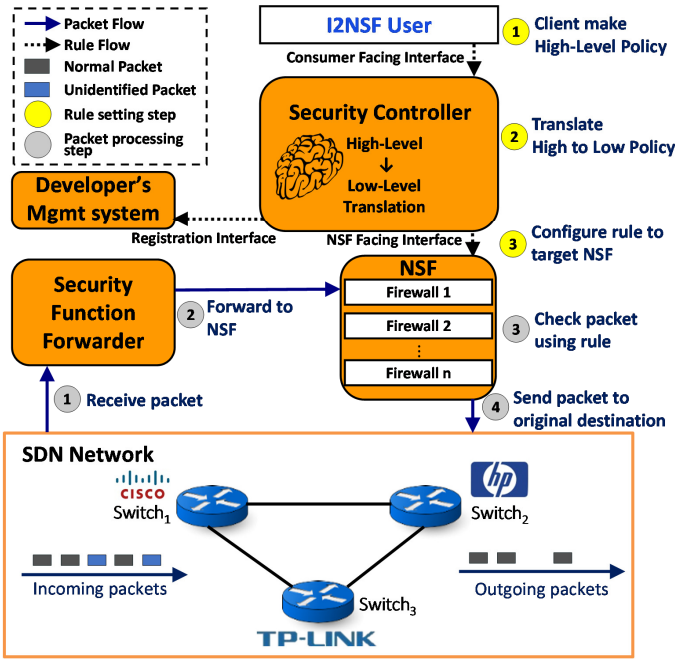

Fig. 1. Security service scheme of I2NSF framework

B. Creating and Applying Policy

This subsection describes how to create and apply policy and rule marked with yellow circle in the Fig. 1 as follows:

1) The administrator makes high-level policy using I2NSF client application and sends it to security controller using Consumer Facing Interface.
2) The security controller translates received high-level policy to low-level policy for NSF. And then sends it to target using NSF Facing Interface.
3) The target NSF is configured by the low-level policy.

C. Processing Packets Received from the SDN Network

This subsection describes how to handle packets received on the SDN network marked with gray circle in the Fig. 1 as follows:

1) The SFF receives unidentified packet from the switch by the SDN network.
2) The SFF forwards the packet to NSF.
3) When the packet arrived, the NSF uses the rule that is low-level policy to examine the packet and determine how to handle it.
4) The NSF sends the packet to original destination or blocks it.

The second step in Processing Packets Received from the SDN Network is not support load balance because it is out of scope to I2NSF. The monitoring case has not yet implemented.

Unlike the legacy I2NSF framework cannot provide monitoring and load balancing incoming packets, our proposed framework can provide these services. In the next section, we propose enhanced framework for above problem, which is based on the I2NSF framework.

III. Our Proposed Framework

This section explains our proposed framework in detail, including YANG data model, information model, and the process of monitoring NSF and load balancing incoming packets. Based on legacy framework, we try to improve the efficiency of processing packet by traffic load balancing.

A. YANG Data Model for Capability

This subsection describes YANG tree and data model for capability to use when the SFF communicate with security controller.

```
+--rw nsf-capabilities* [nsf-name]
   +--rw nsf-name                    string
   +--rw nsf-address
      +-- rw (nsf-address-type)?
         +-- :(ipv4-address)
         |   +--rw ipv4-address      inet:ipv4-address
         +-- :(ipv6-address)
             +--rw ipv4-address      inet:ipv4-address
```

Fig. 2. Capability YANG tree

```
list nsf-capabilities{
    key "nsf-name";
    leaf nsf-name {
        type string;
        mandatory true;
    }
    container nsf-address {
        choice nsf-address-type {
            case ipv4-address {
                leaf ipv4-address {
                    type inet:ipv4-address;
                    mandatory true;
                }
            }
            case ipv6-address {
                leaf ipv6-address {
                    type inet:ipv6-address;
                    mandatory true;
                }
            }
        }
    }
}
```

Fig. 3. Capability YANG data model

Fig. 2 and Fig. 3 show YANG tree and data model for capability based on the draft [11]. The asterisk mark and the question mark indicated in Fig. 2 mean list and value that can be determined below ipv4-address or ipv6-address. The square bracket means key value for list. Each information of NSF is stored in a list and the nsf-name value is used for key value. It also has version 4 or version 6 IP address.

B. Information and Data Model for Monitoring

This section briefly describes the information and data model proposed by IETF I2NSF WG for monitoring NSFs [9], [10] to construct the payload of the packet because there is not any YANG data model to monitor NSF for I2NSF framework.

TABLE I. Message Format for Monitoring NSF

Version	Type	Time	NSF_name	Vender_info	Severity

TABLE I shows the format of the monitoring information to be transmitted when the NSF periodically collect data and produce. NSF communicates with the security controller using this format. It has following items:

- Version: Indicates the version of the data format. The value is a two-digit decimal number starting with 01.
- Type: Use values such as event, warning, alarm, counter, and log. For the enhanced framework proposed in this paper, we need to use resource utilization logs that provides system state, CPU utilization, memory utilization, disk utilization, remaining disk space, and traffic information.
- Time: Indicates the time at which the message was generated.
- NSF_name: Indicates the name or IP address of the NSF that generates the message.
- Vendor_info: indicates the name of the NSF vendor.
- Severity: Indicates the severity of the message and consists of 8 steps from 0 to 7. The smaller the number, the higher the severity.

C. Monitoring NSF and Load Balancing Incoming Packet

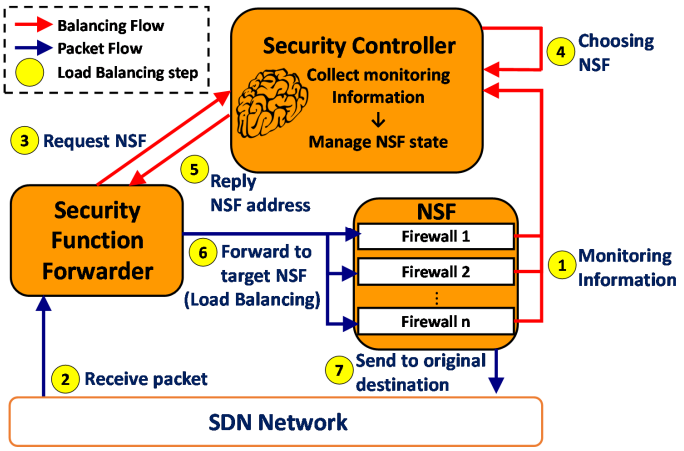

Fig. 4. NSF monitoring and load balancing scheme in our proposed framework

This subsection describes how to monitoring NSFs and to load balance incoming packets marked with yellow circle in the Fig. 4 as follows:

1) The security controller receives and manages periodically monitoring information (e.g., system state, CPU usage rate, memory usage rate, disk usage rate, remaining disk space, etc.) from NSFs, which registered in the framework.
2) The SFF receives unidentified packet from the switch by the SDN network.
3) The SFF asks the security controller to reply with where to send the received packet via NETCONF/YANG communication.

4) The security controller decides IP address of NSF based on monitoring information.
5) The security controller informs the specific IP address of NSF, which will process the packet, to the SFF.
6) The SFF forwards the packet to target NSF. At this point, load balancing occurs.
7) The NSF sends the packet to original destination or blocks it as same as legacy framework.

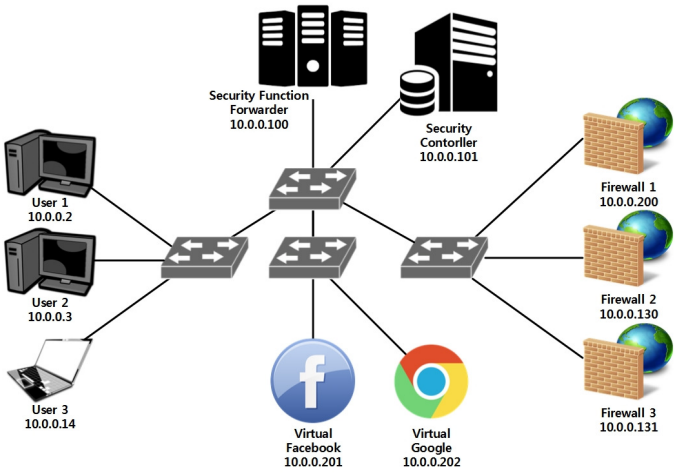

Fig. 5. Mininet topology

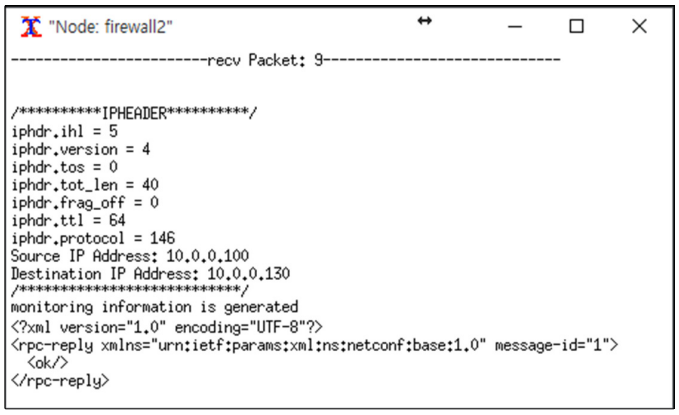

Fig. 6. Generating monitoring information and sending to security controller using NETCONF/YANG and XML

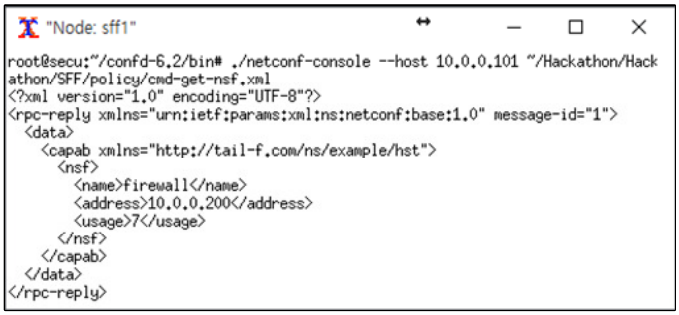

Fig. 7. Getting NSF information using NETCONF/YANG and XML

IV. Implementation

In this section, we explain the implementation to show the feasibility of the proposed framework. We constructed virtual network topology using Mininet [14], which can make virtual network, for the test bed. We also controlled the network traffic using OpenDaylight [15] that can set up policy to the switch.

A. Network Topology

This subsection describes the network topology that we implemented our proposed framework. Fig. 5 illustrates the network topology using Mininet. The topology can be divided into four parts. The left side where users are located can be considered SDN network. The right side where the firewalls is where the NSFs are gathered. The bottom with virtual Facebook and Google is assumed to be the external network. The upper side where SFF and security controller are located is where there are core components that can control the I2NSF framework. In this topology, the packet from left side must go through SFF because of the policy we set in advance using OpenDaylight.

B. Communication between SFF and security controller and among NSFs and Security Controller

This subsection shows the communications among NSFs and security controller and between SFF and security controller using NETCONF/YANG and XML. Fig. 6 shows the NSF generates periodically monitoring information, which configured in XML, and sends it to the security controller, then security controller sends rpc-reply to the NSF. It means the security controller, which is NETCONF server, has received and processed the monitoring information. Fig. 7 shows the SFF requests NSF information to the security controller that has a list of NSFs and manages using the YANG data model described earlier in the Section III-A to get where to send the incoming packet. Then security controller sends rpc-reply, which include NSF information, to the SFF. Based on it, the SFF load balances the incoming packets. The XML parser modifies all XML that used by NETCONF server depend on the situation.

C. Implementation Result

The testing occurs in the following situation: The user1 on SDN network want to connect to Facebook or other website. Fig. 8 illustrates that the packets sent by the user should go to firewall through the SFF, then the firewall examines the packet and sends backs the packets to SFF with inspection result. After receiving packet from firewall, if the inspection result has default actionCode (00) and metadataNum (00), the SFF sends the packet to the original destination. In this situation, the firewall2 dose not receive any packets from SFF because the default forward target is firewall. After a certain period of time, the firewall generates monitoring information and notifies the security controller like Fig. 7 and SFF requests security controller to where the packet should go. The security controller determines firewall, which has minimum usage, and replies to SFF like red box marked in Fig. 9. It also illustrates firewall2 begins to receive packets from SFF after the SFF chooses firewall2. The proposed framework we implemented repeats the above steps and performs load balancing based on monitoring NSF.

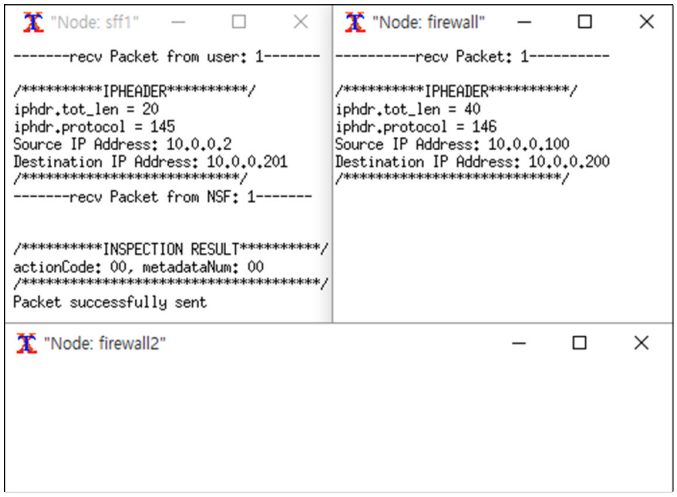

Fig. 8. SFF receives a packet from user and forwards to NSF

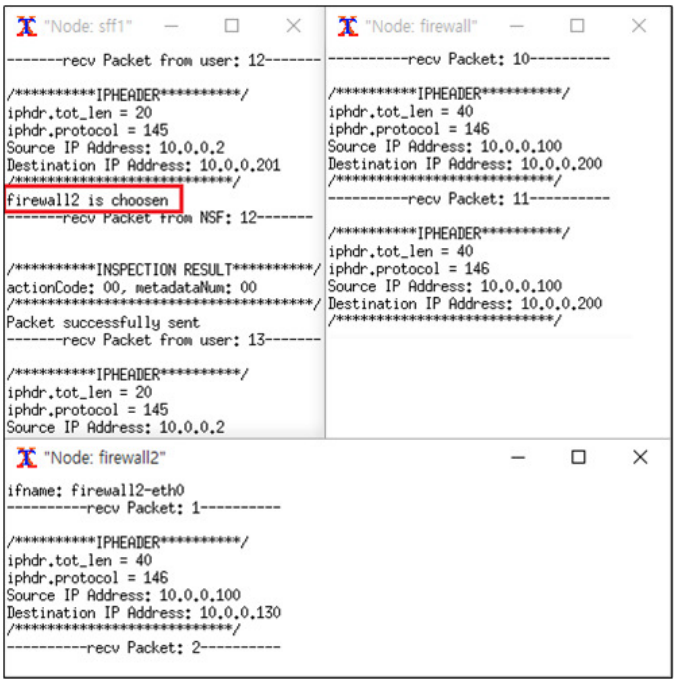

Fig. 9. SFF selects firewall2 based on monitoring information

V. Conclusion

This paper proposes an enhanced Interface to Network Security Functions (I2NSF) framework based on legacy I2NSF framework to monitor Network Security Functions (NSFs) and to distribute incoming packets to NSFs for improving whole packet throughput and managing resource of NSFs. Even if the framework has the similar NSFs, it is inefficient to be unable to distribute packets to multiple NSFs. We believe that the enhanced framework can improve processing efficiency and traffic flow in the framework. It also can contribute to standardization.

For future work, we will move our implemented network topology using Mininet to OpenStack environment to show our proposed framework in real world not simulation. Then, the NFV manager supported by OpenStack can prevent the potential bottleneck when the security controller receive monitoring information from NSFs. We will also study cache which manages addresses of NSFs in Security Function Forwarder (SFF) to improve the performance, since the overhead due to additional steps by monitoring and load balancing is not considered in this paper. Finally, we will use whole monitoring data model.

Acknowledgments

This research was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (Ministry of Science and ICT, MSIT) (No.2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning). This research was supported in part by the MSIT under the ITRC (Information Technology Research Center) support program (IITP-2017-2017-0-01633) supervised by the IITP. Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] Gartner, "Gartner's 2014 Hype Cycle," accessed 2017. [Online]. Available: http://www.gartner.com

[2] ONF, "Software-Defined Networking: The New Norm for Networks," ONF White Paper, 2012.

[3] ETSI-NFV, "Network Functions Virtualization (NFV); Architectural Framework," ETSI GS NFV 002 V1.1.1, Oct. 2013.

[4] IETF, "The Internet Engineering Task Force," accessed 2017. [Online]. Available: https://ietf.org

[5] IETF I2NSF Working Group, "Interface to Network Security Functions(I2NSF)," accessed 2017. [Online]. Available: https://datatracker.ietf.org/wg/i2nsf/charter

[6] K. Jinyoung, M. D. Firoozjaei, J. P. Jeong, H. Kim, and J. S. Park, "SDN-based Security Services using Interface to Network Security Functions," Proceedings of the 7th International Conference on ICT Convergence (ICTC), pp. 526529, Oct. 2015.

[7] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," IETF RFC 6421, Jun, 2011.

[8] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," IETF RFC 6020, Oct. 2010.

[9] L. Xia, D. Zhang, Y. Wu, R. Kumar, A. Lohiya, and H.Birkholz, "An Information Model for the Monitoring of Network Security Functions(NSF)," in draft-zhang-i2nsf-info-model-monitoring-03.pdf, IETF I2NSF WG, Mar. 2017.

[10] D. Hong, J. Jeong, J. Kim, S. Hares, and L. Xia, " I2NSF Network Security Function Monitoring YANG Data Model," in draft-hong-i2nsf-nsf-monitoring-data-model-00.pdf, IETF I2NSF WG, Jul. 2017.

[11] S. Hares, R. Moskowitz, L. Xia, J. Jeong, and J. Kim, "I2NSF Capability YANG Data Model," in draft-hares-i2nsf-capability-data-model-01.pdf, IETF I2NSF WG, Mar. 2017.

[12] D. Lopez, E. Lopez, L. Dunbar, J. Strassner, and R. Kumar, "Framework for Interface to Network Security Functions," in draft-ietf-i2nsf-framework-04.pdf, IETF I2NSF WG, Oct. 2016.

[13] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," IETF RFC 8040, Jan, 2017.

[14] Mininet, "An instant virtual network on your laptop," accessed 2017. [Online]. Available: http://mininet.org

[15] OpenDaylight, "Open Source SDN Platform," accessed 2017. [Online]. Available: http://www.opendaylight.org/