CLOUDS-Pi: A Low-Cost Raspberry-Pi based Micro Data Center for Software-Defined Cloud Computing

Adel Nadjaran Toosi
Monash University, Australia

Jungmin Son
The University of Melbourne, Australia

Rajkumar Buyya
The University of Melbourne, Australia

Software Defined Networking (SDN) is rapidly transforming the networking ecosystem of cloud computing data centers. However, replicating SDN-enabled cloud infrastructures to conduct practical research in this domain requires a great deal of effort and capital expenditure. In this paper, we present the CLOUDS-Pi platform, a small-scale cloud data center for doing research on software-defined clouds. As part of it, Open vSwitch (OVS) is integrated with Raspberry-Pis, low-cost embedded computers, to build up a network of OpenFlow switches. We provide two use cases and perform validation and performance evaluation. We also discuss benefits and limitations of CLOUDS-Pi in particular and SDN in general.

The need for agile, flexible, and cost-efficient computer networks has formed the nucleus for the global efforts towards Software-defined networking (SDN). SDN is an emerging approach to computer networking that separates the tightly coupled control and data (forwarding) planes in traditional networking devices. Thanks to this separation, SDN can provide a logically centralized view of the network in a single point of management. This is achieved via open interfaces and abstraction of lower-level functionalities and transforms the network to a programmable platform that can dynamically adapt its behavior. SDN is becoming so popular that its usage has been spread beyond ordinary networks and is even suggested to address challenges of smart grid systems.[1]

Cloud computing is a successful computing paradigm, which delivers computing resources residing in providers' data centers as a service over the Internet on a pay-as-you-go basis. With the

growing adoption of cloud, data centers hosting cloud services are rapidly expanding their sizes and increasing in number. Therefore, resource management in clouds' large-scale infrastructure becomes a challenging issue. In the meantime, SDN is increasingly being accepted as the new generation of networks in cloud data centers where there is a need for efficient management of large multi-tenant networks of such dynamic and ever-changing environments. In fact, SDN not only reduces the complexity seen in today's cloud data center networks but also helps cloud providers to manage network services from a central management point.

The peaceful meeting of cloud computing and SDN embarks on significant innovation and research activities to fuse these together.[12] However, evaluation and experimentation of SDN-based applications for cloud environments present many challenges in terms of complexity, scaling, accuracy, and efficiency.[7] Major classes of performance evaluation techniques for these methods can be categorized as analytical modeling, simulation, emulation, and measurement. If we exclude analytical modeling due to its limitations and complexity, simulation tools (e.g., CloudSimSDN[16]) play significant roles in the performance evaluation of these methods due to their flexibility and affordability. However, simulations always present imperfect models and are limited by the degree of reproducing real world software-defined clouds. Emulation techniques, on the other hand, seem to be the method of choice for achieving a detailed understanding of the operation of software-defined clouds. Software emulators (e.g., Mininet[11]) expedite prototyping of SDN on a single machine. But there is not enough support for network dynamicity and the performance measurement of the virtualized hosts and Virtual Machines (VMs) in such tools[10]. Rapid and affordable prototyping of algorithmic advances in software-defined clouds is challenging and significant capital expenditure is required to replicate practical implementations and performance measurements.

With these issues in mind, this paper puts together all elements to build a low-cost micro software-defined data center by leveraging off-the-shelf hardware and open source software. We propose a system architecture for constructing a testbed/micro data center for researching SDN-enabled cloud computing. We focus on the cost-effectiveness of our setup by reusing existing equipment and coping with the budget and space limitations of an academic research laboratory. One of the most important benefits of SDN is that commodity hardware can be used to build networking devices. Therefore, we use Raspberry Pis, low-cost and small single-board computers, to create a small-scale data center network. To make a switch out of Raspberry Pi, we integrate each Pi with an Open vSwitch (OVS), which is one of the most widely used virtual switches in SDN. In summary, our main contributions are:

- We propose a system architecture and design to build a low-cost experimental testbed/infrastructure for conducting practical research in the domain of software-defined clouds.
- We present challenges, detailed requriements, all the required elements, and our expriences towards building CLOUDS-Pi, a testbed/micro data center built in the CLOUDS laboratory for researching SDN-enabled cloud computing based on our proposed recipe.
- We discuss two use cases to illustrate how CLOUDS-Pi can be utilized to offer applied solutions and drive research innovations.
- We explore benefits and limitations of such testbed.

SYSTEM ARCHITECTURE

In this section, we present the CLOUDS-Pi platform along with the physical infrastructure setup and utilized software. It is important to note that in contrast to other works in this domain such as Zebra[3] or Elasticcon[4] that particularly focus on architectural frameworks for advancing SDN controllers, we intend to use current advances in SDN and cloud technologies to put together a recipe for constructing a platform for conducting empirical research in software-defined clouds.

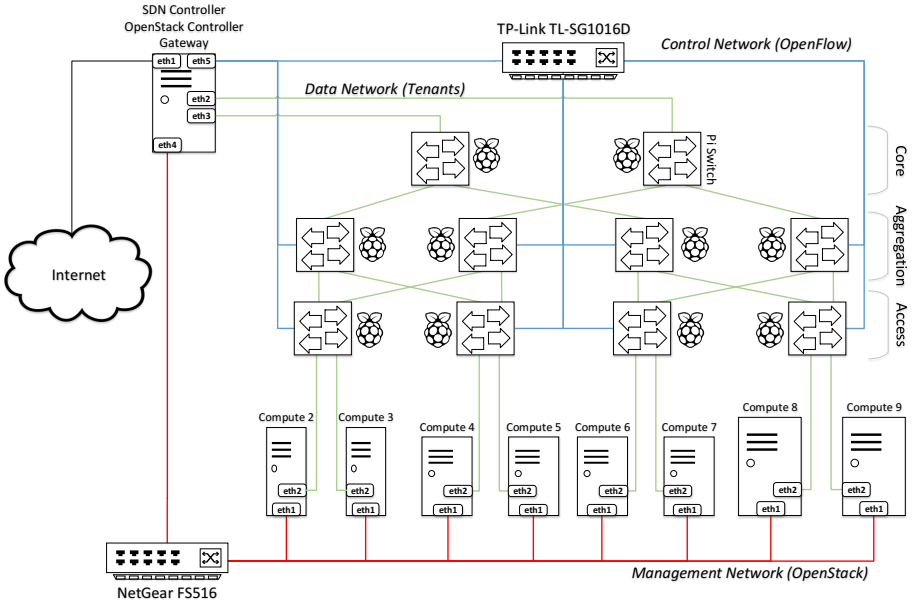

FIGURE 1: System Architecture of CLOUDS-Pi.

Physical infrastructure: The primary aim of our small cloud data center is to provide an economical testbed for conducting research in Software-Defined Clouds (SDCs). Therefore, we focus on reusing existing infrastructure, equipment, and machines (hosts) connected through a network of OpenFlow switches made out of Raspberry Pis, used by others as compute resources[17]. Our platform is comprised of a set of 9 heterogeneous machines with specifications shown in Table 1. To keep up with the common practice, we use three separate networks namely, 1) management, 2) data and 3) control. The *management* network is used by OpenStack, our deployed cloud operating system, for internal communication between its components and resource management. It constitutes a 16-port 10/100Mbps Ethernet Switch (NetGear Model FS516) connecting hosts and OpenStack controller. The *data* network is used for data communication among VMs deployed within the cloud environment and providing Internet access to them through the gateway host. This network is 100Mbps fat-tree[5] like network built on top 10 Raspberry Pis (Pi 3 MODEL B) with OVS integrated each playing a role of 4-port switch with an external port for control. Raspberry Pi 3 Model B has four USB 2.0 port and only one Ethernet interface. Therefore, we used *TP-Link UE200 USB 2.0 to 100Mbps Ethernet* adapters to add four extra Ethernet interfaces to the Raspberry Pi switch. The *control* network connects control ports on Raspberry Pi Switches to the SDN controller and transfers OpenFlow packets between the controller and the switches.

To monitor power consumption of individual machines, all cluster nodes are connected to two Eaton EMAB03 vertical managed enclosure Power Distribution Units (ePDUs). Eaton ePDUs allow us to monitor and switch on/off power outlets connected to an individual machine remotely through the network. These measurements can be used to evaluate the energy efficiency of the testbed and developed algorithms. Figure 1 depicts our data center setup including the network topology for management, data, and control networks and Figure 2 depicts the CLOUDS-Pi platform.

Software: We installed CentOS 7.0 Linux distribution as the host operating system on all nodes. Then using RDO Packstack, we installed OpenStack to build our cloud platform. Given that the OpenStack controller resides beside the SDN controller, we used one of our more powerful machines (IBM X3500) as the controller node. All other nodes play the same role of compute hosts in the design. In addition, we created NAT forwarding rules on the controller using Linux iptables to enable all other nodes to connect to the Internet. In this way, the controller node configured as the default Gateway for the external network access (Internet access) for all other nodes and OpenStack VMs. We also setup L2TP/IPsec Virtual Private Network (VPN) server

using OpenSwan and xl2tpd Linux packages on the controller node to provide direct access to VMs for our cloud users outside the data center network (see Figure 3).

Table 1. Specifications of machines in CLOUDS-Pi.

Machine	CPU	Cores	Memory	Storage
3 x IBM X3500 M4	Intel(R) Xeon(R) E5-2620 @ 2.00GHz	12	64GB (4 x 16GB DDR3 1333MHz)	2.9TB
4 x IBM X3200 M3	Intel(R) Xeon(R) X3460 @ 2.80GHz 4 16GB	4	16GB (4 x 4GB DDR3 1333MHz)	199GB
2 x Dell OptiPlex 990	Intel(R) Core(TM) i7-2600 @ 3.40GHz 4 8GB	4	8GB (2 x 4GB DDR3 1333MHz)	399GB

FIGURE 2: The CLOUDS-Pi Platform.

CLOUDS-Pi uses OpenDaylight (ODL), one of the popular open-source SDN controllers, to provide the brain of the network and handle OpenFlow capable Raspberry Pi switches. ODL is installed on the same host as the OpenStack controller and manages OpenFlow switches via control network (see Figure 3). Every Raspberry Pi in our setup uses a Debian-based Linux operating system of Raspbian Version 8 (Jessie) and have OVS Version 2.3.0 installed as an SDN-capable virtual switch. We configured OVS as a software switch having all USB-based physical interfaces connected as forwarding ports and used Raspberry Pi built-in interface as an OpenFlow control port.

We also developed some open source tools (https://github.com/Cloudslab/sdcon) to provide integrated manageability and monitoring for our platform. For example, *Status Visualizer* is a tool implemented to visualize traffic flows among hosts and virtual machines in the network. This module retrieves the topology information and then plots network links along with the real-time monitored flows with different colors where the thickness of the connector line represents used bandwidth. Figure 4 depicts the screenshot of Status Visualizer web UI showing sample traffic in CLOUDS-Pi between group of VMs.

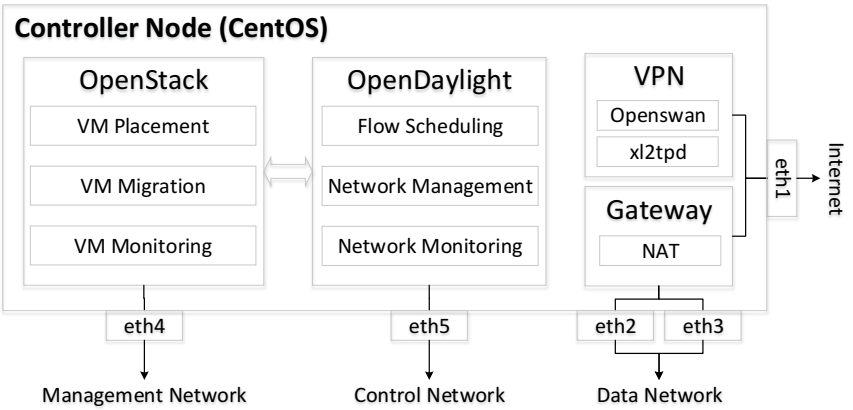

FIGURE 3: Software Stack on the Controller Node.

USE CASES

We introduce two use cases to illustrate how the CLOUDS-Pi platform and its SDN-enabled features can be utilized to offer solutions and drive research innovations. The first use case shows dynamic flow scheduling for efficient use of network resources in a multi-rooted tree topology. The second use case demonstrates that the communication cost of pairwise VM traffic can be reduced by exploiting collocation and network locality through live VM migration.

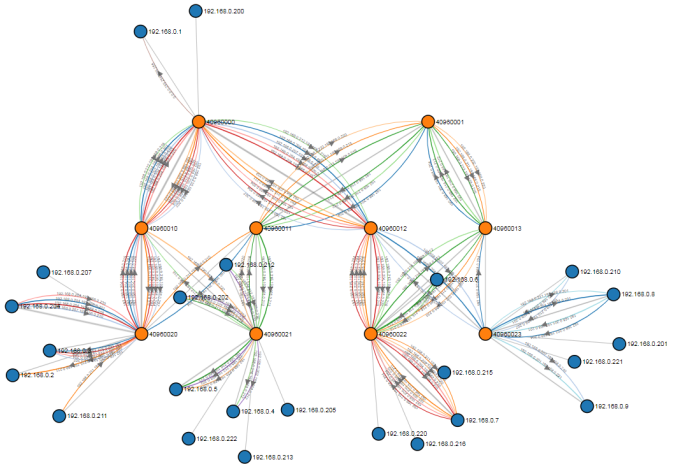

FIGURE 4: Sample network traffic in CLOUDS-Pi visualized by Status Visualizer.

Dynamic Flow Scheduling

Data center network topologies such as fat-tree typically consist of many equal-cost paths between any given pair of hosts. Traditional network forwarding protocols often select a single deterministic path for each pair of source and destination and sometimes protocols such as Equal-Cost Multi-Path (ECMP) routing[8] is used to evenly distribute load on multiple paths. These static mapping of flows to paths do not take into account network utilization and duration of flows. We propose and demonstrate the feasibility of building a dynamic flow scheduling for a given pair of hosts in our multi-rooted tree testbed using ODL APIs.

The key insight is to iteratively redirect a flow of interest (e.g., VM migration traffic) to one of the shortest paths with the lowest load when multiple shortest paths are available between the source and destination. In an SDN-enabled data center, this can be simply performed by a flow

entry setup on the switches along the path which can be configured centrally and then propagated through the entire path. As input, the algorithm receives IP addresses of a given pair of hosts and specifications of the target flow (e.g., protocol and source and destination ports). It then finds multiple paths of equal length between the source and the destination and iteratively measures the average byte rate on the path for the last time interval (e.g., every 15 seconds). Since the target flow happens through one of the shortest paths, we have to make sure that the byte rate of the flow is excluded from the calculation. Thus, the byte rate for the matching flow is deducted from the total calculation. As soon as the shortest path with the lowest average byte rate between the source and the destination is found, appropriate flow rules are pushed into the switches on the path to redirect the flow to this path. In line with this idea, currently, we are working on a more advanced version of dynamic flow scheduling algorithm for efficient migration of virtual machines in in software defined clouds.

To evaluate the impact of the proposed dynamic flow scheduling on bandwidth, using *ipref3* in TCP mode, we generated 10 synthetic and random flows between different hosts in the network. We measured the transmission time and available bandwidth for the flow of interest (given a pair of hosts) with or without enabling dynamic flow scheduling. Figure 5 shows a graphical representation of network topology detected by ODL User Interface (DLUX) along with the labeled given pair of hosts. Table 2 shows the available bandwidth and transmission time for 700MB of data between the given hosts. By enabling dynamic flow scheduling, the transmission time is reduced by 13.6% compared to the static routing method. The average bandwidth was also improved from 35.24Mb/s with no flow management to 40.61Mb/s with dynamic flow scheduling.

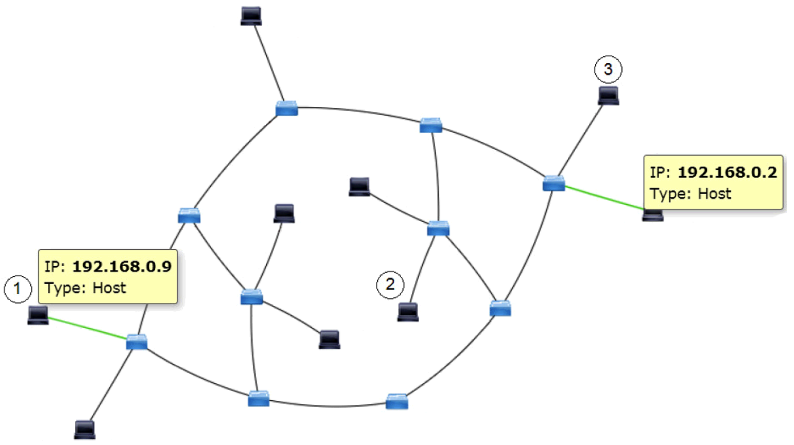

FIGURE 5: Physical Network Topologies detected by OpenDaylight.

Table 2. Transmission time and average bandwidth with/without dynamic flow scheduling.

Method	Transmission Time (s)	Bandwidth (Mb/s)
Without Dynamic Flow Scheduling	159	35.24
With Dynamic Flow Scheduling	137	40.81

The performance gain of the proposed dynamic scheduling is heavily dependent on the rates and duration of the flows in the network. This experiment demonstrates the feasibility of building a working prototype of dynamic flow scheduling in the CLOUDS-Pi platform.

Virtual Machine Management

Live migration is one of the core concepts in modern data centers which allows moving a running VM between physical hosts with no impact on the VM availability. While VMs in data centers are often migrated between hosts to reduce energy consumption or for the maintenance

purposes, live VM migration also provides an opportunity to enhance link utilization and reduce network-wide communication cost. This can be done by relocating communicating VMs to hosts with near vicinity and less number of connecting links in the higher layers of the data center network topology.

We measure the bandwidth between two communicating VMs running on physical hosts (Hosts *192.168.0.2* and *192.168.0.9* in Figure 5) connected through core switches. We run experiments by moving these two VMs closer to each other in the network topology. First, we perform a VM migration to remove any core switches on the connecting shortest path and then another migration to remove both core and aggregation switches. Meanwhile, we investigate the impact of migrations on the available bandwidth.

During the experiment, random and synthetic background traffic is generated between all hosts in the network both periodically and continuously. The graph in Figure 6 shows available bandwidth from the source (Marked as 1) to the destination VM (Marked as 2) measured with *iperf3* tool. Before the first migration (time 0 to 70), the source VM is placed on a different pod from the destination VM. As shown in the graph, during this period the average bandwidth is roughly 71 Mb/s and fluctuates considerably due to the background traffic generated by other hosts. After 70 seconds, a live VM migration is performed to move the source VM to the host marked as 2 in Figure 5 and connected to a different access switch in the same pod of the destination VM. During the live migration process, the VMs' networking becomes unavailable for a short period. After this migration, bandwidth becomes less fluctuating and increases to 79Mb/s on average because less number of background flows are conflicting on the traffic of source/destination VM pair. The last migration is done after 170 seconds of the experiment. The source VM is migrated to the host marked as 3 which is connected to the same access switch as the destination VM. With this migration, bandwidth becomes more stable and rises to the average of 86Mb/s where least background traffic affects the networking performance of the VMs.

The experiment demonstrates the feasibility of building a prototype system allowing network-wide communication minimization in a cloud data center and open up the research possibility on joint VM and traffic consolidation. Other interesting research ideas can be “How to select the destination for a VM migration to get the best bandwidth or performance after migration”, or “how to jointly consolidate VMs and traffic in data centers[15]?”, or “How to place Virtualized Network Function (VNF) instances in a data center supporting Service Function Chaining (SFC)[13]?”.

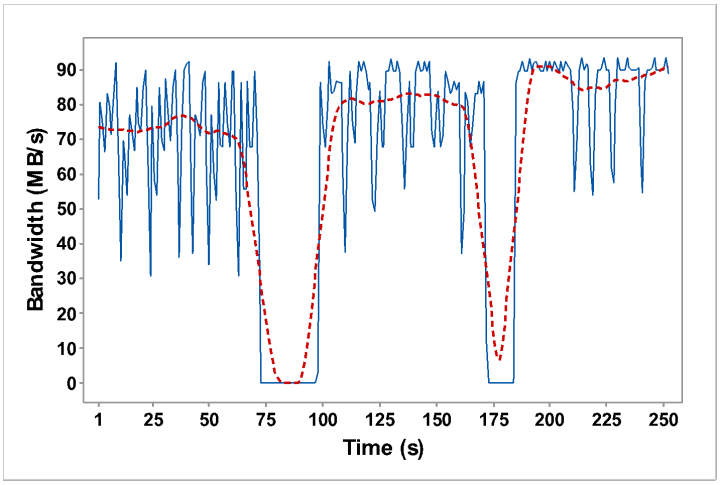

FIGURE 6: Impact of VM live migration on the bandwidth of communicating VM pair.

Discussions and Future Directions

Our results demonstrate the potential of the CLOUDS-Pi platform in enabling investigation on different aspects of SDN-enabled cloud computing. One of the main benefits of CLOUDS-Pi for conducting research on SDN in cloud computing compared to other prototyping methods or emulators such as Mininet is the option of VM management. VM management and the VM migration possibility, in particular, is an essential aspect of cloud computing which allows for VM consolidation to reduce power consumption and increase cost efficiency. Thanks to OpenStack setup in our platform, we can jointly leverage virtualization capabilities and SDN for performing research on VM and traffic consolidation[15].

Another benefit of CLOUDS-Pi is the possibility of conducting innovative research in traffic control and network management with high accuracy and performance fidelity. As shown in the first use case, using ODL northbound APIs and its flow scheduling feature, we can investigate dynamic traffic engineering and load balancing for reducing network congestion with a level of confidence that would otherwise be beyond our reach.

Apart from network management, other research directions that can be followed on our testbed platform include but not limited to Security (e.g., Network intrusion detection), Service Function Chaining, and application-specific networking.

The current setup of CLOUDS-Pi has limited number of resources. The number of switches and hosts used in building the CLOUDS-Pi platform is far from enough to test the scalability of approaches that need to be deployed in cloud data centers hosting tens of thousands of servers and thousands of network switches. In addition, since we are using USB 2.0 ports on Raspberry Pis with a nominal bandwidth of 480 Mb/s and *100 Mb/s USB to Ethernet adapters* as switch ports, our network bandwidth, even though suitable for the scale of our testbed, is much lower than gigabit or terabit networks used in real-world cloud data centers. Although CLOUDS-Pi offers a suitable environment to carry out empirical research into SDN and cloud computing without the expenditure of full-size testbeds, we recommend the use of simulators such as CloudSimSDN[16] to evaluate the scalability of policies.

Besides our testbed limitations, currently, SDN itself faces some challenges such as scalability, reliability, and security that hinder its performance and application. The logically centralized control in SDN causes scalability concerns and especially the controller scalability is one of the problems that need particular attention. The scalability issue of SDN becomes more evident in large-scale networks such as cloud data centers compared to small networks.[9] Controller distribution and the use of east/west APIs is one way to overcome computational load on the controller but it brings consistency and synchronization problems into the picture.[9] It also requires standard protocols for an interoperable state exchange between controllers of different types.

The centralized control plane of SDN, for example, the single host deployment of ODL controller in our testbed, has a critical reliability risk, such as a single point failure. To tackle this problem, running backup controllers is a commonly used mechanism. However, defining the optimal number of controllers and the best locations for the primary and backup controllers is challenging. Synchronization and concurrency issues also need to be addressed in this regard.[1]

SDN has two fundamental security issues which are[14]: 1) the potential for single point of attack and failure and 2) the southbound API connecting the controller and data-forwarding devices is vulnerable to interception and attacks on communications. Even though TLS/SSL encryption techniques can be used to secure communications between the controller(s) and OpenFlow switches, the configuration is very complicated, and many vendors do not provide support of TLS in their OpenFlow switches by default. SDN security is critical since threats can degrade the availability, performance, and integrity of the network. While many efforts are currently being made to address SDN security issues, this topic is expected to draw increasing amounts of attention in coming years.

CONCLUSION

We presented CLOUDS-Pi, a low-cost testbed environment for SDN-enabled cloud computing. All aspects of our platform from its overall architecture to particular software choices are explained. We also demonstrated how economical computers such as Raspberry Pis could be used to mimic a network of OpenFlow switches in SDN-enabled cloud data center in scale. In order to evaluate our testbed, two use cases for dynamic flow scheduling and virtual machine management are identified. We discussed benefits and limitations of CLOUDS-Pi as a research platform for different aspects of SDN in clouds and proposed future research directions. Our work demonstrated the potential of CLOUDS-Pi platform for conducting practical research and prototyping SDN-enabled cloud computing environments.

REFERENCES

1. W. Han, M. Dong, K. Ota, J. Wu, J. Li and G. Li, "SD-OPTS: Software-Defined On-Path Time Synchronization for Information-Centric Smart Grid," The IEEE Global Communications Conference, Singapore, pages 1-6, IEEE, 2017.
2. I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou. A roadmap for traffic engineering in SDN-OpenFlow networks. Computer Networks, 71 (Supplement C):1-30, 2014.
3. H. Yu, K. Li, H. Qi, W. Li and X. Tao. Zebra: An East-West Control Framework for SDN Controllers. The 44th International Conference on Parallel Processing, Beijing, pages 610-618, IEEE, 2015.
4. A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. R. Kompella. Elasticon: an elastic distributed sdn controller. The tenth ACM/IEEE symposium on Architectures for networking and communications systems, pages 17–28, 2014.
5. M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. SIGCOMM Comput. Commun. Rev., 38(4):63-74, 2008.
6. K. Alwasel, Y. Li, P. P. Jayaraman, S. Garg, R. N. Calheiros, and R. Ranjan. Programming SDN-native big data applications: Research gap analysis. IEEE Cloud Computing, 4(5):62-71, Sept 2017.
7. M. Gupta, J. Sommers, and P. Barford. Fast, accurate simulation for SDN prototyping. The second ACM SIGCOMM workshop on Hot topics in software defined networking, pages 31-36. ACM, 2013.
8. C. E. Hopps. Analysis of an equal-cost multi-path algorithm. RFC2992, 2000.
9. M. Karakus and A. Durresi. A survey: Control plane scalability issues and approaches in software-defined networking. Computer Networks, 112:279-293, 2017.
10. H. Kim, J. Kim, and Y.-B. Ko. Developing a cost-effective OpenFlow testbed for small-scale software defined networking. The16th International Conference on Advanced Communication Technology, pages 758-761, Feb 2014.
11. B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. The 9th ACM SIGCOMM Workshop on Hot Topics in Networks, page 19. ACM, 2010.
12. D. S. Linthicum. Software-defined networks meet cloud computing. IEEE Cloud Computing, 3(3):8-10, May 2016.
13. A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz. Service function chaining in next-generation networks: State of the art and research challenges. IEEE Communications Magazine, 55(2):216 - 223, 2017.
14. S. Scott-Hayward, G. O'Callaghan, and S. Sezer. SDN security: A survey. The IEEE SDN for Future Networks and Services, pages 1-7, Nov 2013.
15. J. Son, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya. SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers. IEEE Transactions on Sustainable Computing, 2(2):76-89, 2017.
16. J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya. CloudSimSDN: Modeling and simulation of software-defined cloud data centers. The 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pages 475-484, May 2015.
17. F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros. The Glasgow Raspberry Pi Cloud: A scale model for cloud computing infrastructures. The 33rd IEEE International Conference on Distributed Computing Systems Workshops, pages 108-112, July 2013.

ABOUT THE AUTHORS

Adel Nadjaran Toosi is a Lecturer in Faculty of Information Technology at Monash University, Australia. He was with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne. He also completed his PhD in Computer Science and Software Engineering at the University of Melbourne in 2015. Adel is an active member of ACM and IEEE. Currently, he is working on software-defined networking in clouds. For more information, please visit his homepage: http://adelnadjarantoosi.info.

Jungmin Son is a Research Fellow with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. His research interests include SDN-enabled cloud computing, resource provisioning, scheduling, and energy efficient data centers.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. Recently, Dr. Buyya is recognized as a "Web of Science Highly Cited Researcher" in 2016 and 2017 by Thomson Reuters, a Fellow of IEEE, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. For further information, please visit his cyberhome: www.buyya.com.