OpenFlow Deployment and Concept Analysis

Tomas HEGR, Leos BOHAC, Vojtech UHLIR, Petr CHLUMSKY

Department of Telecommunication Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Technicka 2, 166 27 Prague, Czech Republic

tomas.hegr@fel.cvut.cz, bohac@fel.cvut.cz, uhlirvoj@fel.cvut.cz, petr.chlumsky@fel.cvut.cz

Abstract. *Terms such as SDN and OpenFlow (OF) are often used in the research and development of data networks. This paper deals with the analysis of the current state of OpenFlow protocol deployment options as it is the only real representative protocol that enables the implementation of Software Defined Networking outside an academic world. There is introduced an insight into the current state of the OpenFlow specification development at various levels is introduced. The possible limitations associated with this concept in conjunction with the latest version (1.3) of the specification published by ONF are also presented. In the conclusion there presented a demonstrative security application addressing the lack of IPv6 support in real network devices since most of today's switches and controllers support only OF v1.0.*

Keywords

Analysis, deployment, IPv6, OpenFlow, SDN, security.

1. Introduction

Software-Defined Networking (SDN) has recently become one of the most progressive sectors in terms of research and development in data networks. SDN concept comes up with the idea of breaking away the ties from a particular hardware platform and moving to the abstract model as it is done also in other sectors of information technologies. This shifts transmission and computing capabilities significantly to a higher level.

The original idea of SDN described in [1] was born at Stanford University around 2005. The SDN concept brings the separation of network device features to the control plane, and the data plane. While the control plane is programmatically accessible through well-defined API (Application Programming Interface), data plane ensures a data processing according to the rules uploaded to the device.

The key representative of the protocols enabling the creation and operation of the SDN is the OpenFlow (OF) protocol nowadays. It is also referred to as southbound protocol and it simply enables the transfer of control instructions driving individual data flows in network devices. More specifically it fills up their TCAM switching tables. Other potential future representatives of southbound protocols are XMPP (Extensible Messaging and Presence Protocol) or extended BGP (Border Gateway Protocol).

OpenFlow has been developed since 2007, and the first protocol specification was approved in 2009 [2]. Lately the development was adopted by the Open Networking Foundation (ONF) consortium. The protocol defines the structure of control messages and it describes the way how messages are exchanged. OF is based on the centralized approach with a controller as a main driving element. This controller runs a software platform with the API enabling the direct control of data flows in a network.

Although OF enables to control the data plane, it does not provide any management capabilities for individual network devices. For this reason ONF has published OpenFlow Management and Configuration Protocol shortly called OpenFlow Config [3]. It was designed to support all OF implementations and also the management of both physical and virtual switches. The protocol part running in the switch requires the support of NETCONF protocol that uses an XML-like configuration [4].

Among the presented SDN use cases there are these that enable dynamic creation of the topologically separated networks (scalability beyond VLAN), service management up to the application layer, intelligent load-balancing and even more [5]. It is obvious that the primary application is expected in data centers, mainly in conjunction with the virtualization of services to IaaS (Infrastructure as a Service).

Since the concept of the OF technology is centralized, its practical deployment may be difficult. We have identified and analyzed potential constraints and

shortcomings. This is an important step in a possible use of OF outside the academic test beds in a real production environment.

The real deployment, dealing with shortcomings and security issues such as the attack against IPv6 Duplication Address Detection (DAD) was the reason for the creation of a demonstration admin tool built on currently the most widely spread OF version 1.0. It shows how to cope with constraints of the early OF versions.

2. OpenFlow Analysis

OpenFlow should not be considered only as a protocol, but as a concept built on the fundamental idea separating the control and the data plane. There is defined an unambiguous role of the central controller, which has a number of advantages and disadvantages examined below.

The specification proposes description of the control messages for filling Forwarding Information Base (FIB) realized as so-called FlowTables and at the hardware level utilizing either RAM or TCAM (Ternary Content-Addressable Memory). OF control messages are sent over TLS/SSL secured TCP connection.

Furthermore, the OF specification includes the matching fields definition called tuples. On their basis there are made decisions and taken actions (instructions) and counted statistics. Tuples in OF 1.0 are defined for header fields from the first to the fourth layer if we consider port number as physical layer. There are 12 of them. A switch component model summarizing all OF parts is shown in Fig. 1. The main protocol changes between versions will be discussed below.

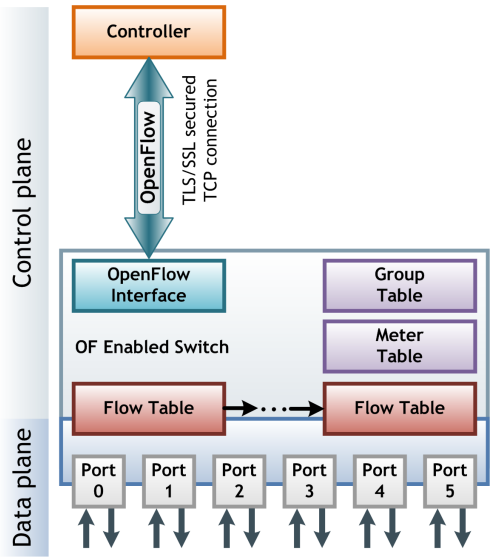

Fig. 1: Component model of an OpenFlow switch.

2.1. Concept Analysis

Regardless of the performance aspects it is necessary to draw attention to the potential risks and limitations associated with the concept of central control. It brings many benefits as collecting of statistical information and network devices status. The realization of the controller is independent of the third party applications creating the network control logic. On the other hand, the central control always raises issues relating primarily to the network reliability and its availability.

Controller as the software platform can be formed either central or distributed. Even though a distributed solution is apparently better in terms of the availability, it brings additional challenges such as consistency level throughout individual controller instances. Therefore we try to examine various restrictions associated with the OF networks control.

1) Hardware Failure

With the centralization of control functions in the controller there is a fundamental issue with the network reliability. While in networks with distributed protocols each switch always decides on the basis of its own view of the network, switches in OF networks are dependent on matching rules downloaded from the controller. In the case of connection failure the switch is not able to provide communication between end devices or it can behave like a standard switch. If we look at the test case in Fig. 2 we can get three failure probability formulas.

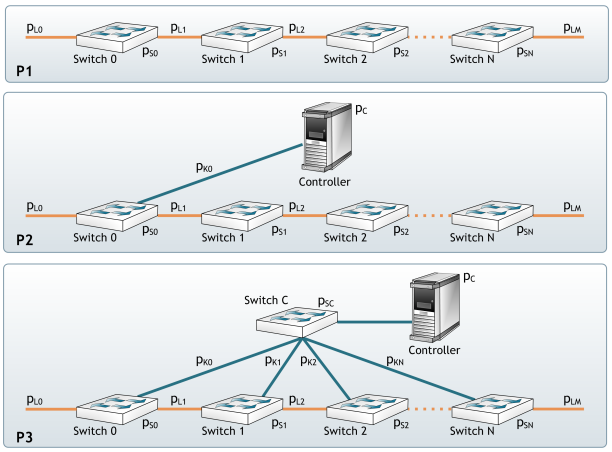

Fig. 2: Connection scheme of a classical non-OF network (P1) and an OF topology with one connection sharing payload means called the in-band control (P2) and full connection also called the out-band control (P3).

The first formula corresponds to the common topology of classical distributed control.

$$P_1 = \prod_{i=0}^{N} p_{Si} \cdot \prod_{j=0}^{M} p_{Lj}, \tag{1}$$

where p_{Si} is the probability that the switch i will be available and p_{Lj} is the probability that the link j will be available.

The second formula assumes in-band control channel e.g. VLAN with single physical connection to the first switch in the chain.

$$P_2 = p_{K0} \cdot p_C \cdot P_1, \tag{2}$$

where p_C is the probability that the controller will be available and p_{K0} is the probability that the link from the Switch 0 to the controller will be available.

The last scenario shows the full connected topology where every single switch has separated physical connection to the controller. The probability of availability can be described as in the third formula.

$$P_3 = \prod_{i=0}^{N} p_{Ki} \cdot p_{Ki} \cdot p_C \cdot p_{SC} \cdot P_1, \tag{3}$$

where p_{SC} is the probability that the concentration switch will be available and p_{Ki} is the probability that the link from switch i to the controller will be available.

The original version of OF 1.0 specifies only one active TCP connection to the controller at the time. Later version 1.2 provides the ability to connect to more controllers at the same time and determine their role, including Master/Slave. The Master controller selection is out of scope the OF specification and it is a controller issue.

Loss detection of the switch-controller connection is not based on TCP features but must be made on the application layer by exchanging OF ECHO messages. There is no specification of the timer value in the specification for exchanging this type of messages. This means that the platform freedom does not need to meet QoS requirements of communicating applications because of late controller reconnection.

A second solution of such situations is a distributed controller, which can partially overcome the loss of the switch-controller connection. Currently, there are solutions such as HyperFlow or Onix by Nicira Company lately adopted by VMware as NVP [6], [7], [8].

Onix provides a higher level of abstraction than traditional controllers using Network Information Base (NBI) and it maintains a consistent state over controllers utilizing Distributed Hash Table (DHT). Onix leaves the level of reliability requirements on the user application.

HyperFlow is an extension for NOX controller and provides synchronization of controllers by propagating events affecting the controller state. Implementation is based on a Distributed File System (DFS) WheelFS [9].

The previous mentioned solutions imply that to ensure higher controller availability it must be always taken into account the time needed to identify the connection loss, the time for network reconfiguration and controller synchronization speed. The inter-plane synchronization is provided by east/westbound protocols. In 2012 there started works at IETF on the recommendation for exchanging messages in SDN networks, which should cover horizontal communication between controllers [10].

2) Management Connectivity

For exchanging control messages between the OF controller and switches, the connectivity must be assured. It has two base types. The connectivity may be provided either on the same infrastructure, which operates payload flows, then it is so-called in-band or on the dedicated infrastructure which is called out-band.

When the in-band connectivity design is inappropriately utilized, there arises a danger of the control loss over the entire network in case of the single link failure. Therefore, it is important to think about appropriate diversification during designing of the controller location and its physical connection.

It is necessary to operate the management network on the basis of today's conventional technology using standard routing protocols such as IS-IS, OSPF, and more.

From this perspective, the switch still functions as a hybrid and that is the reason why there cannot be built the purely OF-based network. In addition, the separate infrastructure brings additional costs to build and operate any network. OF does not even reduce expenditures to the network administrator who has to take care of the initial implementation of the system setup and configuration.

3) Software Failures

Since the matching rules are generated by a third party high level automated application, there can occur certain limitations for a network administrator. The administrator can have lower ability to intervene in the process of matching rules creating, understanding their granularity or capture a problem because of their timeout validity.

The possibility of bugs in the third party applications in comparison with distributed control of conventional networks may increase. Traditional, by standards described, behavior of distributed algorithms has been tested for years. Also a platform API provided to the third party applications may have higher bug rate

than operating systems on network devices developed for decades due to the OF novelty.

It should be noted that unlike bugs occurring in the software of individual network devices, an incorrect central control may negatively affect the entire network. Nevertheless, compared to the traditional distributed approach OF definitely provides much higher degree of flexibility in eventual adjustment of these software bugs.

4) Security Vulnerabilities

As in the case of potential weaknesses in the form of hardware failure of the central controller is the central solution from the perspective of potential cyber-attack a great disadvantage. The secure channel between the switch and the controller is implemented to the transport layer using TLS, the standard protocol of which is derived the level of information security.

At the level of switch management the abilities of an attacker are basically the same as at today's traditional network devices. It depends on the attacker if he gets an access to the management network, consequently to a device CLI.

From the controller point of view, there primarily arises a threat of the third party application attack. This application generates matching rules and its vulnerability is vital for the whole network. This threat is also associated with the need of securing an operating system which runs an OF controller platform. It can be very difficult to detect such an attack or intrusion because of excluding the administrator from the rules generating process.

Neither the distribution controller nor the Master/Slave model provides the solution. Since that any of connected controllers can request Master role unidirectional, for the attacker it is enough to get control only over one instance of a controller. Then he just sets the Master parameter and he can act as a major decisive controller.

5) Scalability

At very large networks there comes the challenge of the sufficient performance on both switch and controller side. In extreme cases of huge flow numbers the controller should be able to manage hundreds of thousands of connections.

At the switch it is possible to move from realizing FlowTables using expensive TCAM that has a limited capacity to de facto software solution through the implementation in RAM. Typical count of entries in the FlowTables using TCAM is around 1500 entries [11]. It is therefore not appropriate to use OF in core switches. The only software implementation brings significant fall in processing time of the matching rules [12].

The scalability on the controller side can be solved in two ways. The first is a server-side scaling, thus utilizing multiple cores. There exist several platforms in this regard as NOX Destiny, Beacon or Maestro [13] [14]. The second approach uses additional controller instances for different areas as in the case with aforementioned HyperFlow or ONIX. Merging these approaches would cover both scalability and reliability issues.

2.2. Deployment Analysis

OF has been presented in the first usable version in 2009. Since then there is a high promoting activity of this specification in the incorporation into commercial equipment and deployment in real networks. The whole four years after its introduction the OF situation is getting clutter even because of rapid development under the ONF leadership.

We will discuss the current (June 2013) OF deployment status below, options in the production environment and the possibility of implementation of superstructure systems. Although the first OF version was perhaps academic matter there was an apparent target of transferring this technology to the commercial virtualization world. For testing reasons there were built several large academic networks such as GENI in the United States or Ofelia in Europe [15] [16].

1) OpenFlow 1.0 Features Development

In addition to the aforementioned limited number of tuples in OF v1.0 it does not include support of simultaneous connection to multiple controllers. This feature is supported up to the version 1.2. Moreover there is introduced replacement of actions by instructions and matching rules pipelining through more FlowTables.

Essential for the assumed application field, i.e. LAN in data centers, is the lack of IPv6 support. At the time of the OF first version development it was already obvious that it will be necessary to move to this type of addresses. The support came in version 1.2. Next was not a support for QoS. Solving QoS was left on the destination port queue. An aid in the form of the Meter Table comes in version 1.3.

2) Vendor Support

Even so, the protocol may be considered young; it is the only one southbound protocol suitable for implementing virtualized networks nowadays. Hence, it can

be assumed that the effort of vendors will be a rapid development of both switches and controllers. However, the situation is different. From the outside view, most vendors after fast implementation of OF v1.0 enabled devices became petrified and the market stagnated. The OF support was often only part of the marketing strategy.

At the beginning of 2013 there was a few controllers often based on open solutions, but some of them are already commercial. One of the first was closed ONIX partly of which came open NOX [17]. Next one is BigSwitch controller based on Floodlight project, which was derived from Beacon controller. Another example is the IBM Programmable Network Controller and PrograMableFlow controller from NEC, which is based on open Helios later melted to Trema project [18], [19].

None of the aforementioned controllers does support OF version higher than 1.0 according to the available sources in June 2013.

Open solutions are based on different licenses as GPL and more. This may not be suitable for production environments because there is no support or warranty. Additionally OF is not standardized; it is just a specification developed by companies and its sustainability is disputable.

There is a reasonable supposition that during the next few years there will be progressive development in both software platforms and network devices over all big network vendors.

3) Development and Simulations

It is important to keep in mind that the controller is only software platform not the application logic forming matching rules. Such applications are in their simple version a natural component of the controller, but they may be provided also as a third party product. This brings more demands on the reliability and the algorithms support. Since there is a responsibility transfer towards to algorithms, they have to be tested in details.

The testing can be divided into equipment conformance testing, control algorithms testing (network simulations) and performance testing such as throughput or latency.

One of the first conformity testing tools was OFtest, which is basically a controller comparing the received OF responses with the expected ones [20]. Currently, there are commercial solutions such as from Ixia [21]. ONF runs its own OpenFlow Conformance Testing Program.

For the development and testing of algorithms there was available Mininet at the first times of the OF release. Mininet creates a semi-virtual network topology built on OpenVSwitch [22]. It allows simulating various virtual network topologies. Its performance is dependable primarily on the test machine performance. Well-known NS-3 platform also supports OF, however it is limited to the internal controller [23]. Both platforms suffer by low degree of statistic gathering options.

There exists an implementation to the truly discrete simulation environment such as OMNeT++ [24]. It is also limited by only internal controller, but this solution can be better for collecting large statistical sets.

Another open solution for testing purposes is NICE project [25] focusing to the bug tracing. It forms an intermediate layer for NOX controller and helps to identify possible reasons of the third party application failure. Lastly there is a performance testing where for example sFlow can be used. Is able to monitor the number of active flows and it is described by RFC in [26].

For testing algorithms it would be appropriate to create a unified set of tests to ensure comparability of solutions supplied by third parties and by different device vendors.

3. Monitoring Application

We have realized that OF is suitable for resolving security threats in a way known as First Hop Security. Focusing the known security threats we have picked up attack against Duplicate Address Detection (DAD) which is used by network hosts during the IPv6 address assignment.

As the first step we have created a security monitoring application. It should serve as a tool for network administrators for identification threats and solving security conflicts in a local network. It is based on Floodlight controller in version 0.9. This stable version supports OF v1.0, thus without the IPv6 address support at all. We have chosen this version because it was the only one accessible and spread version among both physical switches and controllers. Thanks to the OF features we have found the problem solution which has proved to be challenging.

3.1. Problem Definition

Protocol DAD has a significant security issue in potential Deny of Service (DoS) attack as it is described in [27]. The situation when a potential host is unable to assign an address to itself is showed in Fig. 3.

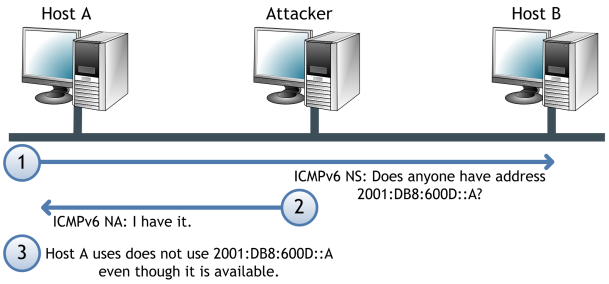

Fig. 3: Attack against IPv6 DAD.

There was an eminent need to extend the Floodlight module implementation for IPv6 address resolution.

3.2. Design and Implementation

Whole monitoring application is depicted in the deployment diagram in Fig. 4. The monitoring application is fundamentally based on the OpenFlow Security Engine (OFSE). OFSE accesses to Floodlight data via a REST API. Information about the network stored in the controller is transported to the OFSE in JSON format. A frontend graphical application Visualizing Console (VC) presents network topology with hosts by a graph using data obtained from OFSE.

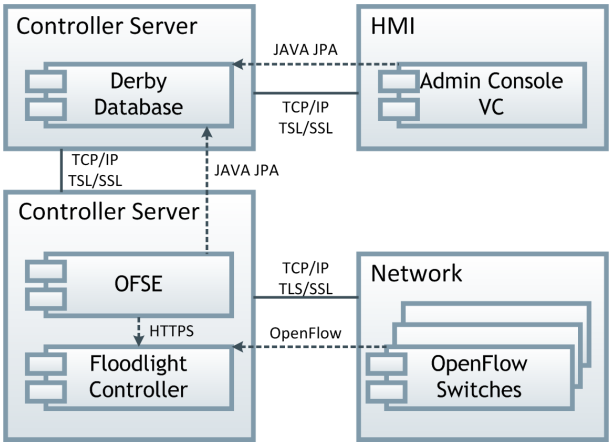

Fig. 4: Monitoring application deployment diagram.

Forwarding module in Floodlight handles every OpenFlow message for unknown data flow. It is sent from the OF switch to the controller. Floodlight has not implemented forwarding functionality for payload frames with Ethertype 0x08DD, thus IPv6. These frames fall into the default forwarding where the switch behaves as a usual learning switch. It was very essential to make modifications which add support to catch these types of frames.

Once a frame with IPv6 protocol is received by controller it is processed by the forwarding module. Then the IPv6 module receives frame thanks to listener; it parses the IPv6 address and it stores the address internally in the controller logic. Further code implementation results that these IPv6 addresses are accessible via the REST API of the Floodlight to OFSE.

In case of DAD security issue, once the OFSE detects a higher rate of new source IPv6 addresses than a trigger level assigned to one particular network host in a given period of time, it supposes it is the attack against DAD. OFSE then marks up the attacker in the database for VC as a potential threat in the network. It is then possible to browse through history of a network topology but also to automatically identify and visualize attackers in a network.

This application is a first step for future improvement when it could be extended to a form of a proactive tool which would allow immediate autonomy action. This is not possible in this version because of a lack of IPv6 tuples and the action depends on the network administrator.

3.3. Evaluation

During the development phase the application was tested in Mininet software within Xubuntu Linux distribution. In the next phase a real network with OF switches was created. It included two switches HP E3800 and one HP 5406zl. Multiple client hosts were connected to these switches with enabled IPv4/IPv6 interfaces. The output for one moment of the testing is in Fig. 5. Red circles represent switches, green ones are for connected hosts and blue one represents the last database action.

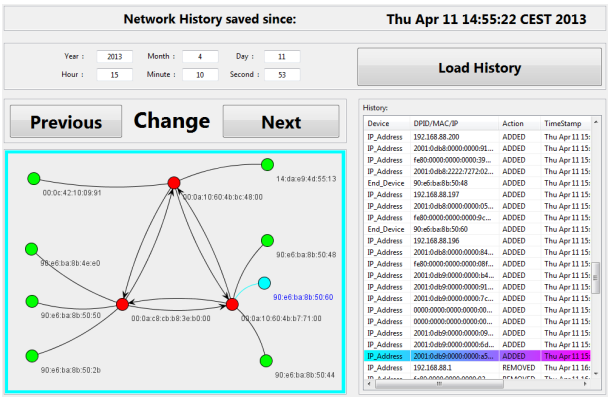

Fig. 5: Application screenshot from the real network testing.

There is a potential glitch in the functionality in case that matching rules within OF switch would be based only upon very low flow tuples e.g. MAC addresses. Since the general rules can cover the communication on higher levels, so it is not possible to catch an IPv6 address. Nonetheless, we found out during the testing on the real network that there was always additional

information which distinguished any type of communication even with the default forwarding and all IPv6 addresses were successfully captured.

4. Conclusion

The main goal of this paper is to present the current state of the OpenFlow protocol development. It seems that four years after the introduction of the first viable version the protocol still has a long way to the mass commercial deployment with the exception of specialized networks.

The analysis shows that the central control concentrated into a single controller brings a number of disadvantages that may be not suitable for certain deployments. Although the main expected deployment field is in data centers for optimizing communication within virtualized infrastructure, there exist use cases covering for example differentiated services on ISPs.

With the centralization there arise a number of questions, not only in a technical field such as ensuring greater availability of controllers but also how to share their control information between different operators. For this and other limitations we concluded that the appropriate OF deployment areas are primarily closed local area networks with a single administration without the critical infrastructure needs. We have also proposed the security demonstrational application solving OF 1.0 shortage in the lack of IPv6 support.

Acknowledgment

This work was supported by Grant Project SGS13/200/OHK3/3T/13.

References

[1] CASADO, M. and N. MCKEOWN. The virtual network system. In: *Proceedings of the 36th SIGCSE technical symposium on Computer science education - SIGCSE '05*. New York: ACM Press, 2005, pp. 76–80. ISBN 1-58113-997-7. DOI: 10.1145/1047344.1047383.

[2] OpenFlow Switch Specication: Version 1.0.0 (Wire Protocol 0x01). In: *OpenFlow* [online]. 2009. Available at: `http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf`.

[3] OpenFlow Management and Configuration Protocol: (OF-Config 1.1.1). In: *OpenFlow* [online]. 2013. Available at: `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1-1-1.pdf`.

[4] ENNS, R., M. BJORKLUND, J. SCHOENWAELDER and A. BIERMAN. Network Configuration Protocol (NETCONF): RFC 6241. In: *RFC Editor* [online]. 2011. Available at: `http://www.rfc-editor.org/rfc/rfc6241.txt`.

[5] SDN Use Cases. In: *SDN Central* [online]. 2012. Available at: `http://www.sdncentral.com/sdn-use-cases/`.

[6] TOOTOONCHIAN, A. and G. YASHAR. HyperFlow: A distributed control plane for OpenFlow. In: *INM/WREN '10: 2010 International Network Management Workshop/Workshop on Research on Enterprise Networking* [online]. 2010. Available at: `https://www.usenix.org/legacy/events/inmwren10/tech/full_papers/Tootoonchian.pdf`.

[7] KOPONEN, T., M. CASADO, N. GUDE, J. STRIBLING, L. POUTIEVSKI, M. ZHU, R. RAMANATHAN, Y. IWATA, H. INOUE, T. HAMA and S. SHENKER. Onix: a distributed control platform for large-scale production networks. In: *Proceedings of the 9th USENIX conference on Operating systems design and implementation*. Berkeley: USENIX Association, 2010, pp. 37–40. ISBN 978-1-931971-79-9.

[8] Nicira Network Virtualization Platform. In: *Nicira* [online]. 2012. Available at: `http://nicira.com/sites/default/files/docs/NVPDatasheet.pdf`.

[9] STRIBLING, J., Y. SOVRAN, I. ZHANG, X. PRETZER, J. LI, M. F. KAASHOEK and R. MORRIS. Flexible, wide-area storage for distributed systems with WheelFS. In: *NSDI '09: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation* [online]. 2009. Available at: `https://www.usenix.org/legacy/events/nsdi09/tech/full_papers/stribling/stribling.pdf`.

[10] YIN, H., H. XIE, T. TSOU, D. LOPEZ, P. ARANDA and R. SIDI. SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains. In: *The Internet Engineering Task Force (IETF)* [online]. 2012. Available at: `http://tools.ietf.org/html/draft-yin-sdn-sdni-00`.

[11] ZAREK, A. OpenFlow Timeouts Demystified. In: *Computer Engineering Research Group: University of Toronto* [online]. Toronto, 2012. Avail-

able at: http://www.eecg.toronto.edu/~lie/papers/zarek_mscthesis.pdf.

[12] HEGR, T., L. BOHAC, Z. KOCUR, M. VOZNAK and P. CHLUMSKY. Methodology of the direct measurement of the switching latency. *Przeglad Elektrotechniczny*. 2013, vol. 89, iss. 7, pp. 59–63. ISSN 0033-2097.

[13] ERICKSON, D. The Beacon OpenFlow Controller. In: *HotSDN'13* [online]. 2013. Available at: http://yuba.stanford.edu/~derickso/docs/hotsdn15-erickson.pdf.

[14] CAI, Z. *Maestro: Achieving Scalability and Coordination in Centralized Network Control Plane*. Houston, 2011. Dissertation thesis. Rice University.

[15] Exploring networks of the future. *GENI: Global Environment for Network Innovations* [online]. 2013. Available at: http://www.geni.net.

[16] KOPSEL, A. and H. WOESNER. OFELIA-pan-european test facility for openflow experimentation. In: *4th European Conference – Towards a Service-Based Internet*. Berlin: Springer, 2011, pp. 311–312. ISBN 978-3-642-24755-2.

[17] GUDE, N., T. KOPONEN, J. PETTIT, B. PFAFF, M. CASADO, N. MCKEOWN and S. SHENKER. NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*. 2008, vol. 38, iss. 3, pp. 105–110. ISSN 0146-4833.

[18] IBM Programmable Network Controller. In: *IBM* [online]. 2012. Available at: http://public.dhe.ibm.com/common/ssi/ecm/en/qcd03018usen/QCD03018USEN.PDF.

[19] ProgrammableFlow Controller. In: *NEC* [online]. 2012. Available at: http://www.necam.com/SDN/doc.cfm?t=PFlowController.

[20] OFTest framework. In: *Project Floodlight* [online]. 2013. Available at: http://www.projectfloodlight.org/oftest/.

[21] IXIA OpenFlow Validation Solutions. In: *IXIA* [online]. 2013. Available at: www.ixiacom.com/pdfs/library/quick_ref_sheets/OpenFlow-qrs.pdf.

[22] LANTZ, B., B. HELLER and N. MCKEOWN. A network in a laptop: rapid prototyping for software-defined networks. In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. New York: ACM, 2010, pp. 1–6. ISBN 978-1-4503-0409-2. DOI: 10.1145/1868447.1868466.

[23] NS-3 OpenFlow switch support. In: *NS-3.13 documentation* [online]. 2013. Available at: http://www.nsnam.org/docs/release/3.13/models/html/openflow-switch.html.

[24] KLEIN, D. and M. JARSCHEL. An OpenFlow Extension for the OMNeT++ INET Framework. In: *Universitaet Wuerzburg* [online]. 2013. Available at: http://www3.informatik.uni-wuerzburg.de/research/ngn/ofomnet/paper-acm_with_font.pdf.

[25] CANINI, M., D. VENZANO, P. PERESINI, D. KOSTIC and J. REXFORD. A NICE way to test OpenFlow applications. In: *9th USENIX Symposium on Networked Systems Design and Implementation*. San Jose: UNISEX Association, 2012, pp. 127–140. ISBN 978-931971-92-8. Available at: https://www.usenix.org/system/files/tech-schedule/nsdi12_proceedings_full.pdf.

[26] PHAAL, P., S. PANCHEN and N. MCKEE. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks: RFC 3176. In: *RFC Editor* [online]. 2001. Available at: http://www.rfc-editor.org/rfc/rfc3176.txt.

[27] NIKANDER, P., J. KEMOF and E. NORDMARK. IPv6 Neighbor Discovery (ND) Trust Models and Threats: RFC 3756. In: *RFC Editor* [online] 2004. Available at: http://www.rfc-editor.org/rfc/rfc3756.txt.

About Authors

Tomas HEGR was born in 1988. He received his M.Sc. in computer science at the Czech Technical University in Prague in 2012. His research interests include industrial networks based on Ethernet and SDN.

Leos BOHAC received the M.S. and Ph.D. degrees in electrical engineering from the Czech Technical University, Prague, in 1992 and 2001, respectively. Since 1992, he has been teaching optical communication systems and data networks with the Czech Technical University, Prague. His research interest is on the application of high-speed optical transmission systems in a data network.

Vojtech UHLIR was born in 1990. He received B.Sc. in Computer Systems in 2013 at FEE, CTU in Prague. His focus is in computer programming, embedded systems and networks.

Petr CHLUMSKY was born in 1985. He received his M.Sc. from the Czech Technical University in Prague in 2010. Since 2010 he has been studying Ph.D. degree. His research interests include wireless transmission, network coding and network simulation.