CLOUDS-Pi: A Low-Cost Raspberry-Pi-Based Micro Datacenter for Software-Defined Cloud Computing

Adel Nadjaran Toosi
Monash University

Jungmin Son
University of Melbourne

Rajkumar Buyya
University of Melbourne

Software-defined networking (SDN) is rapidly transforming the networking ecosystem of cloud-computing datacenters. However, replicating SDN-enabled cloud infrastructures to conduct practical research in this domain requires a great deal of effort and capital expenditure. This article presents the CLOUDS-Pi platform, a small-scale cloud datacenter for doing research on software-defined clouds. As part of it, Open vSwitch is integrated with Raspberry-Pis, low-cost embedded computers, to build up a network of OpenFlow switches. The article provides two use cases and reports on validation and performance evaluation. It also discusses the 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 toward 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 spread beyond ordinary networks and is even suggested to address the challenges of smart-grid systems.[1]

Cloud computing is a successful computing paradigm that delivers computing resources residing in providers' datacenters as a service over the Internet on a pay-as-you-go basis. With the growing adoption of cloud computing, datacenters hosting cloud services are rapidly expanding their

Copublished by the IEEE CS and IEEE ComSoc
2325-6095/18/$33.00 USD ©2018 IEEE

sizes and increasing in number. Therefore, resource management in clouds' large-scale infrastructure becomes a challenge.

In the meantime, SDN is increasingly being accepted as the technology for a new generation of networks in cloud datacenters, where the need exists for efficient management of large multitenant networks of dynamic, ever-changing environments. In fact, SDN not only reduces the complexity seen in today's cloud datacenter networks but also helps cloud providers manage network services from a central management point.

The fruitful combination and deployment of cloud computing and SDN require significant innovation and research to fuse them together.[2] However, evaluation and experimentation of SDN-based applications for cloud environments present many challenges in terms of complexity, scaling, accuracy, and efficiency.[3] The major classes of performance evaluation techniques for these methods can be categorized as analytical modeling, simulation, emulation, and measurement.

If we exclude analytical modeling owing to its limitations and complexity, simulation tools (e.g., CloudSimSDN[4]) play a significant role in the performance evaluation of these methods owing to their flexibility and affordability. However, simulations always present imperfect models and are limited in their ability to reproduce real-world software-defined clouds (SDCs).

Emulation techniques, on the other hand, seem to be the method of choice for achieving a detailed understanding of the operation of SDCs. Software emulators (e.g., Mininet[5]) 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.[6] Rapid and affordable prototyping of algorithmic advances in SDCs is challenging, and significant capital expenditure is required to replicate practical implementations and performance measurements.

With these issues in mind, this article puts together all the elements to build a low-cost micro software-defined datacenter by leveraging off-the-shelf hardware and open source software. We propose a system architecture for constructing a testbed and micro datacenter for researching SDCs. 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 datacenter network. To make a network switch out of a 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 the following:

- We propose a system architecture and design to build a low-cost experimental testbed or infrastructure for conducting practical research in the domain of SDCs.
- We present the challenges, detailed requirements, all the required elements, and our expriences building CLOUDS-Pi, a testbed and micro datacenter built in the CLOUDS (Cloud Computing and Distributed Systems) Laboratory at the University of Melbourne 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 the benefits and limitations of such a testbed.

SYSTEM ARCHITECTURE

In this section, we present the CLOUDS-Pi platform along with the physical-infrastructure setup and utilized software. In contrast to other work in this domain, such as Zebra[7] or Elasticcon,[8] that particularly focuses 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 SDCs.

Physical Infrastructure

The primary aim of our small cloud datacenter is to provide an economical testbed for conducting research in 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 computational resources.[9] Our platform comprises a set of nine heterogeneous machines with the specifications shown in Table 1. To keep up with common practice, we use separate networks for management, data, and control.

Table 1. Specifications of machines in CLOUDS-Pi.

Machines	CPU	No. of cores	Memory	Storage
3 × IBM x3500 M4	Intel Xeon E5-2620 @ 2.00 GHz	12	64 Gbytes (4 × 16 Gbyte DDR3, 1,333 MHz)	2.9 Tbytes
4 × IBM x3200 M3	Intel Xeon X3460 @ 2.80 GHz	4	16 Gbytes (4 × 4 Gbyte DDR3, 1,333 MHz)	199 Gbytes
2 × Dell OptiPlex 990	Intel Core i7-2600 @ 3.40 GHz	4	8 Gbytes (2 × 4 Gbyte DDR3, 1,333 MHz)	399 Gbytes

The management network is used by OpenStack, our deployed cloud OS, for internal communication between its components and resource management. It constitutes a 16-port 10/100 Mbps Ethernet switch (NetGear Model FS516) connecting the hosts and OpenStack controller.

The data network is used for data communication among VMs deployed within the cloud environment and for providing Internet access to them through the gateway host. This network is a 100 Mbps fat-tree-like network[10] built on top of 10 Raspberry Pis (Pi 3 Model B) with OVS integrated, each playing a role of a 4-port switch with an external port for control. Each Raspberry Pi 3 Model B has four USB 2.0 ports and only one Ethernet interface. Therefore, we used TP-Link UE200 100 Mbps USB-to-Ethernet adapters to add four extra Ethernet interfaces to the Raspberry Pi switch.

The control network connects control ports on the Raspberry Pi switches to the SDN controller and transfers OpenFlow packets between the controller and the switches.

To monitor the 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 and 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 datacenter setup, including the network topology for the management, data, and control networks. Figure 2 depicts the CLOUDS-Pi platform.

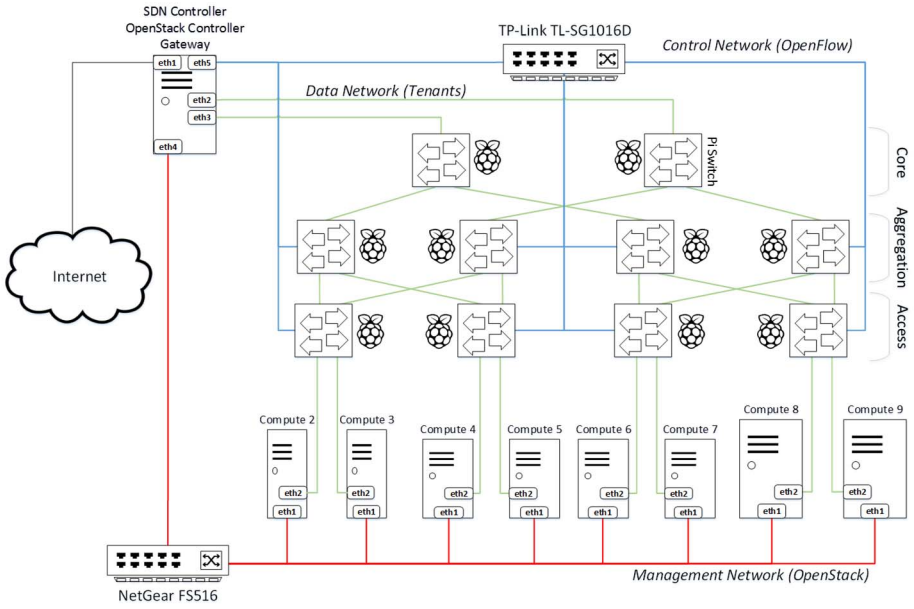

Figure 1. The system architecture of CLOUDS-Pi.

Figure 2. The CLOUDS-Pi platform. The top-left image shows a Raspberry Pi in an enclosure with four TP-Link USB-to-Ethernet adapters connected to available USB ports. The bottom-left image illustrates the 10 Raspberry Pi switches in a fat-tree-like topology. The middle image shows CLOUDS-Pi's tower servers in the racks. The right image shows the two Eaton ePDUS (enclosure power distribution units) for power monitoring and management.

Software

We installed the CentOS 7.0 Linux distribution as the host OS 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 (an IBM X3500) as the controller node. All other nodes play the same role of computation hosts in the design.

In addition, we created NAT (network address translation) forwarding rules on the controller, using Linux iptables to enable all other nodes to connect to the Internet. In this way, the controller node is configured as the default gateway for external network access (Internet access) for all other nodes and OpenStack VMs. We also set up an 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 datacenter network (see Figure 3).

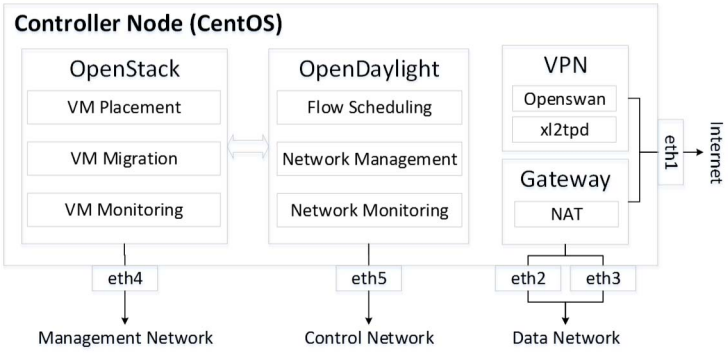

Figure 3. The software stack on the controller node. VM = virtual machine, VPN = virtual private network, eth = Ethernet interface, and NAT = network address translation.

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 the control network (see Figure 3). Every Raspberry Pi in our setup uses a Debian-based Linux OS of Raspbian Version 8 (Jessie) and has 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's 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, the *Status Visualizer* is a tool implemented to visualize traffic flows among hosts and VMs 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 the used bandwidth. Figure 4 depicts the screenshot of the Status Visualizer web UI showing sample traffic in CLOUDS-Pi between groups of VMs.

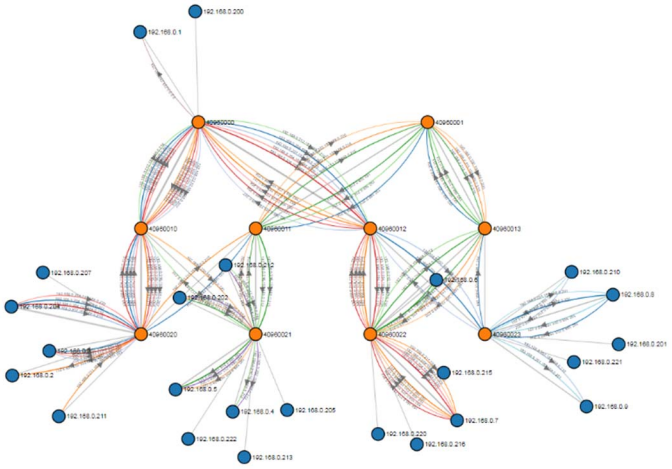

Figure 4. Sample network traffic in CLOUDS-Pi visualized by the Status Visualizer.

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 multirooted 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.

Dynamic Flow Scheduling

Datacenter network topologies such as fat trees 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 multipath (ECMP) routing[11] are used to evenly distribute the load on multiple paths. This static mapping of flows to paths does not take into account network utilization and the duration of flows. We propose and demonstrate the feasibility of building dynamic flow scheduling for a given pair of hosts in our multirooted-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 datacenter, this can be simply performed by a flow entry setup on the switches along the path, which can be configured centrally and then propagated throughout the entire path.

As input, the algorithm receives the IP addresses of a given pair of hosts and specifications of the target flow (e.g., the 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, we are working on a more advanced version of a dynamic-flow-scheduling algorithm for efficient migration of VMs in SDCs.

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 the network topology detected by the ODL User Interface (DLUX), along with the labeled given pair of hosts. Table 2 shows the available bandwidth and transmission time for 700 Mbytes of data between the given hosts. Enabling dynamic flow scheduling reduced the transmission time by 13.6% compared to the static-routing method. The average bandwidth also improved from 35.24 Mbps with no flow management to 40.61 Mbps with dynamic flow scheduling.

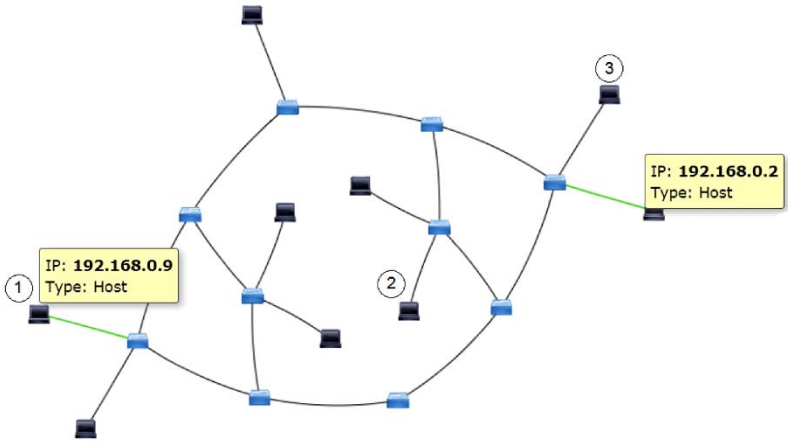

Figure 5. Physical-network topologies detected by OpenDaylight.

Table 2. The transmission time and average bandwidth with and without dynamic flow scheduling.

Method	Transmission time (s)	Bandwidth (Mbps)
Without dynamic flow scheduling	159	35.24
With dynamic flow scheduling	137	40.81

The performance gain of the proposed dynamic scheduling was 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, one of the core concepts in modern datacenters, allows moving a running VM between physical hosts with no impact on the VM's availability. While VMs in datacenters are often migrated between hosts to reduce energy consumption or for maintenance purposes, live VM migration also provides an opportunity to enhance link utilization and reduce network-wide communication costs. This can be done by relocating communicating VMs to nearby hosts with fewer connecting links in the higher layers of the datacenter network topology.

We measured 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 ran experiments by moving these two VMs closer to each other in the network topology. First, we performed a VM migration to remove any core switches on the connecting shortest path and then another migration to remove both the core and aggregation switches. Meanwhile, we investigated the impact of migrations on the available bandwidth.

During the experiment, random and synthetic background traffic was generated between all hosts in the network, both periodically and continuously. The graph in Figure 6 shows the available bandwidth from the source VM (labeled 1 in Figure 5) to the destination VM (labeled 2 in Figure 5), measured with the iperf3 tool. Before the first migration (time 0 to 70), the source VM was placed on a different pod from the destination VM. As shown in the graph, during this period the average bandwidth was roughly 71 Mbps and fluctuated considerably owing to the background traffic generated by other hosts.

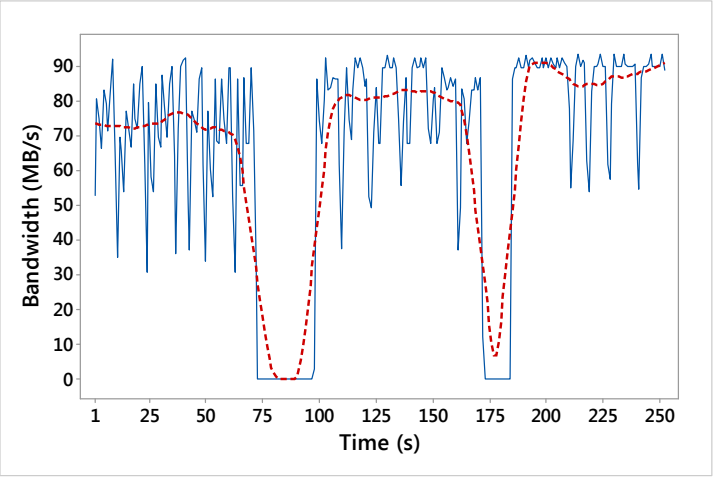

Figure 6. The impact of VM live migration on the bandwidth of a communicating VM pair.

After 70 seconds, a live VM migration was performed to move the source VM to the host labeled 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 became unavailable for a short period. After this migration, the bandwidth fluctuated less and increased to 79 Mbps on average because fewer background flows conflicted with the traffic of the source-and-destination VM pair.

The last migration was done after 170 seconds of the experiment. The source VM was migrated to the host labeled 3 in Figure 5, which was connected to the same access switch as the destination VM. With this migration, bandwidth became more stable and rose to an average of 86 Mbps, where the least background traffic affected the networking performance of the VMs.

The experiment demonstrates the feasibility of building a prototype system allowing network-wide communication minimization in a cloud datacenter. It also opens up the possibility of research on joint VM and traffic consolidation. Other interesting research ideas include

- how to select the destination for a VM migration to get the best bandwidth or performance after migration,
- how to jointly consolidate VMs and traffic in datacenters,[12] and
- how to place virtual network function (VNF) instances in a datacenter supporting service function chaining (SFC).[13]

Discussion and Future Directions

Our results demonstrate the potential of the CLOUDS-Pi platform in enabling investigation of 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, in particular, the possibility of VM migration are essential aspects of cloud computing that allow for VM consolidation to reduce power consumption and increase cost efficiency. Thanks to the OpenStack setup in our platform, we can jointly leverage virtualization capabilities and SDN for performing research on VM and traffic consolidation.[12]

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's 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, research directions that can be followed on our testbed platform include, but are not limited to, security (e.g., network intrusion detection), SFC, and application-specific networking.

The current setup of CLOUDS-Pi has a 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 datacenters 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 Mbps and 100 Mbps 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 datacenters. Although CLOUDS-Pi offers a suitable environment to carry out empirical research on SDN and cloud computing without the expenditure of full-size testbeds, we recommend the use of simulators such as CloudSimSDN[4] to evaluate the scalability of policies.

Besides our testbed limitations, SDN itself currently 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 controller scalability especially is one of the problems that need particular attention. The scalability issue of SDN becomes more evident in large-scale networks such as cloud datacenters, compared to small networks.[14] Controller distribution with the use of east–west APIs is one way to overcome computational loads on the controller, but it brings consistency and synchronization problems into the picture.[14] 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 the ODL controller in our testbed—has a critical reliability risk, such as a single point of failure. To tackle this problem, running backup controllers is a common approach. 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.[15]

SDN has two fundamental security issues:[16]

- The potential exists for a single point of attack and failure.
- The southbound API connecting the controller and data-forwarding devices is vulnerable to interception and attacks on communication.

Even though TLS/SSL encryption techniques can be used to secure communication between controllers and OpenFlow switches, the configuration is very complicated, and many vendors do not provide support for 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 increased attention in the 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, were explained. We also demonstrated how economical computers such as Raspberry Pis could be used to mimic a network of OpenFlow switches in an SDN-enabled cloud datacenter on a small scale. In order to evaluate our testbed, two use cases for dynamic flow scheduling and VM management were identified. We discussed the benefits and limitations of CLOUDS-Pi as a research platform for different aspects of SDN in clouds, and we proposed future research directions. Our work demonstrates the potential of the CLOUDS-Pi platform for conducting practical research and prototyping SDN-enabled cloud-computing environments.

REFERENCES

1. W. Han et al., "SD-OPTS: Software-Defined On-Path Time Synchronization for Information-Centric Smart Grid," *Proceedings of the 2017 IEEE Global Communications Conference* (GLOBECOM 2017), 2017, pp. 1–6.
2. D.S. Linthicum, "Software-defined networks meet cloud computing," *IEEE Cloud Computing*, vol. 3, no. 3, 2016, pp. 8–10.
3. M. Gupta, J. Sommers, and P. Barford, "Fast, accurate simulation for SDN prototyping," *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (HotSDN 13), 2013, pp. 31–36.
4. J. Son et al., "CloudSimSDN: Modeling and simulation of software-defined cloud datacenters," *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (CCGrid 15), 2015, pp. 475–484.
5. B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (HotNets 10), 2010, p. 19.
6. H. Kim, J. Kim, and Y.-B. Ko, "Developing a cost-effective OpenFlow testbed for small-scale software defined networking," *Proceedings of the 16th IEEE International Conference on Advanced Communication Technology*, 2014, pp. 758–761.
7. H. Yu et al., "Zebra: An East-West Control Framework for SDN Controllers," *Proceedings of the 44th IEEE International Conference on Parallel Processing*, 2015, pp. 610–618.
8. A. Dixit et al., "Elasticon: An elastic distributed SDN controller," *Proceedings of the 10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2014, pp. 17–28.
9. F.P. Tso et al., "The Glasgow Raspberry Pi Cloud: A scale model for cloud computing infrastructures," *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems Workshops*, 2013, pp. 108–112.
10. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity datacenter network architecture," *ACM SIGCOMM Computer Communication Review* (SIGCOMM 08), vol. 38, no. 4, 2008, pp. 63–74.
11. C.E. Hopps, "Analysis of an equal-cost multi-path algorithm," RFC2992, ACM, 2000.
12. J. Son et al., "SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud datacenters," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, 2017, pp. 76–89.
13. A.M. Medhat et al., "Service function chaining in next-generation networks: State of the art and research challenges," *IEEE Communications Magazine*, vol. 55, no. 2, 2017, pp. 216–223.
14. M. Karakus and A. Durresi, "Control plane scalability issues and approaches in software-defined networking," *Computer Networks*, vol. 112, 2017, pp. 279–293.
15. I.F. Akyildiz et al., "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, no. C, 2014, pp. 1–3.
16. S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," *Proceedings of the 2013 IEEE Conference on SDN for Future Networks and Services* (SDN4FNS 13), 2013, pp. 1–7.

ABOUT THE AUTHORS

Adel Nadjaran Toosi is a lecturer in Monash University's Faculty of Information Technology. He currently works on software-defined networking (SDN) in clouds. He previously was with the University of Melbourne's Cloud Computing and Distributed Systems (CLOUDS) Laboratory. Toosi received a PhD in computer science and software engineering from the University of Melbourne. He's a member of ACM and IEEE. Contact him at adel.n.toosi@monash.edu; http://adelnadjarantoosi.info.

Jungmin Son is a research fellow at the University of Melbourne's Cloud Computing and Distributed Systems (CLOUDS) Laboratory. His research interests include SDN-enabled

cloud computing, resource provisioning, scheduling, and energy-efficient datacenters. Son received a PhD from the University of Melbourne's School of Computing and Information Systems. Contact him at json1@unimelb.edu.au.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and the director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne. His research interests are cloud computing, distributed systems, and fog and edge computing. Buyya received a PhD in computer science and software engineering from Monash University. He has been a Web of Science Highly Cited Researcher and a Scopus Researcher of the Year and has received an Excellence in Innovative Research Award from Elsevier. He's an IEEE Fellow. Contact him at rbuyya@unimelb.edu.au; www.buyya.com.