INITIALIZATION TIME OPTIMIZATION OF A WIRELESS TRANSPORT EMULATOR USED FOR SDN APPLICATIONS DEVELOPMENT

Alexandru Stancu[1], Alexandru Vulpe[2], Simona Halunga[3]

Software-Defined Networking (SDN) is a novel network architecture that emerged in order to mitigate the limitations proven by traditional networks. It is not yet a mature solution, some standardization activities being still in progress. Having the appropriate tools for helping this process is an important advantage. The Wireless Transport Emulator (WTE) is a software tool that offers the possibility of simulating wireless transport network elements and links between them, while exposing some information models proposed by the Open Networking Foundation (ONF), like TR-532 and TR-512. This paper presents an evaluation of the time needed for the simulator initialization and proposes a solution for optimizing this time.

Keywords: Software-Defined Networking, Open Networking Foundation, Wireless Transport Networks

1. Introduction

Software-Defined Networking (SDN) is a paradigm that emerged around the year 2009, from the activity conducted in the OpenFlow project in the Stanford University [1]. It appeared as a solution for mitigating the limitations demonstrated by traditional networks, such as: complex management, slow network innovation, vendor dependency or network policies inconsistencies.

SDN proposes the decoupling of the data and control planes, centralizing the network control in a logical entity represented by the SDN controller [2]. With this approach, network elements become simple forwarding elements that are instructed by the controller about where to forward the packets in the network. This enables network programmability, through software applications that can run on top of the SDN controller, having a network-wide view [3].

[1]PhD Student, Faculty of Electronics, Telecommunications and Information Technology, University "Politehnica" of Bucharest, Romania, e-mail: alex.stancu@radio.pub.ro

[2]Lecturer, Faculty of Electronics, Telecommunications and Information Technology, University "Politehnica" of Bucharest, Romania

[3]Professor, Faculty of Electronics, Telecommunications and Information Technology, University "Politehnica" of Bucharest, Romania

SDN aims to be applicable in all types of networks, from campus [4], or data-center networks [5], to optical [6], wireless transport [7] or, more recently, vehicular networks [8]. In some cases, production deployments already exist, but in other cases standardization activities are still ongoing.

The Open Networking Foundation (ONF) is an organization that promotes the adoption of SDN through the development of open standards and open-source software ecosystems. The Wireless Transport (WT) project, which is part of this organization, focuses on enabling SDN in WT networks. The group created an information model to be used for managing wireless transport devices: the Microwave Information Model (TR-532) [9]. Several Proofs of Concept (PoCs) were conducted in order to validate the proposed model [10, 11, 12]. It is transformed in the Yet Another Next Generation (YANG) [13] modeling language, for providing the ability to be used by the Network Configuration Protocol (NETCONF) [14]. This approach implies that the wireless transport devices would need a NETCONF interface for communicating with the SDN controller.

For providing an easy and flexible approach for testing the Microwave Information Model, a simulator was developed by the main author of this paper: the Wireless Transport Emulator (WTE). This enables users to simulate wireless transport devices, while exposing the TR-532 information model and to emulate links between such equipment. Its functionality is best suited for SDN applications developers that want to use TR-532, which can simulate wireless transport networks, thus eliminating the need of owning real, expensive devices. It can be used also by network operators for testing SDN applications that are based on TR-532 and their interactions in an emulated environment, without needing to alter the production network for testing.

This paper is organized as follows: section 2 presents a high-level overview of the WTE, section 3 illustrates the initialization time of the simulator before and after the optimization, while describing the improvements that were implemented to achieve faster initialization time, and section 4 concludes the paper.

2. **Wireless Transport Emulator Overview**

The purpose of the Wireless Transport Emulator is to enable emulation of wireless transport devices and links between them, while exposing the information models proposed by ONF: TR-532, the Microwave Information Model, and TR-512, the Core Information Model [15]. The simulation is done on a single Linux host. A high-level overview of a network element simulated with WTE is illustrated in Figure 1.

WTE uses a JavaScript Object Notation (JSON) file that describes the topology to be simulated. Each Network Element (NE) is represented as a docker container, inside which a Linux Operating System (OS) runs, along

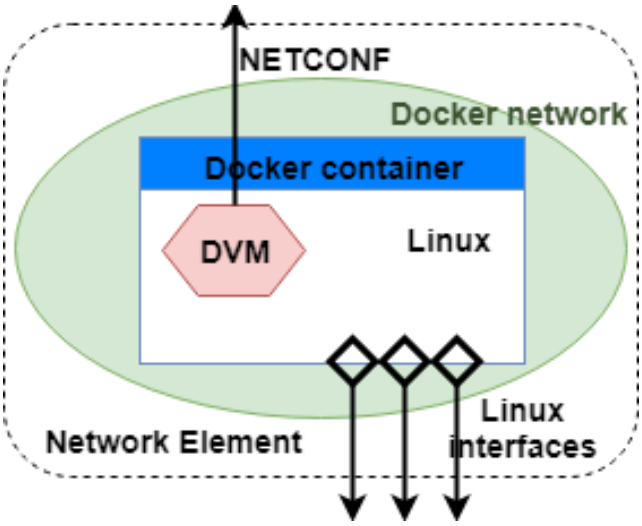

Fig. 1. High-level overview of an NE simulated with WTE.

with a NETCONF server implementation. The latter is responsible for providing the NETCONF interface to SDN controllers and for advertising the aforementioned information models. This implementation is called Default Values Mediator (DVM) and it is based on the OpenYuma framework, its architecture being described in [16].

The interfaces of each simulated NE, that reside on different transport layers, as described in the information models, are represented in the simulator as Linux interfaces inside each docker container. The links are represented using virtual Ethernet pairs (*veth* pairs), which are actually tunnels inside the Linux host OS. The characteristics, such as the bandwidth, delay or packet loss of the interfaces that form the link are altered according to device attributes, like channel bandwidth, or modulation used, in order to emulate wireless links. Each docker container can be run inside a docker network, for achieving network isolation between the simulated NEs.

The initialization of the WTE uses the following workflow: the file containing the topology to be simulated is analyzed, then the docker networks and docker containers associated with each NE are created. The following step is adding the needed interfaces for each simulated device in the corresponding Linux OS from the docker container, and then creating the connections between the interfaces that define links between NEs.

3. Initialization time optimization

The initialization time of the WTE represents the duration, in seconds, between the moment when the user issues the start command and the moment when the simulator finishes representing all the elements that form the topology and waits for commands in its Command Line Interface (CLI).

The initialization time is given by the duration of executing the commands of creating the docker networks, containers, adding the necessary Linux interfaces inside each of them and creating the links between the containers. This time influences only the user experience, because it is not relevant anymore after WTE is ready and the network is being simulated. The docker engine does not permit the parallelization of commands like network or container creation, so it is not possible to optimize the initialization time by running those commands in parallel. A sequence diagram representing the initialization of the WTE can be seen in Figure 2. As illustrated there, the optimization could be implemented in the loop that creates the Linux interfaces in the docker containers.

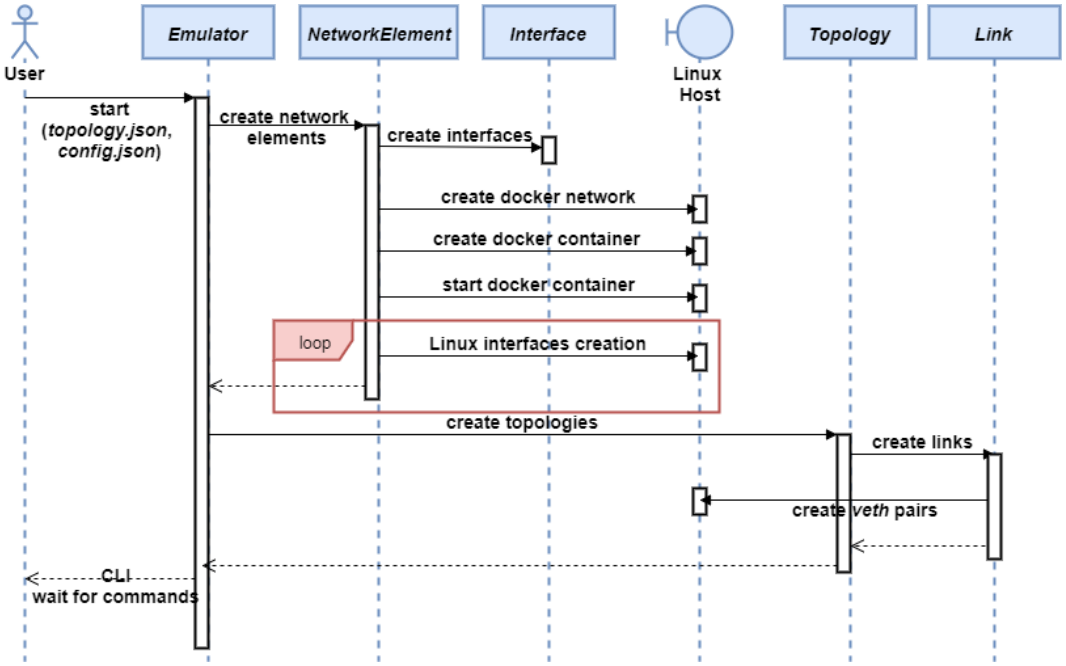

Fig. 2. Sequence diagram for the initialization phase of the WTE.

Optimizing the initialization time of the WTE was done with regards to the creation of Linux interfaces inside each docker container. The previous implementation behaved in the following manner: the interfaces from the topology file were iterated, and for each interface defined there, a command was sent to the associated container in order to create that specific object inside the Linux OS. This approach introduced a lot of overhead represented by sending multiple very small commands. The optimization introduced in this article consists in changing this approach. Instead of sending each command to the docker container, it is written inside a script file associated with the simulated NE. For each of the corresponding interfaces of that specific device, the commands that would create the Linux objects were written in that specific file. After iterating over all the interfaces of a device, the script file would

contain all the commands necessary for adding the Linux objects inside the container. This script file was executed and all the interfaces were added as a batch. The execution time associated with these operations decreases significantly, compared to the previous approach. An illustration of the relevant part of the sequence diagram of the initialization, before and after the optimization was introduced can be seen in Figure 3.

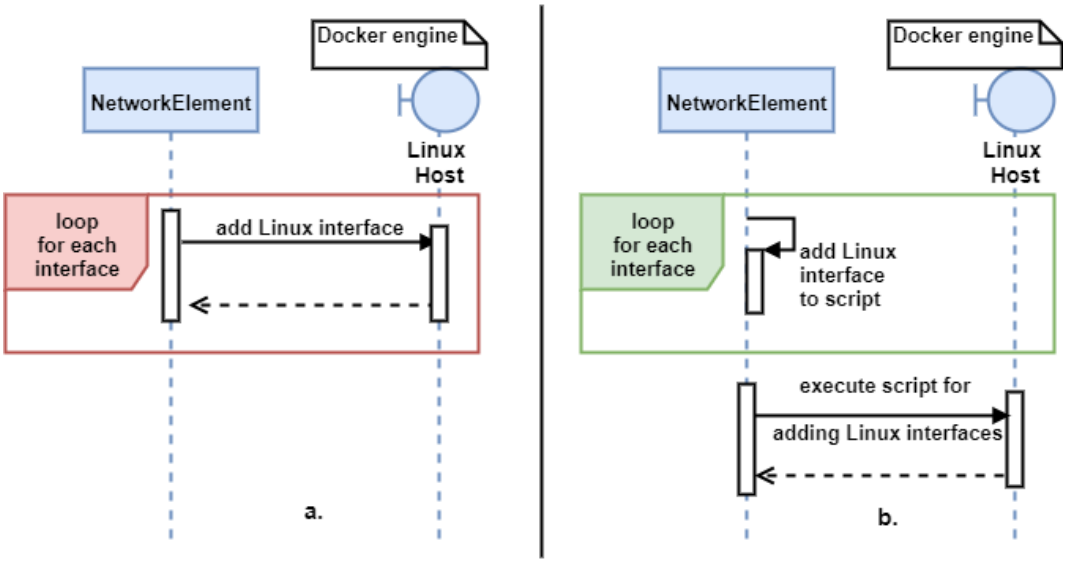

FIG. 3. Relevant part of the sequence diagram of the initialization phase of the WTE: a) before optimization, b) after the optimization.

For measuring the benefits introduced by this optimization, the following method was used: the simulator environment was installed in two cloud environments: Orbit Cloud, which represents a wireless network testbed, as described in [17], and in a cloud provided by Deutsche Telekom (DT) for the 4th WT PoC [12]. The measurements were conducted on a two types of topologies: ring and full mesh, while varying the number of simulated devices.

In the case of the ring topology, the number of simulated devices varied from 10 to 200, with a pace of 10. This implies also varying the number of simulated interfaces, because, in this case, each device contains a number of 8 interfaces, according to the information models proposed by ONF. This implies a variation of the interfaces between 80 and 1600.

For the full mesh topology, the number of devices varied between 3 and 10, with a pace of 1. In this type of topology, the number of interfaces does not depend in a linear manner anymore on the number of NEs, but a quadratic dependency exists. This means that the number of interfaces varied between 18 and 270.

The values for the initialization time were collected both before and after implementing the optimization described previously. The results of the measurements are illustrated in Figures 4, 5, 6 and 7.

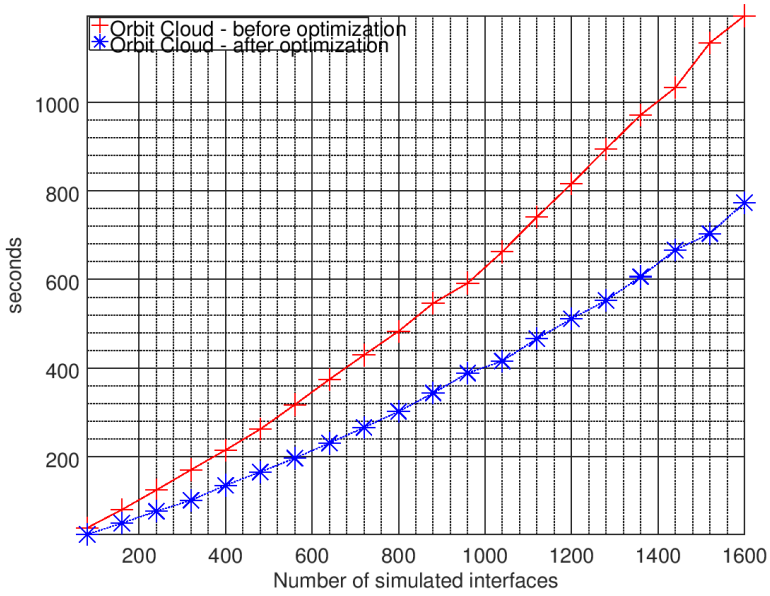

FIG. 4. Initialization times in the Orbit Cloud environment in a **ring** topology.

It was chosen to use two different simulation environments for verifying that the improvements given by the optimization are applicable in a general manner and do not depend on the system used for the simulation. The measured results differ slightly between the two systems, Orbit and DT, with regards to the absolute values. This is caused by the processing power of the two systems, which is different for those cloud environments. This difference is not applicable with regards to relative values. We observe an approximately linear dependency between the number of simulated interfaces and the number of seconds necessary for the initialization of the WTE in both cases.

The charts reveal fairly high values for the initialization time, before the optimization was implemented. This time could reach values around 400-450 seconds for ring topologies containing approximately 800 simulated interfaces, and reach values higher than 1000 seconds for big topologies, containing around 1600 simulated interfaces. In the case of full mesh topologies, it can be seen that the improvement is even bigger, because for this topology type, for the same number of simulated interfaces, there are less NEs, hence the time needed for starting the associated docker networks and containers is smaller.

After implementing the previously described optimization, the values for the initialization time decrease. The average improvement, regardless of the

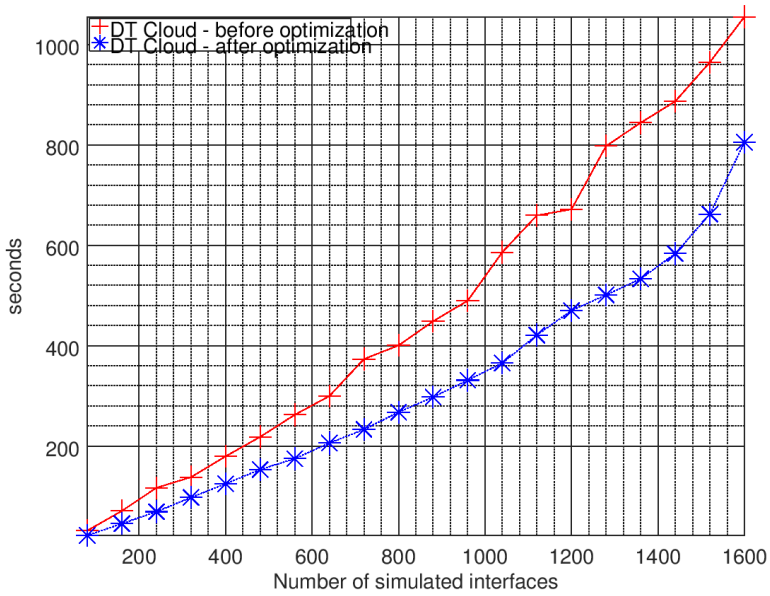

FIG. 5. Initialization times in the DT Cloud environment in a **ring** topology.

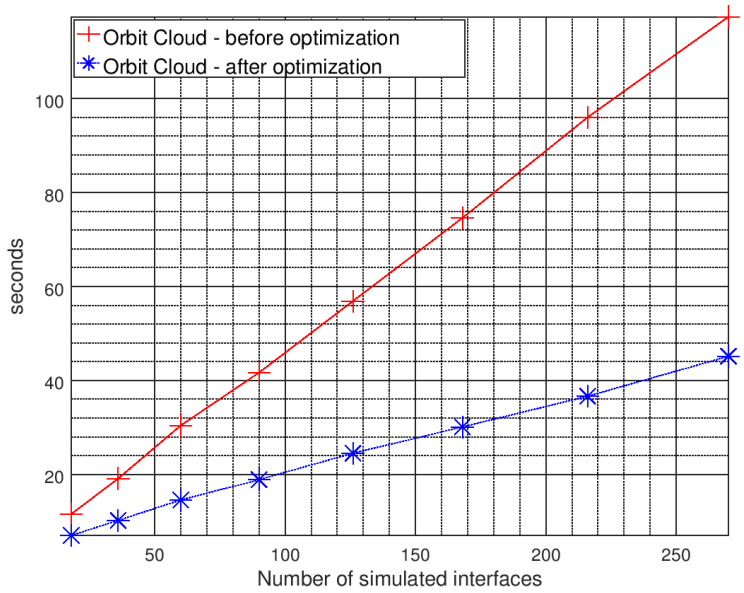

FIG. 6. Initialization times in the Orbit Cloud environment in a **mesh** topology.

system used for running the evaluation or the number of simulated interfaces in the topology, is around 37%, in the case of ring topologies. For the full mesh

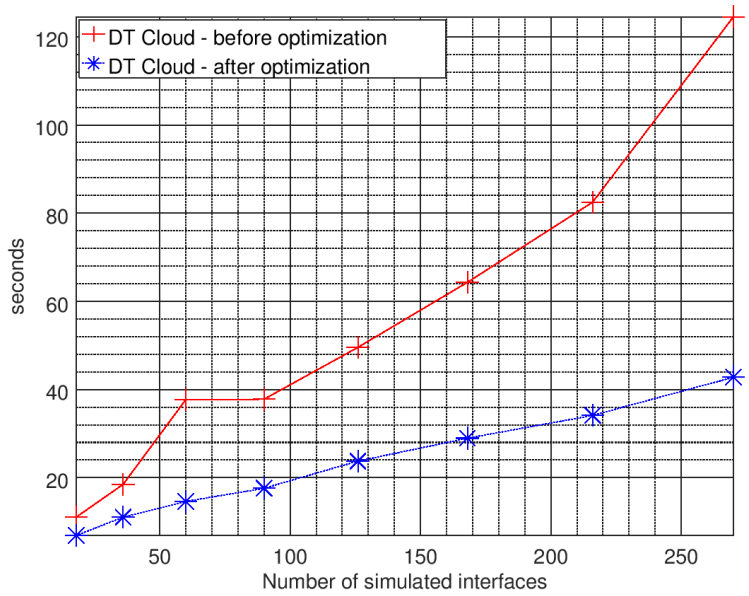

FIG. 7. Initialization times in the DT Cloud environment in a **mesh** topology.

topologies, given the fact that the number of simulated interfaces depends in a quadratic manner on the number of devices, the improvement reaches almost 47%. Reducing the initialization time with such a high percentage greatly improves user experience, even though this is irrelevant after the simulator is started.

4. Conclusions

Software-Defined Networking is a paradigm with a high momentum given by the research and standardization activities that are revolving around it. It is a key enabler for novel technologies, like the Internet of Things (IoT), vehicular networks or 5G.

Having the right tools to support the research and standardization activities is an important advantage. Mininet is a great example in this direction, enabling the simulation of software-defined networks that use the OpenFlow protocol as a southbound interface. The Wireless Transport Emulator is a complementary tool that can be used for simulating WT networks, while using a NETCONF southbound interface that exposes the newly emerged information models proposed by ONF: TR-532, the Microwave Information Model and TR-512, the Core Information Model.

The WTE can be extended to incorporate other information models as well, through its modular and flexible architecture, with the observation that

these models need to be translated into the YANG modeling language beforehand.

In this paper, an optimization for the initialization process of the WTE was implemented. In a nutshell, instead of adding the relevant Linux interfaces inside the docker containers representing NEs one at a time, the improvement was using a batch approach, adding all interfaces at once in every simulated device. This method eliminated the overhead given by the duration of sending the commands to the docker containers. Instead of triggering multiple short instructions, only one batch command is given to each simulated device, this approach being, as shown in the results, faster by approximately 37%, in the case of ring topologies, and up to 47% in the case of full mesh topologies.

Further research directions could be represented by investigations in the newer versions of the docker engine about enabling parallelization when creating docker networks or containers. These would bring a great advantage from the initialization time of the WTE point of view. Also, a comparison for this characteristic, between *mininet* and WTE could be interesting.

REFERENCES

[1] *N. Feamster, J. Rexford and E. Zegura*, The Road to SDN: An intellectual history of programmable networks, ACM Queue, **XI**(2013), No. 12.

[2] *D. Kreutz, F. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky and S. Uhlig*, Software-defined networking: A comprehensive survey, Proceedings of the IEEE, **103**(2015), No. 1, 14-76.

[3] *A. Stancu, S. Halunga, G. Suciu, and A. Vulpe*, An Overview Study of Software Defined Networking, Proceedings of the IE 2015 International Conference, (2015), 50-55.

[4] *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, **38**(2008), No. 2, 69-74.

[5] *D. Li, Y. Shang, and C. Chen*, Software defined green data center network with exclusive routing, INFOCOM, 2014 Proceedings IEEE, (2014), 1743-1751.

[6] *D. Simeonidou, R. Nejabati, and M.P. Channegowda*, Software defined optical networks technology and infrastructure: Enabling software-defined optical network operations, Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), (2013), 1-3.

[7] *C.J. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L.M. Contreras, H. Jin, and J.C. Zúñiga*, An architecture for software defined wireless networking,IEEE wireless communications, **21**(2014), No. 3, 52-61.

[8] *Y. Cao, J. Guo and Y. Wu*, SDN enabled content distribution in vehicular networks, Fourth International Conference on Innovative Computing Technology (INTECH), (2014), 164-169.

[9] *Open Networking Foundation*, TR-532: Microwave Information Model, December 2016, [Online], `https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/05/TR-532-Microwave-Information-Model-V1.pdf`.

[10] *Open Networking Foundation*, Wireless Transport SDN Proof of Concept 2 Detailed Report, June 2016, [Online], `https://rs.opennetworking.org/wiki/download/`

attachments/262144003/2nd_Wireless%20Transport_PoC_White_Paper.pdf?api=v2.

[11] *Open Networking Foundation*, Third Wireless Transport SDN Proof of Concept White Paper, December 2016, [Online], https://rs.opennetworking.org/wiki/download/attachments/262144003/3rd_Wireless%20Transport_PoC_White_Paper.pdf?api=v2.

[12] *Open Networking Foundation*, Fourth Wireless Transport SDN Proof of Concept White Paper, August 2017, [Online], https://rs.opennetworking.org/wiki/download/attachments/262144003/4th_Wireless_Transport_PoC_White%20Paper.docx?api=v2.

[13] *Internet Engineering Task Force*, RFC 6020: YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), October 2010, [Online], https://tools.ietf.org/html/rfc6020.

[14] *Internet Engineering Task Force*, RFC 6241: Network Configuration Protocol (NETCONF), June 2011, [Online], https://tools.ietf.org/html/rfc6241.

[15] *Open Networking Foundation*, TR-512: Core Information Model, November 2015, [Online], https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/ONF-CIM_Core_Model_base_document_1.1.pdf.

[16] *A. Stancu, A. Avram, M. Skorupski, A. Vulpe, and S. Halunga*, Enabling SDN application development using a NETCONF mediator layer simulator, Ninth International Conference on Ubiquitous and Future Networks (ICUFN), (2017), 658-663.

[17] *D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh*, Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols, Wireless Communications and Networking Conference, **3**(2005), 1664-1669.