Flowtracker: A SDN Stateful Firewall Solution with Adaptive Connection Tracking and Minimized Controller Processing

Conference Paper · May 2016

DOI: 10.1109/ICSN.2016.7501925

CITATIONS
3

READS
40

2 authors:

Vinh Thuy Tran

7 PUBLICATIONS **13** CITATIONS

SEE PROFILE

Heejune Ahn

Skolkovo Institute of Science and Technology

54 PUBLICATIONS **403** CITATIONS

SEE PROFILE

FlowTracker: A SDN stateful firewall solution with adaptive connection tracking and minimized controller processing

Thuy Vinh Tran
Department of Electrical and Information Engineering
Seoul National University of Science and Technology
Seoul, Republic of Korea
tranvinhthuy@seoultech.ac.kr

Heejune Ahn
Department of Electrical and Information Engineering
Seoul National University of Science and Technology
Seoul, Republic of Korea
heejune@seoultech.ac.kr

***Abstract*—The introduction of Software Defined Networking (SDN) enables possibilities for the next generation of network where the network logic operation is separated from the constraints of underlying hardware. However, the new architecture of SDN also exposes many security risks such as controller DoS attack, configuration channel compromise. This paper analyzes the challenges of stateful firewall realization in SDN environment and presents FlowTracker – a novel stateful firewall solution focusing on maintaining the accuracy and agility of stateful firewall with reduced controller processing and communication overhead between control and data plane. The GENI test bed experiments validates FlowTracker its stateful packet tracking and acceptable level of latency increase.**

Keywords—SDN; Overflow; Firewall; Stateful firewall; connection tracking; POX controller; GENI testbed

I. INTRODUCTION

Software define networking (SDN) promises the future of faster evolving, easy to maintain and hardware independent network. Albeit the active research and development along with wide industry adaptation, SDN still raises discussions and concern about network security. The central control server unifies the network features and processes; however, this new network paradigm also introduces emerging problems such as controller scalability and performance degradation, single point of failure. SDN security survey in [1] categories 10/13 SDN securities risks affect control plane while the work in [3] lists compromised controller attack as one of the new issues that SDN network is exposed to.

Firewall is a network entity directly affected from the changes in SDN network structure. The flow concept in SDN allows forwarding devices to treats packets with similar connection traits in the same manner instead of perform action on individual one. Accordingly, each forwarding devices can maintain a set of flow rules representing firewall policy and behave as a distributed firewall. From Fig. 1 we can see that stateless firewall suits the control plane-data plane separation architecture perfectly [14]. Since stateless firewall only operates in Datalink and Network layers thus the involvement of controller in stateless firewall goes as far as setup firewall rules in forwarding device while the firewall operation require minimum interaction. This is not the case with stateful

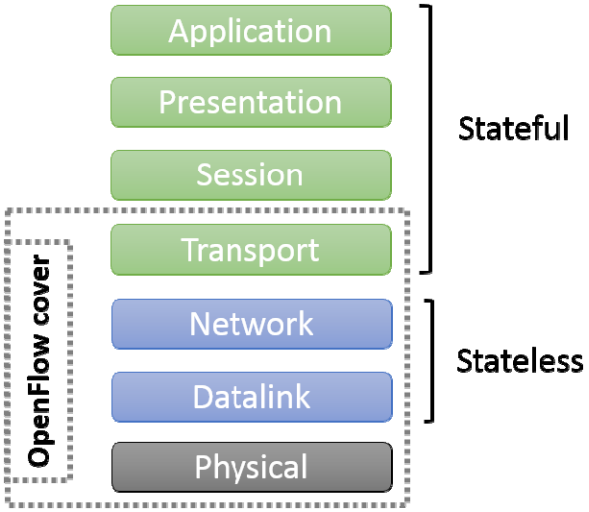

Fig. 1. Overall SDN structure in correalation with network layers

firewalls. Stateful firewall detect and monitor every connections in the network, including the information of originator host, connection setup, teardown etc. In order to do that, stateful firewall need to access the data in Transport layer. This require frequent communication between the control plane and data plane. The requirement of actively controller participation in stateful firewall operation could potentially cause the surge in controller overhead and amplify the previously mentioned issues. That being said, stateful firewall is not incompatible with SDN design; however, the approach to SDN stateful need to consider the controller-data plane communication overhead.

This research presents a detailed analysis of the challenges in establish an efficient stateful firewall solution for SDN. To the best of our knowledge, yet a literature work looks into these issues and suggest the improvement for SDN stateful firewall. With this intention, we propose a comprehensive SDN solution aiming at accurate and timely tracking of network connections without sacrificing the controller performance and network resource.

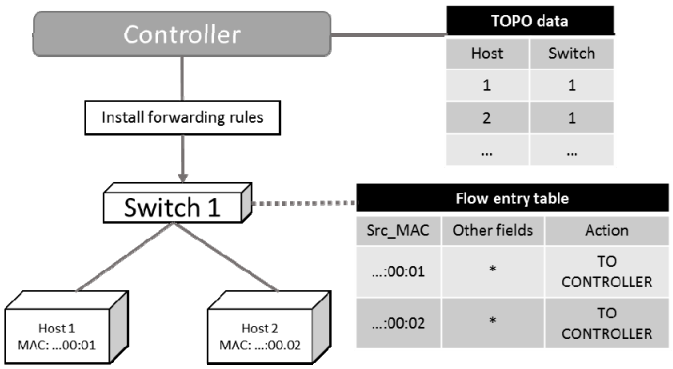

Fig. 2. Selective flow rule installation for new connection detection

II. SDN STATEFUL FIREWALL CHALLENGES

The key function of stateful firewall is to monitor state of each connection in the network and to filter the packets by matching packet header info with existing connections. The decision of action to perform on the packet entirely depends on the validity of the connection state and trust level of the initiator. E.g. if the packet does not belong to any existing connections or belong to a terminated one, then the packet will be marked as suspicious and dropped.

The complete separation between network control logic and underlying forwarding hardware in SDN environment means that the hardware switches cannot detect and record the connection state by themselves without redirecting a number of connection packets to controller for handling this logic. This inadvertently amplifies the weak points of SDN environment since the centralized controller has to deal with watching all the traffic in the network, which likely leads to overhead problem.

We summarize the current obstacles with SDN stateful firewall in the next subsections below.

A. *Detect new connection*

Stateful firewall are capable of detecting every connection initiated in the network. For a centralized control logic of SDN, that means a burdensome task for controller. Since the function of SDN data plane is solely forwarding packets according to the flow rules defined by centralized controller, to realize connection tracking in SDN there is no other way than relying on the controller examining the packet headers redirected from underlying hardware. It is impossible for all the network switches forward all network traffic to controller for monitoring and catching a new connection just appear. The controller overhead could lead to a severe increase in latency [4]. Efficiently watching and taking record of the network connections still an open issue for SDN stateful firewalls.

B. *Monitoring state of connections in network*

This challenge also tightly connects to the controller overhead issue explained above. However, monitoring connection state is even more challenging than catching the new connections in network. There is abundant number of traffic traces and patterns to keep track of. Though OpenFlow and as result SDN stateful firewall only operates on layer 2 to layer 4, the numbers of existing protocols already rack up to hundreds. There are naturally stateful protocols like TCP that we can take advantage of the TCP flags in packet for state tracking. However, many protocols using UDP, which have stateless trait and provides no way to track connection state timely and accurately. Correct tracking is even more important in removing the connection in connection list right after detecting connection termination. Otherwise, the network could be exposed to relay attack when the hacker tries to retransmit the modified packets belonging to the terminated connection that was approved by the firewall.

C. *Minimizing monitoring overhead*

Monitoring state of the connection involves packet extraction for state information e.g. TCP flags. An additional delay for end-to-end connection is inevitable. The controller has to reduce this delay in order to meet the demanding QoS, especially for UDP streaming.

III. FLOWTRACKER : PROPOSED SOLUTION

We examine the three issues above one by one and introduce the counter mechanisms for each individual problem.

A. *Topology learning based for selective flow control rule installation approach*

The idea of selective flow rules installation by topology learning was first applied in our previous work [8] for stateless firewall for shorten firewall setup time and reduce traffic redundancy. With similar approach, FlowTracker populates and utilize the network topology data to selectively install flow rules in each switch so that when a host sends the first packet to the closest switch, it will be redirected to controller for examine the new connection request. The process is illustrated in the Fig. 2.

After installing the forwarding rules in the switch 1 table, we can see that the entire packets originated from host 1 or host 2 will be forwarded to controller and if they belong to a new connection, the controller can be aware of it right away. Moreover, these selective flow rules also work well for differentiating the traffic flows through the switch. As the source MAC address of the host is specified in flow rule, the traffic comes from other switches are unaffected and will be forwarded to next routing node normally without referring to the controller. In other words, the switch at the initiation of the connection will be in charge of noticing the controller while other switches in the connection travel route only simply forward the traffic.

B. *Adaptively monitoring connection state*

As stated before, traffic in layer 4 can be stateless or stateful protocols. Here we consider two representative protocols: TCP, which is connection-oriented and UDP, a common connectionless protocol. Stateless protocols and stateful protocol have their own characteristic so we need to handle them separately.

1) *For TCP connections*

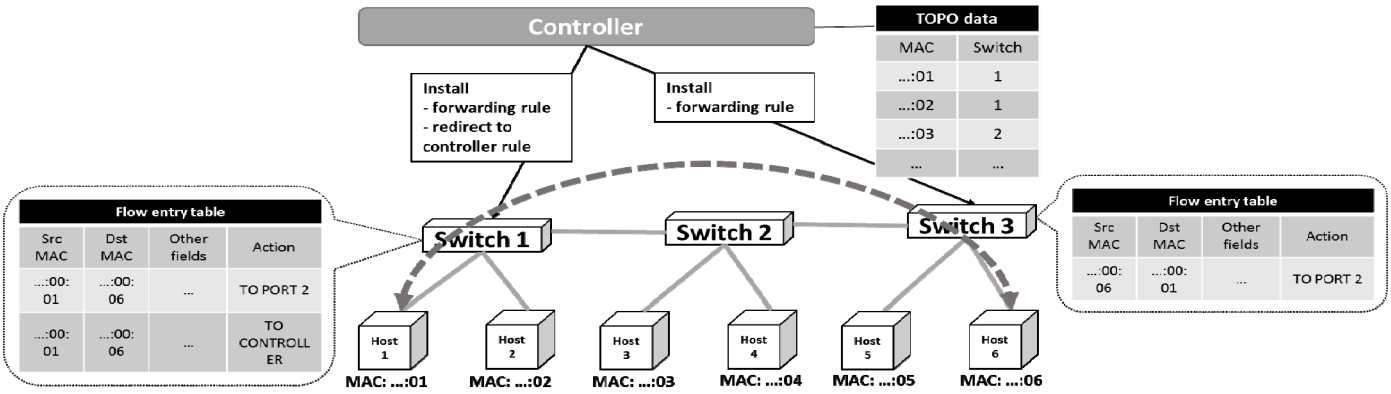

Fig. 3. Tracking connection with minimum end-to-end delay

TCP protocol is connection-oriented protocol itself so the state of a TCP connection can be firmly defined. In the establishment phase of a TCP connection, a host will send the first TCP packet with TCP flag SYN set to true in order to indicate a new TCP session beginning. After that, the receiver will reply with a TCP packet, which has SYN flag and ACK flag set to true. For the TCP connection termination, the TCP notification packet will have FIN flag set to true.

We can see that by analyzing the TCP flags we can have definite tracking of TCP connection state. So for TCP connections, the firewall logic in the controller will make sure to examine each packet forwarded from the switches to controller to catch the new connection and be alerted when one of the tracking connection ends.

2) For UDP connections

Unlike TCP, the state of an UDP connection cannot be conclusively defined since UDP itself is a connectionless protocol. Because UDP packets do not have sequence number or indication flag fields whatsoever, the firewall only can track the connection in a pseudo-stateful manner. The detection of new connection is through considering unique combination of UDP packet header fields and then comparing with previous recorded connections characteristics for deciding to add a new connection record. Similarly, to detect connection termination we have to base on connection timeout for removing UDP connection off the connection list.

For detecting new UDP connection, the firewall logic will operate the matching process for each UDP packet coming from the data plane to controller. The controller will compare the packet header fields with the corresponding fields of each connection in connection list of controller. If there is no match then the new connection will be added to the connection-tracking list of controller.

To be informed of the connection tear down, we can utilize the timeout feature of OpenFlow flow entry. By setting timeout for the flow entry matching the connection traffic, after specified time without any packet belonging to the UDP connection comes through the switch, the flow entry will be removed. Consequently, the switch will fire a FlowRemoved event to notify the controller about the entry removal. Because the FlowRemoved event includes the matching field information, the stateful firewall logic in controller can find the corresponding connection record and remove it from its list. The switches can detect connection termination using flow rule timeout without forwarding packets to controller for checking. This mechanism compliment UDP quite well as generally speaking, UDP protocol is used where the connection speed and QoS demand are put into forefront.

C. Tracking without interfere the connection and minimizing end-to-end delay

After detecting the new connection, the stateful firewall logic will install the flow rules in switches so the traffic can begin exchanging between the two hosts of the connection. Controller topology map again will be exploited here to rapidly collect the two switches information directly connected to the source and destination host. From now on, we denote them as source switch and destination switch.

Considering the traffic between two hosts, the packets going into each host can flow smoothly through the network since they are not matched with the flow rules used for catching new connections. However, for the traffic coming out of each host, we need flow rules with higher priority than catching connection flow rules so the packets belong to existing connection can bypass new connection checking. With this intention, the controller will install one flow rule in each switch found above. The source MAC address field will be set to the host directly connect with the switch and the destination MAC address will be set to the host on the other edge of connection.

However we have to consider the connection state differences between TCP and UDP stated in previous section to set the appropriate action for the flow rule in order to detecting the connection termination.

For UDP connection, we use flow timeout, thus the flow rule action can be simply set to corresponding forwarding port together and additionally specify idle timeout.

For TCP connection, to detect the connection ending accurately we have no other choice than redirect the traffic to controller for TCP flag checking. The redirecting is only needed in one side of the connection hosts since TCP connection termination is bi-directional thus it can be detected from TCP packet with SYN flag set in either sides. Therefore we install flow rule with action "FORWARD TO CONTROLLER" in the host initiating the

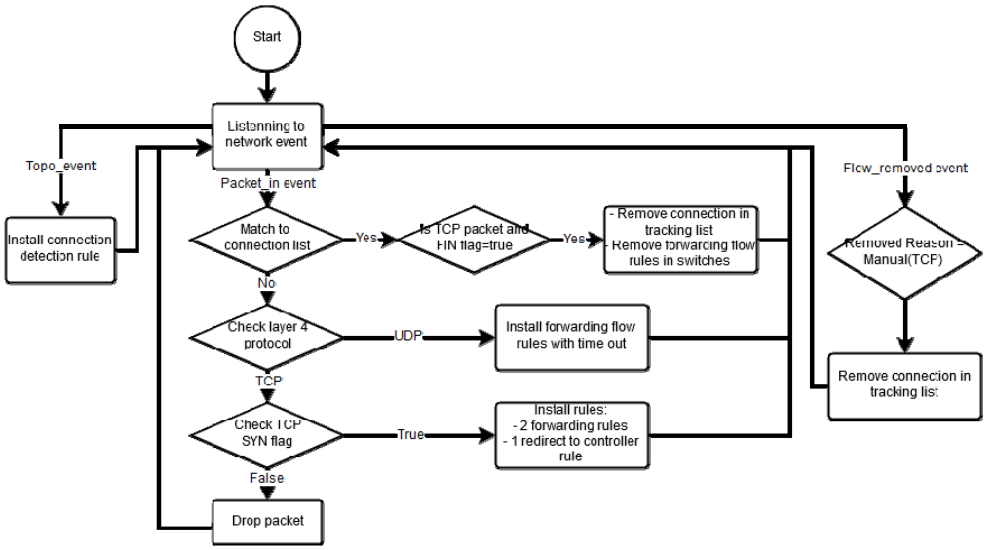

Fig. 4. Overall operation flowchart of FlowTracker

connection while install the flow rule with action set to specific forwarding port in the other host.

We notice that forwarding every packets to controller will increase the latency significantly. We apply a counter solution for this issue by using double action rules. By installing additional flow rule with action forwarding to specific port together with action FORWARD_TO_CONTROLLER, the switch will forward the packet to destination along with forwarding a copy packet to controller at the same time. As a result, the delay in waiting to send packet to controller and receive back controller decision will be eliminate.

The overall process is demonstrated in fig. 3.

D. *The complete FlowTracker solution*

After covering all problems in the previous section, now we would like to present a complete flow of SDN stateful firewall logic. The flow chart in fig. 4 describe the operation of SDN stateful firewall in details.

IV. EXPERIMENT RESULT

A. *Experiment setup*

We performed the experiment in GENI testbed [16]. We assert the performance of new stateful firewall solution on a linear topology of open virtual switches (OVS), each switch connects to four hosts. FlowTracker is implemented on POX controller [17]. Therefore, to demonstrate the performance level of FlowTracker, we choose the same topology network with learning switch module as performance baseline. In the learning switch module, the solely control logic function is layer 2 forwarding.

As a result, when we compare the performance level of SDN network running learning switch module against FlowTracker, the difference will reflect the controller overhead and network performance shifting when bearing the additional stateful firewall solution.

B. *Experiment result*

1) *Stateful filtering test*

To test the stateful firewall function, we experiment with the generic scenario where connection tracking function of stateful firewall shows it strength: only allow hosts in the network whitelist to initiate new connections, otherwise reject the connection if the initiator is a untrusted host. This security requirement is certainly infeasible with stateless firewall.

We setup host 2 mac address in the whitelist while host 8 mac address is in blacklist of FlowTracker. Using *hping3* [18], we simulate the new connection from each host. The host logs in fig. 5 show that host 2 can successfully start a new connection with TCP while conversely the similar attempt from host 8 is failed. Also the controller log in fig. 5 shows the connection from host 2 is accepted by FlowTracker and the connection information is stored while the connection request from host 8 is dropped after matching the firewall blacklist.

2) *Latency test*

For TCP case, we simulate an increasing number from 10 to 1000 simultaneous connections in the network by set up webserver on ten hosts and send HTTP requests from the rest at the same time.

We can observe from fig. 6 that the download time differences between FlowTracker and LearningSwitch vary from 22.26ms to 47.36 ms for the number of simultaneous connection from ten to one hundred. The recorded times of FlowTracker are in the range of 30.62 ms to 70.76 ms while these of LearningSwitch are 7.8ms to 23.4ms.

To have a deeper insight, we calculate the increasing percentage of delay time caused by stateful firewall logic after each step and obtain the mean value. The average increasing rate of the delay caused by FlowTracker is 8.95%. Note that the numerical values are from the experiments on a XEN virtual machine of limited processing power, not a real network setup.

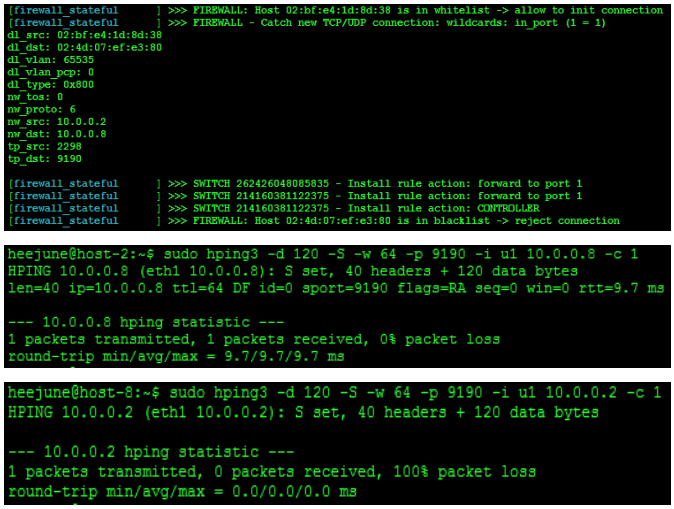

Fig. 5. FlowTracker and host logs screenshot

V. CONCLUSION AND FUTURE WORK

In this paper, we present a stateful firewall solution for SDN environment, toward the goal of accurate and active connection tracking while reducing end to end delay effect and controller operation overhead. Through the implementation and experiments, FlowTracker proves the ability to detect and monitor the network connection in a correct and timely manner, preserving the security requirement for a stateful firewall. The additional delay, processing resource for packer examining by FlowTracker is inevitable and the performance benchmark yields acceptable latency increase, and we believe that this is a worthy tradeoff regarding the SDN network with high security demand and real-time connection supervision is a priority. For the future work of this research, we are working on an algorithm to utilize the connection information collected by FlowTracker for activity profiling and DoS detection.

ACKNOWLEDGMENT

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program (IITP-2015-R0134-15-1030) supervised by the IITP (Institute for Information and Communication Technology Promotion)".

REFERENCES

[1] Scott-Hayward S., Natarajan S., Sezer S., "A Survey of Security in Software Defined Networks", IEEE Communications Surveys & Tutorials issue 99, July 2015

[2] Kreutz D., Ramos F.M.V., Esteves Verissimo P., Esteve Rothenberg C, "Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE, pp. 14–76, January 2015.

[3] Dabbagh M., Hamdaoui B., Guizani M., Rayes A., Software-defined networking security: pros and cons, IEEE Communications Magazine, vol. 53, June 2015

[18] Hping3 security tool, http://www.hping.org/hping3.html

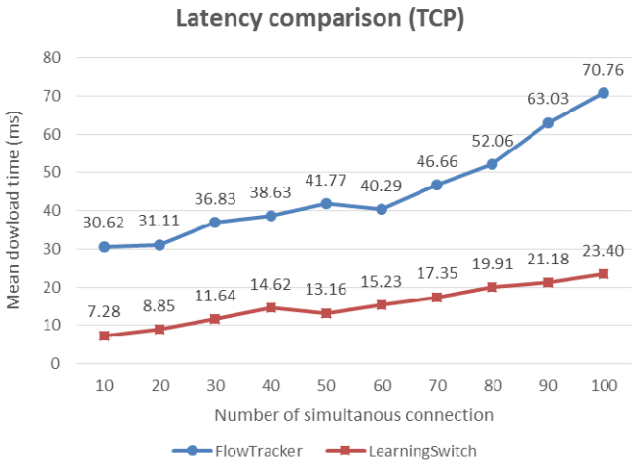

Fig. 6. Latency comparison – FlowTracker tracking TCP connections

[4] Seungwon Shin, Guofei Gu, "Attacking software-defined network: a feasibility study", ACM SIGCOMM workshop on Hot topics in SDN, August 2013

[5] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, Haiyong Xie, "A Survey on Software-Defined Networking", IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17, NO. 1, FIRST QUARTER 2015 (page 11-12).

[6] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti, "A Survey of Software-Defined Networking Past, Present, and Future of Programmable Networks", IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 16, NO. 3, THIRD QUARTER 2014 (page 6).

[7] Fei Hu, Qi Hao, and Ke Bao, "A Survey on Software-Defined Network andOpenFlow: From Concept to Implementation", IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 16, NO. 4, FOURTH QUARTER 2014

[8] Thuy Vinh Tran, Heejune Ahn, "A network topology-aware selectively distributed firewall control in SDN", ICTC International conference, Oct 2015

[9] Lara A., Kolasani A., Ramamurthy B., "Network Innovation using OpenFlow: A Survey", IEEE Communications Surveys & Tutorials vol. 16, pp. 493-512, 2014

[10] Scott-Hayward S., O'Callaghan G., Sezer S., "Sdn Security: A Survey", Future Networks and Services (SDN4FNS) 2013 IEEE SDN, pp. 1-7, 2013

[11] Justin Gregory V. Pena and William Emmanuel Yu, "Development of a Distributed Firewall Using Software Defined Networking Technology", Information Science and Technology (ICIST), 4th IEEE International Conference, April 2014, pp.449–452.

[12] Kaur K. , Kumar K. , Singh J. , Ghumman N.S., "Programmable firewall using Software Defined Networking," in Computing for Sustainable Global Development (INDIACom) International Conference, March 2015, pp.2125-2129

[13] Sakir Sezer, Sandra Scott-Hayward, and Pushpinder Kaur Chouhan, "Are We Ready for SDN? Implementation Challenges for Software-Defined Networks" in Communications Magazine, IEEE, July 2013, pp.36-43

[14] Lori MacVittie, "SDN Prerequisite: Stateful versus Stateless", DevCentral, https://devcentral.f5.com/articles/sdn-prerequisite-stateful-versus-stateless, April 2014

[15] OpenFlow.org, "OpenFlow specification v1.1.0", February 2011, http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

[16] GENI Experimental testbed, "http://www.geni.net/"

[17] NOXRepo.org, POX documentation, http://www.noxrepo.org/pox/documentation/