Comprehensive Prediction Models of Control Traffic for SDN Controllers

Bong-yeol Yu, Gyeongsik Yang, Chuck Yoo
Department of Computer Science and Engineering
Korea University
Seoul, Republic of Korea
{byyu, ksyang, chuckyoo}@os.korea.ac.kr

***Abstract*—In SDN, as the control channel becomes a performance bottleneck, modeling the control channel traffic is important. Such a model is useful in predicting the control channel traffic for network provisioning. However, previously proposed models are quite limited in that they assume only the forwarding function of a specific controller for their models. To overcome the limitations, first, this paper analyzes the control traffic by seven functions (including forwarding function) of a controller. Then, we build a seven-function model to predict control channel usage and evaluate the prediction accuracy that achieves as high as 94%. Note that previous models did not have any quantitative evaluation. Our model is built with the Open Network Operating System (ONOS) controller and extended to Floodlight and POX controllers. We show that the extended model also achieves similar prediction accuracy (95%). Furthermore, we compare three controllers in terms of control channel usage through our model.**

***Index Terms*—Control traffic, Control channel, OpenFlow, Software-defined networking, SDN controller, Scalability**

I. Introduction

Software-defined networking (SDN) has become a fundamental technology for future networking architectures. The fifth-generation wireless system (5G) networking architecture applies SDN to achieve flexible networking [1]. In addition, modern data centers benefit from the advantages of SDN, such as fast network reconfiguration and reduced configuration errors [2]. SDN flexibility comes from separation of the data plane from the control plane of the network infrastructure; these separated planes communicate via a so-called southbound interface. OpenFlow is a representative example, and OpenFlow control messages constitute the control channel between the separated planes.

In SDN, all network configurations are performed using the controller; thus, the control channel can be a performance bottleneck. Curtis et al. [3] noted that the control channel bandwidth scalability can cause significant problems for SDN, and various studies have been conducted to investigate this problem. In [4], [5], the forwarding behaviors of the Open Network Operating System (ONOS) [6] and OpenDayLight (ODL) controller [7] were modeled and the control channel was evaluated. However, both works are limited, because their evaluations were restricted to a specific controller. Furthermore, only the forwarding function of the SDN controller was considered, although the SDN controller performs multiple management functions more than just forwarding. Traffic monitoring to perceive the changes in the network state, link discovery, and flow expiration are examples of the functions.

Considering the above-mentioned limitations, we conduct a comprehensive analysis of ONOS control traffic, categorizing this traffic into seven management functions as shown in Fig. 1. Based on the analysis of the functions, we build a control channel usage prediction model and evaluate the model accuracy through Mininet with fat-tree and Internet Service Provider (ISP) topologies. We also extend the traffic model to Floodlight and POX controllers. Through experiments, we show that the models for both controllers achieve a similar prediction accuracy to ONOS. Finally, we compare three controllers in terms of the control channel usage.

Thus, the contributions of our study are three-fold:

- First, we analyze the control traffic of the ONOS controller by seven functions.
- Second, we build a prediction model of seven functions with 94% accuracy, as demonstrated through experiments.
- Third, we build the models to Floodlight and POX controllers and show that similar accuracy is achieved.

This paper is organized as follows. Section II discusses the background and motivation of our study. The management function categorization and developed models are described in Section III. Our evaluation results and analyses are presented in Section IV. Finally, Section V concludes the paper.

II. Background and motivation

SDN is a network structure in which the control and data plane are logically separated. The control plane is centralized to control and manage the data plane. In SDN, the control channel is an important resource as it allows the logically centralized control plane to communicate with the data plane. For example, the control plane gathers the data plane's hardware information or the network flow statistics to manage the physical network and controls these flows by implementing network policies in the form of flow rules for the data plane.

*This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (2015-0-00288, Research of Network Virtualization Platform and Service for SDN 2.0 Realization). This research was also supported by the MSIT(Ministry of Science and ICT), Korea, under the SW Starlab support program(2015-0-00280) supervised by the IITP.

978-1-5386-4633-5/18/$31.00 ©2018 IEEE

A. Control channel protocol

Various control channel protocols (e.g., OpenFlow [8], ForCES [9], and NetConf [10]) exist, but OpenFlow is arguably the most famous and most widely used. The OpenFlow protocol is used in many controllers. The controller sends the OpenFlow control messages to a switch in order to add rules or configure settings, and the switches send OpenFlow messages to the controller in order to inform the controller of statistical information or an uninstalled or expired rule.

In the OpenFlow protocol specification, messages are categorized into three types according to the subject who sends the control message, i.e., the controller-to-switch messages are sent by a controller to a switch and the response from the switch is required in some form, e.g., as statistics, *packet_out*, or *flow_mod* messages. In addition, asynchronous messages are sent by a switch to a controller to provide notification of arrival of a new packet, removal of a flow rule, or a switch state change, e.g., the *packet_in* message. Asynchronous messages do not require a controller response. Finally, symmetric messages are those sent by a controller and switch with the same message type, e.g., the hello or echo messages.

However, this control message categorization is inadequate to describe the controller behavior and analyze the control traffic. Most controllers include various functions for physical network management, and the control messages described above are used for these functions. Therefore, from the controller perspective, the categorization in the OpenFlow protocol specification creates difficulty in analyzing the controller and predicting the control traffic volume.

B. Control channel scalability analysis

The control channel traffic scalability problems were first highlighted in the DevoFlow [3] based on simple formulations. The authors divided the controller functions into two categories: 1) central visibility and 2) central control functions. The central visibility functions include those used by a controller to obtain visibility information, such as bandwidth, the number of passed packets and flow paths, and overall flows. In contrast, the central control functions include those used by the controller for network flow management, such as those related to routing, flow rule installation, and removal. However, their limitation is that detailed categorization of each function in terms of control traffic and the volume of traffic required for each function was not performed in [3], except the case of flow rule installation and removal. To install a bi-directional flow, the aforementioned study determines that the controller must exchange 2N+4 messages between itself and the N-switches; this loads the network with 94+144N bytes based on OpenFlow 1.0. However, as previously mentioned, the control traffic for all other management functions, except for forwarding, was not considered.

The studies [4], [5] analyzed the ONOS and ODL controllers' forwarding behaviors, respectively, and evaluated their control traffic scalability. The number of packets for forwarding and the upper and average bounds of the amount of traffic with the forwarding path length were estimated. However, the control traffic only for packet forwarding was considered. The SDN controller performs many other management functions; thus, it is important to also analyze the effects of those functions on the network traffic. In addition, these studies did not have any quantitative evaluation.

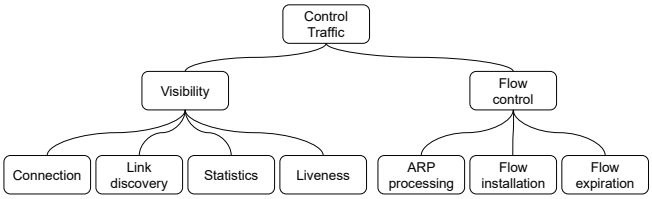


Fig. 1: Management function categorization

TABLE I: Notations used in control traffic model design

Notation	Description
s	Number of switches
p	Number of physical ports attached to all switches
h	Number of hosts
f	Total number of flows
hd_i	Hop distance of i-th flow
$data_i$	Size of first packet of the i-th flow

III. Control traffic model

In this section, we build our model with analysis of OpenFlow-based control traffic. First, we analyze the ONOS, which has many management functions, and present a corresponding control traffic prediction model. We also analyze two other well-known controllers, Floodlight and POX, to cover common SDN controller management functions. We build the models when three controllers use default apps. For example, the ONOS uses *fwd* and *proxyARP* for forwarding and ARP processing, respectively, and Floodlight uses *Forwarding* for forwarding. On the other hand, POX does not provide rich functions in default because it is an SDN platform for rapid prototyping and experiments. For model creation, we analyze the packets captured between a controller and emulated physical network when traffic is flowing through the network.

A. Control traffic model for the ONOS

We analyze the ONOS control traffic and develop a mathematical model based on our analysis results. Then, we categorize the controller management functions into control and visibility functions as in [3]; however, we further subdivide them into seven functions (Fig. 1). To derive the control traffic model, the notation listed in Table I is used.

*1) **Visibility**:* Considering the ONOS management functions, we categorize the visibility functions into four functions: connection, link discovery, liveness, and statistics.

Connection: Connection refers to the traffic used to connect a controller with a switch. In this phase, a controller establishes connection with a switch via two-way handshaking, while also configuring the switch and obtaining the switch's specification. These functions are only required when a switch initially connects with the controller. In general, the ONOS and switch are connected by exchanging hello messages. Then, the controller requests the switch's information and receives the

TABLE II: ONOS control traffic prediction model

Control traffic types	Number of packets	Total traffic (bytes)	Conditions
Connection(s,p)	$21 \times s$	$3012 \times s + 48 \times p$	Connection with s switches
Link discovery(s, p, h)	$2 \times (2p - h)$	$676 \times p - 334 \times h$	Cycle of 3 seconds
Statistics(s, p ,f, hd)	$6 \times s$	$17176 \times s + 104 \times p + 88 \times \sum_{i=1}^{f} hd_i$	Cycle of 15 seconds
ARP processing(h)	$h + 2$	$(h-1) \times (ARP + 90) + (262 + 3 \times ARP)$	When creating a new flow
Flow installation ($hd_i, data_i$)	$3 \times hd_i + 2$	$(470 + 2 \times data) \times hd_i$	When creating a new flow
Flow expiration (hd_i)	$4 \times hd_i$	$440 \times hd_i$	When removing a flow

reply while sending a configuration to the switch. In addition, the ONOS installs default rules, which allow a switch to send the ARP, IPv6, Link Layer Discovery Protocol (LLDP), and Broadcast Domain Discovery Protocol (BDDP) messages to the controller in the switch. In this phase, the controller exchanges 21 messages, totaling traffic of $3012 \times s + 48 \times p$ bytes to complete the connection.

Link discovery: To manage and control the network, the controller should acquire the network topology. The ONOS uses the LLDP and BDDP to find the topology and broadcasting area. The ONOS sends *packet_out* messages containing an LLDP packet to the switch for the number of physical ports attached to the switch, and performs the same process for the BDDP packet. A switch that receives the LLDP or BDDP messages from a neighboring switch sends a *packet_in* message with the LLDP or BDDP packet to the controller. In other words, the controller sends the packets of the number of all ports attached to all switches and receives the returned packets, except those to go to hosts. Therefore, a total of $2 \times (2p - h)$ messages with $676 \times p - 334 \times h$ bytes are exchanged between the controller and networks.

Statistics: In SDN, the controller always monitors the network state to appropriately manage and respond to changes. For network monitoring, the controller periodically requests statistics information by sending statistic request messages and receives switch reply messages. This information is used for flow monitoring, load balancing, and failover. The ONOS periodically requests and provides three statistical information types: port, table, and flow. In general, the ONOS fetches statistics information on all tables, flows, and ports in a switch. In response to the request messages, the switch sends reply messages to the controller. Although the number of messages for statistics is 6 per switch, the traffic volume for statistic reply messages varies, being dependent on the statistic types and network state, e.g., the number of ports and flows. However, in a general case, the ONOS and switches exchange $16,412 \times s$ bytes for table statistics on all tables in a switch (in this case, 255 tables), $268 \times s + 104 \times p$ bytes for port statistics, and $496 \times s + 88 \times \sum_{i=1}^{f} hd_i$ bytes for flow statistics on the installed flows in the entire network.

Liveness: It is essential for the controller to maintain connection with the switches in order to control and manage the network. Periodically checking the link liveness between the controller and switches by sending echo messages enables the controller to check connection liveness. The ONOS checks a switch's liveness by receiving regular responses to the statistics request; thus, no extra control traffic is generated for liveness checking. However, as illustrated in Section III-B, some controllers use echo messages to check switch liveness.

*2) **Flow control**:* We categorize ONOS flow control functions into three functions: ARP processing, flow installation, and flow expiration.

ARP processing: Before flow setup, the sending host broadcasts ARP request messages to obtain the receiving host's MAC address. The ONOS uses the *proxyARP* application to process ARP messages. When the switch triggers a *packet_in* message with an ARP request packet, *proxyARP* broadcasts the ARP messages only to the edge switches. Then, the edge switches deliver ARP request packet to all hosts attached to them. When a receiving host sends the ARP reply message, the *proxyARP* directly relays it to the switch attached to the sending host. A total of $h + 2$ packets and $(h-1) \times (90 + ARP) + (262 + 3 \times ARP)$ bytes are used to process one ARP request.

Flow installation: This involves traffic for forwarding physical network flows via rule addition. For flow rule installation, we consider the reactive approach only. When a new flow packet arrives at an edge switch that does not have the rule, the switch sends a *packet_in* message to the controller to request a rule about how to process that packet. Then, the controller calculates the flow path and sends *flow_mod* messages to all switches on the path with a *barrier_request* message to ascertain if the operation is completed. The controller also sends a *packet_out* message to the edge switch attached to a destination host in response to the *packet_in* message. When the switch completes the rule installation, it sends a *barrier_reply* message to the controller. Thus, the controller exchanges $hd_i \times 3 + 2$ packets and $(470 + 2 \times data_i) \times hd_i$ bytes with the data plane for flow installation.

Flow expiration: A controller should continuously manage network flows, because the network states change continuously. Flow expiration processing can be controller- or switch-driven. When the expiration is performed by the switch, the controller presets the idle and hard timeout values for each flow rule. When the timeout values are fired, the switch alerts the controller about the event. However, ONOS uses the controller-driven approach, constituting regular management of the installed rules. The ONOS centrally manages the rules' lifecycles with flow statistics messages instead of timeout. If a rule expires, the controller sends a *flow_mod* message to the switch with a delete command and a *barrier_request* message. Then, the switch deletes that rule and sends a *flow_removed* message with a *barrier_reply* message. Here, four packets are exchanged to delete a rule, exchanging 440 bytes between the controller and a switch.

TABLE III: Analysis of the management functions of well-known controllers

Controller	ONOS	Floodlight	POX
Connection	* Configure switches * Install basic rules	* Configure switches	* Configure switches
Link discovery[1]	LLDP (81), BDDP (81) / Cycle : 3s	LLDP (75), BDDP (43) / Cycle : 15s	LLDP (41) / Cycle : 5s
Statistics	Table, port and flow / Cycle : 5s	X	Port and flow / Cycle : 5s
Liveness	X	Use echo messages / Cycle : 2s	Use echo messages / Cycle : 20s
ARP processing	*proxyARP*	* Flood request messages and deliver * Flow rule installation for reply message	* Flood request messages and deliver * No flow rule installation for reply message
Flow installation	End-to-end flow rule installation	End-to-end flow rule installation	Hop-by-hop flow rule installation
Flow expiration	Controller-driven	Switch-driven	Switch-driven

[1]The value with each protocol is the packet size.

*3) **Control traffic model**:* Table II presents a summary of the mathematical models representing the abovementioned functions with parameters. Clearly, each function's control channel usage can be derived using topology parameters, e.g., the number of switches, hosts, ports, and hop distance.

B. Analysis of heterogenous controllers

We also analyze the functions of two widely used controllers, Floodlight and POX, to identify common management functions and verify applicability of our model to other controllers. Overall, the mechanism and messages for each function differ for different controllers. However, most control operations can be grouped and characterized within our categorization (Section III-A).

Table III summarizes the analysis for the three controllers. For liveness and flow expiration, Floodlight and POX use the similar schemes (e.g., echo messages for liveness and switch-driven checking for flow expiration). However, regarding link discovery, Floodlight use LLDP and BDDP, but POX employs only BDDP. (Note that both controllers use shorter packets than ONOS.) In addition, for statistics, Floodlight does not request any statistics by default while POX requests port and flow statistics. For ARP request, Floodlight and POX send a *pakcet_out* message on all ports of a switch which delivered the ARP request (by a *packet_in* message) to the controller. Then, neighbor switches, which receive the ARP request, also send *packet_in* messages to the controller. By performing the same process on each switch, the ARP request reaches all hosts. On the other hand, for ARP reply messages, Floodlight installs flow rules for the replies on the path, but POX installs a single flow rule whenever it receives a *packet_in* message rather than installing whole flow rules for a path upon the first *packet_in* message.

We omit the detailed description for each controller due to the space limit, however we build control traffic prediction models for Floodlight and POX in the same manner as ONOS.

IV. Evaluation and Analysis

In this section, we discuss our evaluation and results to prove the model feasibility. We compare the values estimated from our models with real control traffic information running ONOS. We also use control traffic models for Floodlight and POX to evaluate our model accuracy for these controllers as well. To validate the model accuracy, we set six types of topology: 4-, 6-, 8-ary fat-tree topologies, and 8, 16, 32 hosts-per-pop ISP topologies with four pops (Fig. 2). The detailed parameters of each topology are listed in Table IV; the average hop distance value is used as the hop distance to predict a volume of traffic in our model. The characteristics are considered for the parameters of our model. We use Mininet for physical network emulation to collect real control traffic. In each topology, each host sends a ping message to a random host in the network every 0.5 s for 10 min.

TABLE IV: Topology parameters

(a) Fat-tree topology

Ary	Switches	Ports	Total hosts	Hop distance
4	20	80	16	1, 3, 5
6	45	252	54	
8	80	576	128	

(b) 4-pop ISP topology

Hosts per pop	Switches	Ports	Total hosts	Hop distance
8	32	160	32	1, 3, 8
16		192	64	
32		256	128	

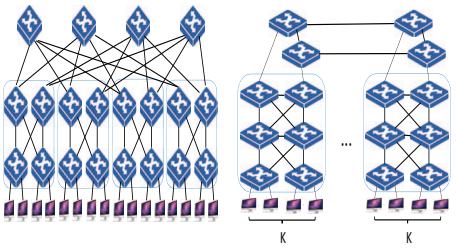
A. Model accuracy

In this study, we conduct experiments to verify that our model predicts the control channel traffic volume appropriately, as described in Section III-B. Fig. 3 shows the bandwidth of the traffic measured in our experiments and the values predicted using our model in the fat-tree and ISP topologies. The total predicted bandwidth is similar to the measured bandwidth for all topologies (94.19%). Further, all bandwidths in the fat-tree topology case increase with an increase in the number of switches and ports, whereas all bandwidths in the ISP topology case do not increase regardless of the number of hosts. On average, our model has 92.25% and 96.12% accuracy for ISP and fat-tree topologies, respectively. The difference between the estimated and actual values results from the flow rule requests of temporary packets, such as IPv6 neighbor discovery, Bootstrap, and DNS.

We also measure our control traffic model's accuracy when applied to Floodlight and POX with ISP topologies. Fig. 4 shows the average accuracy for each controller model that shows 94%. Our model has high accuracy for all three controllers. The prediction error occurs because we use the average hop distance in the model to predict the traffic and generate traffic between random pairs of hosts in experiments.

B. Comparison of controller channel usages between controllers

We compare the control channel usage between the ONOS, Floodlight, and POX. Table V lists the control channel band-



(a) Fat-tree topology (b) ISP topology

Fig. 2: Topologies

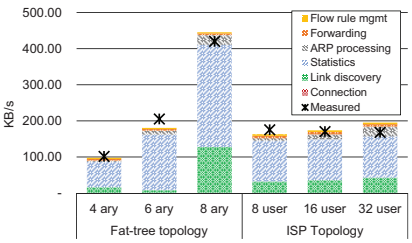


Fig. 3: Control traffic estimation with ONOS controller

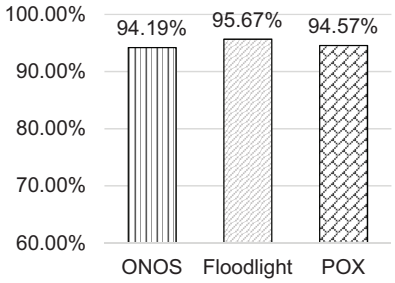
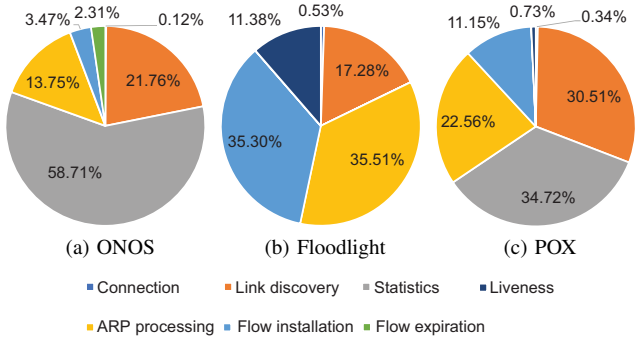


Fig. 4: Average accuracy of our model

TABLE V: Control channel usage of three controllers

	ONOS (KB/s)	Floodlight (KB/s)	POX (KB/s)
Connection	0.24	0.11	0.11
Link discovery	42.42	3.51	9.73
Statistics	114.43	-	11.07
Liveness	-	2.31	0.23
ARP processing	26.80	7.21	7.19
Flow installation	6.76	7.17	3.55
Flow expiration	4.50	-	-
Overall	**195.15**	**20.31**	**31.88**



(a) ONOS (b) Floodlight (c) POX

Fig. 5: Management functions share of three controllers

widths predicted by our models when the physical network hosts handle the same load for ISP topology with 32 users-per-pop. ONOS generates ten and six times more control traffic than Floodlight and POX, respectively. Note that ONOS uses more traffic for link discovery, statistics, and ARP processing, because ONOS requests statistic for all tables, ports, and flows, and employs larger-size LLDP and BDDP packets. ONOS also broadcasts BDDP packets to all ports of each switch, whereas the Floodlight controller sends them only to the edge ports and the POX controller does not use BDDP packets at all. Moreover, for ARP processing, ONOS broadcasts the ARP requests to all edge switches with *packet_out* messages (*proxyARP*). However, other controllers simply flood them; thus, the traffic for processing is relatively low.

Fig. 5 shows the control traffic share predicted by our model for management functions of the three controllers. Each controller has a completely different control traffic usage distribution. ONOS and POX, having the statistics functions, use 58.64% and 34.98% of the control traffic for these functions, respectively. In addition, while ONOS and POX use about 35% of the control traffic for forwarding, Floodlight uses about 80% of the control traffic for the same. For all three controllers, 20-30% of the control traffic is used for link discovery.

V. CONCLUSION

Because SDN management is conducted by controllers through the control channel, analysis of control channel scalability is crucial. In our study, we analyze the management functions of the ONOS, Floodlight, and POX controllers, and present a control traffic model to evaluate the controller scalability. We verify the model accuracy with six different topologies and the abovementioned three controllers. On average, our model exhibits 94.8% prediction accuracy.

The results demonstrate that our model can predict the amount of control traffic as well as the ratio of control traffic used to perform each of controller functions. We believe that our model can be useful for the provisioning of controller and switches in designing SDN network. In the future, we plan to extend our model to other controllers.

REFERENCES

[1] I. F. Akyildiz, S. Nie, S.-C. Lin, and M. Chandrasekaran, "5g roadmap: 10 key enabling technologies," *Computer Networks*, vol. 106, pp. 17–48, 2016.

[2] T. Koponen, K. Amidon, P. Balland, M. Casado, A. Chanda, B. Fulton, I. Ganichev, J. Gross, P. Ingram, E. J. Jackson *et al.*, "Network virtualization in multi-tenant datacenters." in *NSDI*, vol. 14, 2014, pp. 203–216.

[3] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[4] A. Bianco, P. Giaccone, A. Mahmood, M. Ullio, and V. Vercellone, "Evaluating the sdn control traffic in large isp networks," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5248–5253.

[5] A. Bianco, P. Giaccone, R. Mashayekhi, M. Ullio, and V. Vercellone, "Scalability of onos reactive forwarding applications in isp networks," *Computer Communications*, vol. 102, pp. 130–138, 2017.

[6] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[7] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*. IEEE, 2014, pp. 1–6.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[9] J. Halpern, "Forwarding and control element separation (forces) forwarding element model," 2010.

[10] R. Enns, M. Bjorklund, and J. Schoenwaelder, "Network configuration protocol (netconf)," *Network*, 2011.