Enhancing Availability of Services Using Software-Defined Networking

Martin KLEPAC, Tomas HEGR, Leos BOHAC

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

klepamar@gmail.com, tomas.hegr@fel.cvut.cz, bohac@fel.cvut.cz

DOI: 10.15598/aeee.v13i5.1498

Abstract. *The immense growth of client requirements imposed on data centre and cloud providers results in a conflict with traditional networking concepts lacking the required agility. In order to promote flexibility, which data centre providers promise to their clients, this discrepancy needs to be resolved, for instance by employing the novel concept of Software-Defined Networking (SDN). This paper utilises this concept in order to minimise service downtime while performing live virtual machine migration. The work is aimed at small/medium-sized data centres and hence the findings are based on real communication patterns found in such environments. Results show that packet loss is slightly diminished while available throughput is increased thanks to the proactive approach taken during network topology changes when compared to the traditional approach based on L2 forwarding.*

Keywords

Live VM migration, ONOS, Open vSwitch, SDN, software defined networking.

1. Introduction

In the last couple of years, the business has largely focused on improving its efficiency. In terms of IT and its infrastructure, a transition from maintaining a dedicated server room with a number of physical hosts to running visualized servers hosted by a cloud provider has taken place.

The set of requirements held by clients affects how cloud providers organize their internal data centre networking topology. The requirement of flexibility and scalability, expressed by building numerous virtual machines (VMs) and virtual networks practically instantaneously, must also be considered when employing a particular network design.

Traditional networking principles fail to comply with the requirements mentioned above, notably flexibility when it comes to creating new virtual networks and applying security policies. Networking has long been known for the burden of configuring every device separately. Each device, be it a switch or router, represents an independent instance with its own logic. The distributed concept of control plane thus hinders flexibility as required by customers.

The flexibility problem defined in the previous paragraph is aimed to be solved by an emerging *Software-Defined Networking* (SDN) paradigm. The paper and the subsequent application focuses on a service availability improvements in SDN-based data centers mostly in conjunction with an orchestrator system, which is responsible for providing and keeping track of virtual machines on a hypervisor level. The findings show that available throughput during the critical phase of live VM migration is approximately doubled when compared to the traditional, non-SDN-based approach. On the other hand, average packet loss throughout the entire VM migration is slightly diminished in the SDN-based scenario.

The paper is structured as follows. In the following Section 2. , related works are discussed. Next, we briefly analyze the problem of live VM migration from the networking perspective in Section 3. . The structure and requirements of the implemented application with regard to other components will is outlined in Subsection 3.1. and Section 4. . The performance of the program, which proactively implements topology changes, is to be compared to the traditional, reactive L2 forwarding in datacentre-like topology in Section 5.

2. Related Works

The actual principles and techniques of VM migration are not discussed in the paper, as they have been well described elsewhere, for instance in [1].

From the networking perspective, topology changes following a live VM migration in traditional L2 switching include a rewrite of CAM table entries for a VM MAC address as a result of VM relocation. The most recent arrival port is always associated with a given MAC address within the switch CAM table – at least in the Cisco Catalyst series, but the same concept is expected to be utilized universally [2]. CAM entries are updated *reactively* – when the newly migrated VM attempts to communicate. This creates a potential black hole because switches are unaware of any changes and will send the traffic to the original host until the VM starts communicating from a different host.

The *proactive* approach usually relies on sending *gratuitous ARP* (Address Resolution Protocol) either by the destination hypervisor or the VM itself. Gratuitous ARP sent by the VM itself to some extent contradicts the transparency requirement by which the VM should be completely unaware of its migration. This approach is employed by open source Xen hypervisor [3], whereas OpenNebula orchestrator system (and the underlying Kernel-based Virtual Machine (KVM) hypervisor) utilizes gratuitous ARP sent by the physical host [4].

VMware uses a different approach and instead of gratuitous ARP, ESXi hosts send *reverse ARP* (RARP) packets on behalf of the VM undergoing live migration. Since RARP is now considered obsolete, the broadcast message sent by the hypervisor is ignored by hosts within the broadcast domain with the exception of switches, which update their CAM entries as a result of receipt of such packets [5].

In terms of overlay networks realizing multi-tenancy within a data centre, several solutions providing layer 2 segments above existing IP-based (layer 3) networks have emerged in the past couple of years.

VLANs have been the most prevalent overlay technology for the past 15 years, but the limited scalability (up to 4096 VLANs) is unacceptable in large data centres. Nevertheless, VLANs represent a traditional and well-supported traffic isolation mechanism befitting most other environments.

More robust solutions employ multicast/unicast VXLAN technology. While the former is standardized in [6], the latter is VMware proprietary, and hence, has become an integral part of collaborative effort by VMware and Cisco in Cisco Nexus 1000V distributed virtual switch located within ESXi hypervisor.

VMware NSX for multi-hypervisor environment represents another overlay solution. However, instead of proprietary virtual switches it employs Open vSwitch (OVS), a de-facto Openflow (virtual) switch reference implementation [7]. Openflow is the most significant SDN-based protocol through which *SDN controller*, downloads appropriate forwarding rules to both physical and virtual networking devices.

3. Analysis

The orchestrator system within a data center is an entity that possesses all the required information regarding virtual machines of respective tenants (allocated hardware resources, location, billing information). A simplistic life cycle of a VM within an orchestrator system is depicted in Fig. 1. The figure does not take into account several states such as suspend, resume or delete. Instead, the gray triangle focuses on live migration only. VM live migration may succeed, resulting in VM being hosted by a different hypervisor. On the other hand, due to a lack of computing resources within the potentially new hypervisor or networking issues, live migration may fail, and the control application must be capable of handling both cases.

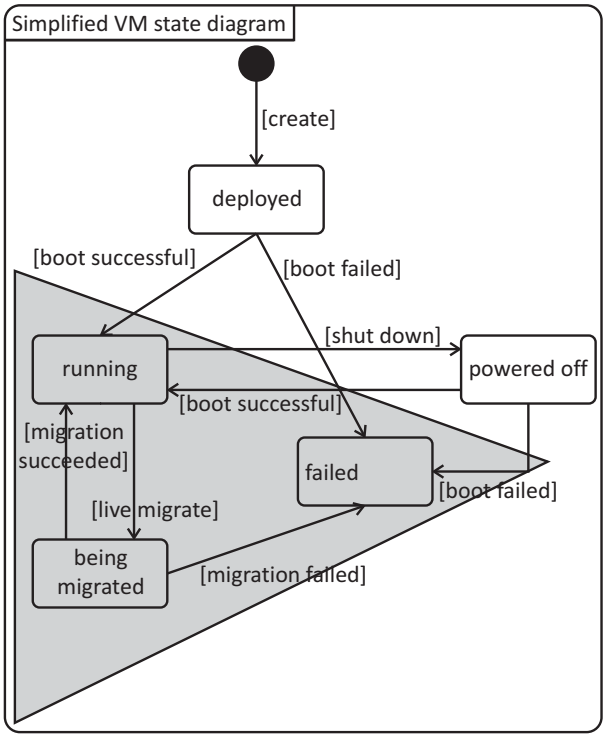


Fig. 1: Simplified life cycle of a VM within an orchestrator system.

3.1. Requirements to Control Application

The purpose of the control application is to implement topology changes resulting from a live VM migration proactively using the principles of SDN, i.e. centralized control plane downloads forwarding rules to respective networking devices. In other words, the goal of the application is to ensure that all established communication between the VM undergoing migration and other entities persists after the migration. More specifically, the list of requirements imposed on the application is as follows:

- decrease in service downtime period during live VM migration,
- cooperation between the orchestrator system and the SDN controller,
- adoption of OpenFlow protocol to manage forwarding rules,
- multi-tenancy via the means of VLANs.

4. Implementation

The control application assumes *OpenNebula* as the orchestrator system. It gathers information from multiple sources, namely *libvirt* virtualization API and OpenNebula orchestrator system. The application needs to obtain data in terms of VM location from OpenNebula, perform some computation and propagate the results to ONOS. The controller then updates underlying flow entries using Openflow protocol. The flow of information between the application and other entities is highlighted in Fig. 2.

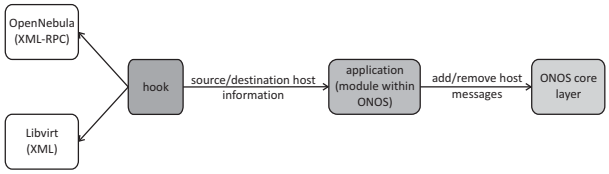


Fig. 2: Flow of information between components.

By means of XML-RPC, the application queries OpenNebula and obtains network-related (network, subnet mask, VLAN ID) as well as location-related pieces of information. However, OpenNebula does not possess all the required information in time, and thus, additional communication with libvirt API is required. Unlike OpenNebula, which performs regular polling to determine the state of its virtual machines, libvirt uses the concept of hooks. Hooks are custom scripts triggered by specific system events [8]. This allows the application to determine the source and destination physical servers on top of which the VM undergoing live migration is being run (i.e. the source host) or will be hosted after the migration (i.e. the destination host). Libvirt hook is implemented as a simple bash script, which asynchronously informs the application of a live VM migration.

In terms of communication with ONOS, the application is run as a module within ONOS, and therefore it possesses access to ONOS internal structures and methods.

The access to ONOS internal resources is divided into two areas. While any application may access existing topology information, only a subset of applications is allowed to communicate with the ONOS core layer responsible for performing topology changes (adding switch, end hosts, etc.). Therefore, during a live VM migration, the application requests ONOS core layer to register VM undergoing live migration using the new location. On the other hand, the application requests ONOS core layer to deregister the VM with the original location while keeping VM forwarding rules.

The overall list of components along with their versions is depicted in Tab. 1.

Tab. 1: Versions of SW components.

Type of SW	Name	Version
Orchestrator system	OpenNebula	4.12
Hypervisor management	Libivrt	1.2.9-9
SDN controller	ONOS	1.0.1

5. Evaluation

5.1. Lab Environment

In order to create a data-centre-like architecture, leaf and spine design has been designated. The topology initially consisted of physical switches of multiple vendors – HP with their Procurve and Comware series and Cisco Catalyst series. All physical switches support OpenFlow 1.3 and hence were supposed to be managed by an instance of ONOS SDN controller.

Additionally, two physical servers – HP DL360 G7 and Dell R210 II – were connected to leaf switches at the opposite edge of the network. The physical servers run OpenNebula and host virtual machines, which are migrated from the original server across the entire network to the other host. Both servers run Open vSwitch and hence are also managed by ONOS via OpenFlow 1.3. The out-of-band approach to OpenFlow control has been chosen, i.e. a *controlled* VLAN is managed by ONOS, whereas the *controlling* VLAN, which inter-

connects ONOS and respective networking devices, is managed using traditional L2 forwarding.

During the evaluation, it turned out that physical switches (both HP and Cisco) refused to be managed by an instance of ONOS SDN controller. In case of HP, ONOS according to one of its developers sends layer 2 rules down to table 0 of the switch expecting a single table abstraction [9]. However, HP switches implement OpenFlow table 0 as read-only and, therefore, refuse to add any forwarding rules into it. In case of Cisco, ONOS developer this time could not detect the cause of the malfunction. These problems were probably caused by immaturity of the OpenFlow implementation on both sides ONOS controller and Cisco switches.

All in all, an alternative topology consisting of merely Open vSwitches had to be considered and implemented. In order to utilise leaf and spine design, physical switches had to be replaced with dedicated physical servers running Open vSwitch. The final topology is depicted in Fig. 3.

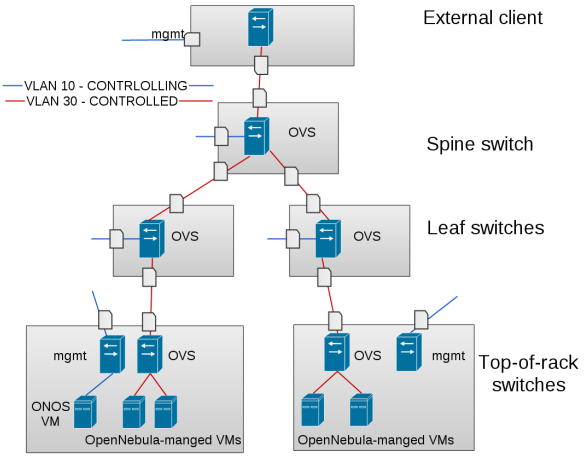


Fig. 3: Networking topology used for the application evaluation.

5.2. Methodology

Cisco states that approximately 75 percent of the data centre traffic is local, and hence, never leaves the premises of DC [10]. Examples promoting east-west traffic pattern in data centers include:

- communication between web/application and database server – short SQL select may fetch a large portion of the database,
- parallel computing – a master node allocates tasks that are computed by slave nodes passing the complex results back to the master.

These traffic flow patterns were simulated using *iperf* and *flowping* utilities. The iperf application was used to measure throughput for TCP communication between an iperf client and server. The traffic is sent by the iperf client and received on the server side. Flowping, on the other hand, is a UDP-based application, which provides an alternative to regular ping utility [11]. Unlike ping, which measures latency at L3, flowping measures latency as detected by an application layer, since it operates at L7 of ISO/OSI model.

Tab. 2: Description of tools employed during the application evaluation.

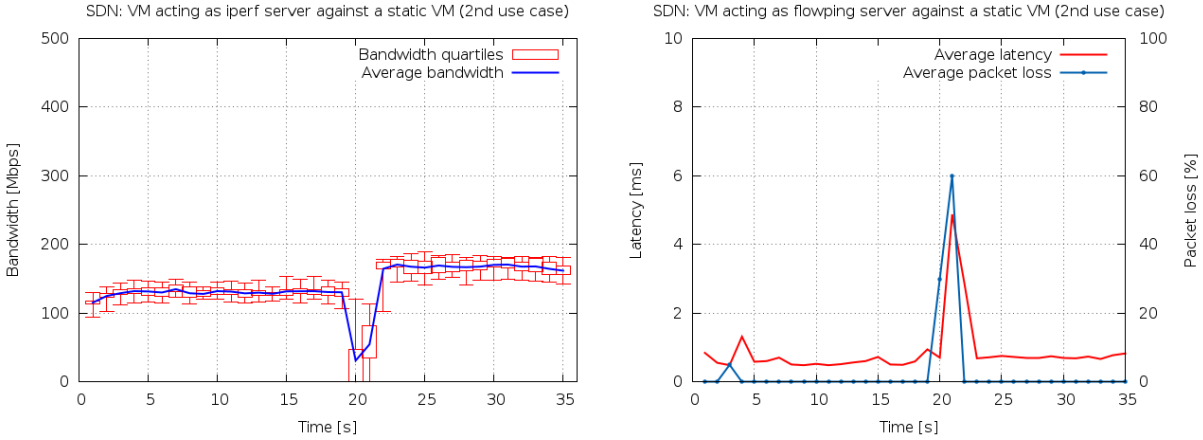
Tool	iperf	flowping
Entity	Throughput (TCP)	Latency, Packet Loss (UDP)
SW version	2.0.5	1.2.6
Invocation on the server	`iperf -s --interval 1`	`flowping -S -q`
Invocation on the client	`iperf -c SERVER -f m -t 50 --interval 1`	`flowping -h SERVER -c 50 -f FILE`

Scenarios mentioned below were evaluated for both SDN and non-SDN cases. The SDN scenario utilizes the control application implemented in chapter 4, whereas the non-SDN scenario will employ the traditional L2 learning switching. Both uses the topology as displayed in Fig. 3 – the only difference being the fact that in the non-SDN mode Open vSwitches handle forwarding themselves without being managed by an external SDN controller. The evaluated test cases are as follows:

- VM being migrated (iperf client) vs. static VM (iperf server). The VM being migrated acts as a database server while the static VM represents an application/web server.
- VM being migrated (iperf server) vs. static VM (iperf client). The VM being migrated acts as a web/application server, whereas the static VM represents a DB server.
- VM being migrated (iperf client) vs. physical host (iperf server). The VM being migrated acts as a server for an external client, e.g. web, FTP, application server. This use case simulates external traffic leaving the DC premises. The external client performs download operation.

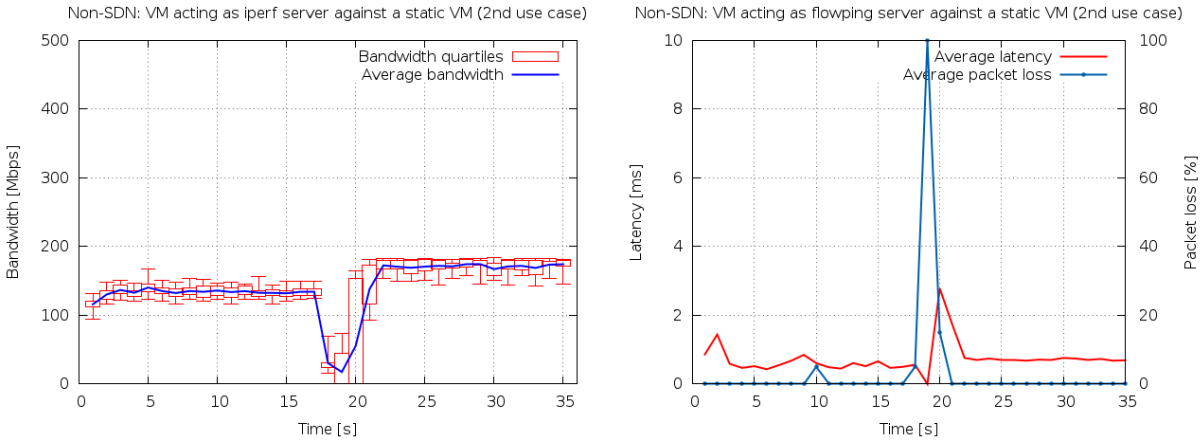
5.3. Results

The following section points out the results of the evaluation based on the three test cases defined in the previous Subsection 5.2. All test case measurements



(a) Iperf (throughput) measurements (b) Flowping (latency, packet loss) measurements

Fig. 4: SDN: VM acting as iperf (a) and flowping (b) server against a static VM (2nd use case).



(a) Iperf (throughput) measurements (b) Flowping (latency, packet loss) measurements

Fig. 5: Non-SDN: VM acting as iperf (a) and flowping (b) server against a static VM (2nd use case).

were performed using both the SDN-based and non-SDN-based principles. The resultant figures Fig. 4 and Fig. 5 corresponding to either iperf throughput or flowping latency measurements are an average of 20 VM migrations. These figures point the results for the second test case, but, generally speaking, all test cases exhibit similar behavior. What is common to all of the test cases is the fact that VM migration finishes between 18th and 22nd second. During this critical time average frame packet loss and latency increases, whereas throughput drops. This applies to both the SDN-based and non-SDN-based scenarios.

The second test case represents a situation in which the migrated VM acts as an iperf/flowping server against a static VM running iperf and flowping client applications. Originally, both virtual machines reside on the same host, but once the migration has been completed, the traffic between the VMs spans across the entire data center. The SDN case (Fig. 4) is superior to its non-SDN counterpart (Fig. 5) in all major aspects – average packet loss during the entire migration (2.85 % compared to 3.75 % in the traditional L2 forwarding), flowping downtime during the critical part of VM migration (0.9 second vs 1.2 seconds) and last but not least, throughput during the transition (15.5 Mbps for the SDN case is almost double the figure obtained in the non-SDN case – 7.5 Mbps).

Table 3, Tab. 4 and Tab. 5 compare and contrast the results of respective test cases for both the SDN and non-SDN scenarios. Besides average packet loss throughout the entire migration, average downtime and minimal throughput during the critical phase of VM migration, the tables contain expected values and corrected standard deviations for maximum latency as detected by flowping during the 5-second time frame (18th-22nd second).

Proactive topology changes via the means of SDN result in better performance during the critical phase of VM migration for all relevant attributes (service downtime, throughput). Average packet loss throughout the entire migration is also lower in the SDN case. Effectively, if iperf and flowping were replaced by real-world services, VM undergoing migration would probably lose fewer TCP sessions (which corresponds to a decrease in flowping downtime) while serving more clients (which corresponds to an increase in iperf throughput).

Tab. 3: First test case performance comparison.

Aspect	SDN	Non-SDN
Total packet loss	2.25 ± 1.33 %	2.40 ± 1.23 %
Lack of connectivity during transition	0.75 ± 0.44 s	0.80 ± 0.41 s
Minimal throughput during transition	34.26 ± 34.05 Mbps	13.68 ± 15.33 Mbps
Maximal latency during transition	4.76 ± 0.74 s	3.41 ± 2.53 s

Tab. 4: Second test case performance comparison.

Aspect	SDN	Non-SDN
Total packet loss	2.85 ± 1.18 %	3.75 ± 1.33 %
Lack of connectivity during transition	0.90 ± 0.30 s	1.20 ± 0.41 s
Minimal throughput during transition	15.58 ± 14.93 Mbps	7.45 ± 11.54 Mbps
Maximal latency during transition	4.50 ± 0.49 s	2.94 ± 1.23 s

Tab. 5: Third test case performance comparison.

Aspect	SDN	Non-SDN
Total packet loss	2.40 ± 1.23 %	2.85 ± 0.67 %
Lack of connectivity during transition	0.75 ± 0.44 s	0.95 ± 0.22 s
Minimal throughput during transition	31.20 ± 36.63 Mbps	15.68 ± 9.72 Mbps
Maximal latency during transition	2.89 ± 0.37 s	0.80 ± 0.60 s

6. Conclusion

The purpose of the paper was to analyze the extent to which the novel concept of software-defined networking represents a feasible alternative to the traditional networking concepts in the small/medium-sized datacenter environment. The paper specifically focused on a single use case – service availability enhancement during live VM migration.

A significant portion of the paper is dedicated to the analysis and implementation of the application maintaining flow rules among the communicating entities. The application is run as a module within ONOS SDN controller and gathers information from multiple sources, namely libvirt virtualisation API and OpenNebula orchestrator system. The application proactively performs topology changes (i.e. relocation of the VM undergoing migration) in accordance with the SDN principles.

The application is evaluated using leaf-and-spine network topology, which is commonplace in data centers. Since the original topology consisting of both physical and virtual switches cannot be fully managed by ONOS SDN controller, an alternative topology comprised of virtual switches was created. Throughput, latency and packet loss are measured for three particular use cases, which represent both intra-DC and external communication. The measurements are carried out for both the SDN-based and non-SDN-based scenarios.

The proactive approach taken by the application decreases packet loss and provides larger throughput during the critical phase of VM migration.

Although the SDN-based approach does not outperform the traditional L2 forwarding dramatically, one should keep in mind that SDN is more than a novel traffic forwarding mechanism – instead, implementing security policies such as port security may potentially be solved in a programmatic manner via SDN.

On the other hand, the problems encountered while trying to manage physical switches of multiple vendors by ONOS SDN controller emphasize the current immaturity of SDN implementations. However, the results indicate that the ability to program networks, which is the principal advantage of SDN over traditional networking, will prevail and SDN will eventually become commonplace in certain environments, data centers being one of them.

Acknowledgment

This work was supported by Czech Technical University in Prague grant no. SGS13/200/OHK3/3T/13.

References

[1] CLARK, C., K. FRASER, S. HAND, J. G. HANSEN, E. JUL, C. LIMPACH, I. PRATT, A. WARFIELD. Live Migration of Virtual Machines. In: *NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation.* Berkeley: USENIX, 2005, pp. 273-286.

[2] Catalyst 6500/6000 Switches ARP or CAM Table Issues Troubleshooting. *Cisco* [online].

2009. Available at: http://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/71079-arp-cam-tableissues.html.

[3] JALAPARTI, V. [Xen-users] Xen live migration: from where is the ARP sent? *GossamerThreads* [online]. 2011. Available at: http://www.gossamer-threads.com/lists/xen/research/201826.

[4] KOOMAN, S. VM HOOK scripts to send Gratuitous ARP replies on behalf of VMs in OpenNebula cloud (ONE). *GitHub* [online]. 2014. Available at https://github.com/hydro-b/one-grarp.

[5] NOBEL, R. The vSwitch "Notify Switches" setting. *Rickard Nobel* [online]. 2012. Available at http://rickardnobel.se/vswitch-notify-switches-setting/.

[6] IETF RFC 7348. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. *Internet Engineering Task Force* [online]. 2014. Available at: http://tools.ietf.org/html/rfc7348.

[7] PEPELNJAK, I. What is OpenFlow (part 2)? *IpSpace* [online]. 2011. Available at: http://blog.ipspace.net/2011/10/what-is-openflow-part-2.html.

[8] MONTERO, R. S. How does opennebula monitor the vm's state. *Opennebula* [online]. 2012. Available at: http://lists.opennebula.org/pipermail/users-opennebula.org/2012-November/038418.html.

[9] KLEPAC, M. ONOS vs. physical switches. *Groups* [online]. 2015. Available at: https://groups.google.com/a/onosproject.org/forum/#!topic/onos-dev/U_BcylyY_kw.

[10] Cisco Global Cloud Index: Forecast and Methodology 2013–2018 White Paper. *Cisco* [online]. 2014. Available at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html.

[11] VONDROUS, O. FlowPing - UDP based ping application. *Comtel* [online]. 2012. Available at: http://flowping.comtel.cz/.

About Authors

Martin KLEPAC was born in Levice, Slovakia. His research interests entail cloud computing and virtualization.

Tomas HEGR received his M.Sc. in computer science at the Czech Technical University in Prague in 2012. He participates in teaching activities at the department of Telecommunication engineering. His research interests involve industrial networks based on Ethernet and Software-Defined Networking in all research areas.

Leos BOHAC received the M.Sc. 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.