# Software-Defined Infrastructure and the SAVI Testbed

Joon-Myung Kang[1], Thomas Lin[2], Hadi Bannazadeh[2], and Alberto Leon-Garcia[2]

[1] HP Labs., Palo Alto, CA, 94304, USA,
joon-myung.kang@hp.com
[2] Dept. of Electrical and Computer Engineering, University of Toronto
Toronto, ON, M5S 3G4, Canada
{t.lin,hadi.bannazadeh,alberto.leongarcia}@utoronto.ca

**Abstract.** In this paper we consider Software-Defined Infrastructure (SDI), a new concept for integrated control and management of converged heterogeneous resources. SDI enables programmability of infrastructure by enabling the support of cloud-based applications, customized network functions, and hybrid combinations of these. We motivate SDI in the context of a multi-tier cloud that includes massive-scale datacenters as well as a smart converged network edge. In SDI, a centralized SDI manager controls converged heterogeneous resources (i.e., computing, programmable hardware, and networking resources) using virtualization and a topology manager that provides the status of all resources and their connectivity. We discuss the design and implementation of SDI in the context of the Canadian SAVI testbed. We describe the current deployment of the SAVI testbed and applications that are currently supported in the testbed.

**Key words:** virtualization, cloud computing, software defined networking, resource management

## 1 Introduction

The delivery of content and software applications is being revolutionized by application platforms that encompass massive datacenters, the Internet, and smart phones. Cloud computing, typically in very large remote datacenters, provide unprecedented flexibility and economies of scale in the support of applications. Software-defined networking (SDN) allows fine-grained control of application flows. Together cloud computing and SDN promise a future open marketplace where applications can be readily and rapidly programmed on a converged infrastructure. Major collaborative open source efforts are helping advance these two technologies, OpenStack [6] for cloud computing and OpenFlow [1] for SDN.

We view the cloud as being multi-tier in nature, with massive remote datacenters in one tier, and converged smart edge nodes closer to the users. Computing and networking resources in the smart edge are essential to support applications with low-latency requirements, to execute security functions, and to promote efficient content distribution through local caching resources.

The location of the smart edge is roughly where telecom service providers are placed. Therefore it is natural that the design of the smart edge should consider the challenges of the service provider. The overarching challenge today is the need to invest huge capital expenditures to increase wireless capacity to accommodate higher traffic, while coping with slower revenue growth from competition and customer expectation for continual sustained improvement. We believe that virtualization can play a role in addressing these twin challenges.

The remote massive datacenter leverages virtualization of computing and networking resources to deliver flexibility and compelling economies of scale. In contrast, the smart edge is significantly smaller in scale and much more heterogeneous in its resources. The smart edge especially when defined to include wireless and wired access networks include nonconventional computing resources, namely FPGAs, network processors, ASICs for signal processing within purpose-built boxes. We believe that flexibility and economies of scale can be attained in the smart edge through the virtualization of computing, networking, and non-conventional computing resources and the introduction of control and management systems for converged resources.

Until recently, control and management approaches have focused on the separate management of different infrastructure resources. For example, cloud controllers such as OpenStack provide cloud resource provisioning, while network controllers such as an OpenFlow and other SDN controllers provide network control. In the smart edge, an integrated management and control system for converged network and generalized computing resources can be more effective in providing flexibility and performance in a cost-effective manner. Open interfaces for controlling and managing these shared heterogeneous resources can provide software programmability for dynamically deploying new functionality. In addition, advanced monitoring and measurement techniques and user access to infrastructure information can provide customized resource allocation or networking. Therefore, we need a "software-defined infrastrastructure (SDI) to satisfy their requirements beyond SDN and VMs.

In [2] we introduced the notion of SDI and in [3] we introduced the initial design of the control and management of the SAVI testbed. In this paper we first present the SDI architecture for designing a testbed for future applications and services, focusing specifically on the SDI manager and its associated topology manager. Next, we present the current design and implementation of the SAVI testbed and its control and management system for converged heterogeneous resources based on the SDI architecture. We describe the SAVI Testbed which has been deployed across much of Canada and used to demonstrate the management of physical and virtual resources. We also describe the hands-on workshop provided to SAVI researchers to promote the usage of the testbed. We describe a tutorial to introduce users to a configuration management service, as well as to the SDI manager to query and manage the physical and virtual network infrastructure in the SAVI testbed.

The paper is organized as follows. Section 2 describes a high-level system architecture of SDI using major components. Section 3 presents SAVI cluster configuration and heterogeneous resources and a design of SAVI testbed resource control and management system. The current SAVI testbed deployment

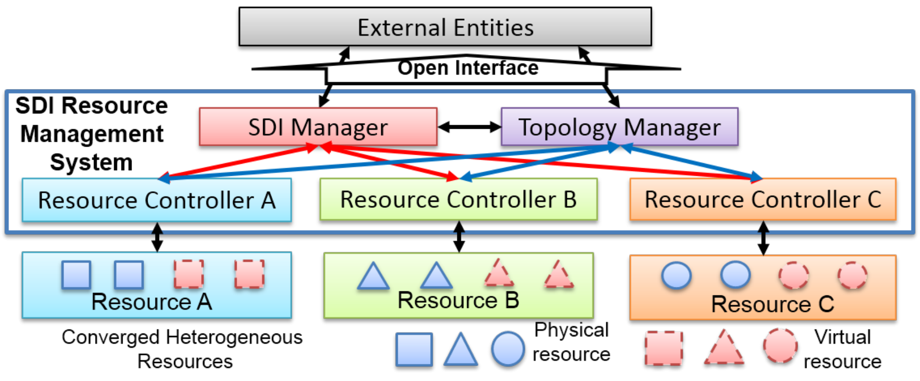

**Fig. 1.** A system architecture for SDI resource management

and status are presented in Section 4. Finally, conclusion and future work are presented in Section 5.

## 2 Software-Defined Infrastructure

In this section, we define SDI and present an SDI resource management architecture for the converged heterogeneous resources. In SDI, "Software-Defined means providing open interfaces to: control and manage converged heterogeneous resources in different types of infrastructures for software programmability; and give an access to infrastructure resource information such as topology, usage data, etc. We design the SDI architecture to support those requirements.

Fig. 1 shows a high-level architecture of the SDI Resource Management System (RMS), in which an SDI manager can control and manage a resource of type A, B, and C using a corresponding resource controller A, B, or C, respectively. External entities obtain virtual resources in the converged heterogeneous resources via the SDI RMS through "Open Interfaces. The converged heterogeneous resources are composed of virtual resources and physical resources. Virtual resources include any resource virtualized on physical resources, such as virtual machines. Physical resources include any resource that can be abstracted or virtualized, such as computing servers, storage, network resources (routers or switches), and reconfigurable hardware resources.

The SDI RMS provides resource management functions for the converged heterogeneous resources to the external entities. These functions include provisioning, registry/configuration management, virtualization, allocation/scheduling, migration/scaling, monitoring/measurement, load balancing, energy management, fault management, performance management (delay, loss, etc.), and security management (authentication, policy, role, etc.). The external entities can be applications, users (service developers or providers), and high-level management systems.

The SDI manager performs coordinated and integrated resource management for converged heterogeneous resources through a resource controller and the topology manager. The SDI manager performs major integrated resource management functions based on resource topology information provided by the topology manager. Each resource controller is responsible for taking the high-level user descriptions and managing the resources of a given type. The topology manager maintains a list of the resources, their relationships, and monitoring and measurement data of each resource. Furthermore, the topology manager provides up-to-date resource information to the SDI manager for infrastructure-state-aware resource management. Examples of the integrated resource management functions that can be performed by the SDI manager include: fault tolerance, green networking (energy efficient and/or low-carbon emitting), path optimization, resource scheduling optimization, network-aware VM replacement, QoS support, real-time network monitoring, and flexible diagnostics.

## 3 SAVI Testbed based on SDI Concept

The Smart Applications on Virtual Infrastructure (SAVI) project was established to investigate future application platforms designed for applications enablement [3]. We have developed a SAVI testbed system for controlling and managing converged virtual resources focused on computing and networking. In previous work [3], we extended Virtual Application on Network Infrastructure (VANI) [5] for supporting non-conventional computing resources. In this section we extend the previous SAVI testbed management system to one based on an SDI architecture that provides a uniform abstraction for heterogeneous resources. First, we present the SAVI cluster configuration and its heterogeneous resources. Second, we present a high-level design of our Control and Management system based on SDI. Third, we present an SDI manager which is a core component for integrated resource management. Finally, we present the topology manager which provides status of SAVI testbed nodes and resources.

### 3.1 SAVI Cluster

SAVI explores a multi-tier cloud that includes massive core datacenters, smart edge nodes, and access networks, wherein all resources are virtualized. SAVI has designed a node cluster that can provide virtualized and physical computing and networking resources, including heterogeneous resources. We anticipate that the Smart Edge will leverage these heterogeneous resources to provide greater service flexibility, improved resource utilization and cost efficiencies. A SAVI cluster provides heterogeneous resources interconnected by a 10GE OpenFlow fabric. SAVI has developed an approach to allow heterogeneous resources to be managed with OpenStack [6]. Currently, SAVI clusters include Intel Xeon servers, storage, OpenFlow switches, GPUs, NetFPGAs, Alteras DE5-Net and ATOM servers.

### 3.2 High-Level Control and Management System Design

Fig. 2 shows the design of a control and management system for a SAVI node based on the SDI architecture to manage cloud and networking resources. A

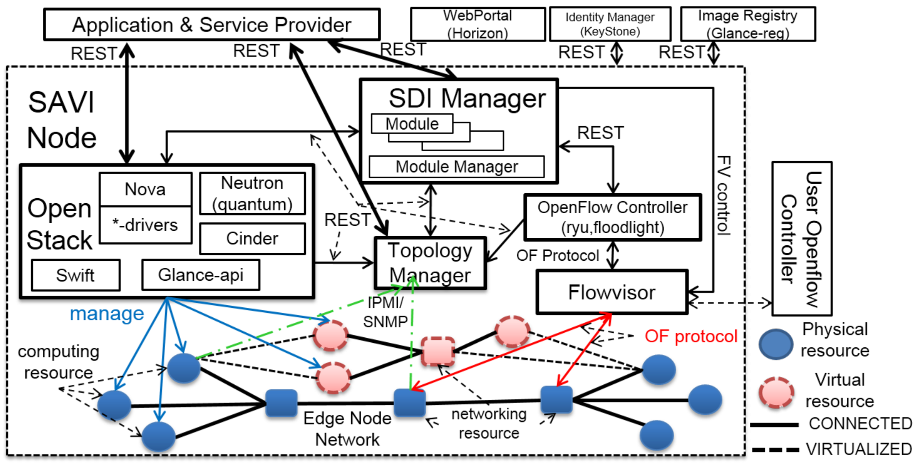

**Fig. 2.** Design of a control and management system for SAVI node [2]

SAVI node controls and manages virtual resources using OpenStack and Open-Flow controller. In the Edge node (converged) network, a variety of heterogeneous computing and networking resources are available as shown in Fig. 2. The SDI manager controls and manages virtual computing resources by virtualizing physical computing resources using OpenStack. The OpenFlow controller is used for controlling networking resources. The OpenFlow controller receives all events from the OpenFlow switches and creates a flow table including actions. The SDI manager performs all management functions based on data provided by the OpenStack and the OpenFlow controller, and it determines appropriate actions for computing and networking resources using management modules inside.

The SDI manager has a module manager to manage specific functional modules such as a scheduling module, a networking control module, a fault-tolerant management module, or a green networking module. Details of each module are out of the scope of this paper. As in SDN, we have separated the data and control planes in the SAVI node. The OpenStack and OpenFlow controller are modules for communicating directly with computing and networking resources. The SDI manager in Fig. 2 is responsible for control and management tasks. The topology manager collects cloud computing resource information using OpenStack and networking resource information using OpenFlow. In addition, the topology manager can collect system information from the physical resources using IPMI and SNMP. Application and service providers can access not only control and management functions but also topology information through RESTful APIs which is a kind of open interfaces.

In the SAVI testbed, we have used and extended the following projects from OpenStack: 1) Keystone for Identity management, 2) Nova for Compute and a cloud computing fabric controller, 3) Swift for Storage, a highly available, distributed, eventually consistent object/blob store, 4) Glance for Image management, 5) Neutron (formerly named Quantum) for network management, and

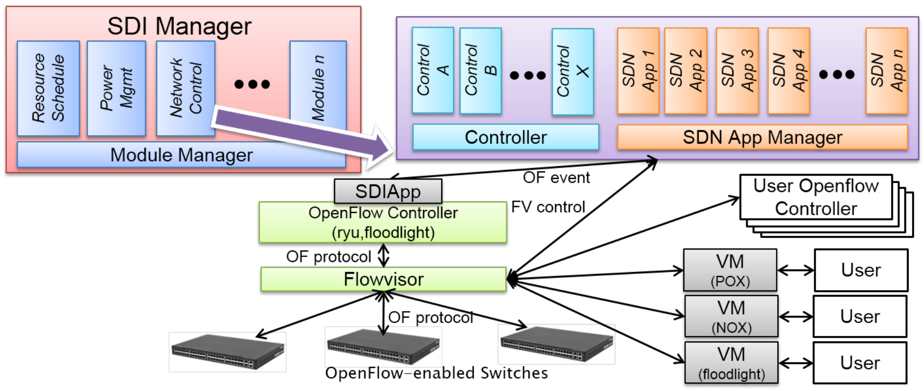

**Fig. 3.** SDI manager design and an expanded view of the network control module

6) Cinder for volume management. Because the original OpenStack Nova does not support virtualization of unconventional resources such as FPGA, NetFPGA or GPU, we have extended Nova to support virtualization of such resources by adding new device drivers. These are depicted with `*-drivers` under Nova in Fig. 2.

We have used FlowVisor [8] as a controller that acts as a transparent proxy between OpenFlow switches and multiple OpenFlow controllers. FlowVisor creates rich slices of network resources and delegates control of each slice to a different controller, while enforcing isolation between the slices. Internally, we have used the Ryu OpenFlow controller [9] which serves as a network control proxy for the SDI networking control module. Through FlowVisor, any user can use his/her own OpenFlow controller, even though it is outside the SAVI testbed as shown in Fig. 3 [7].

## 3.3 SDI Manager

The SDI manager provides integrated resource management for converged heterogeneous resources by abstraction. As shown in Fig. 3, we have designed the SDI manager as a module platform where each module is pluggable and realizes a certain function of SDI control and management such as resource scheduling, power management, network control, and so on. The SDI manager includes not only predefined modules developed by us but also SDI services through open interfaces for end users such as resource allocation APIs. By providing SDI services, end users may easily implement their services/applications using available information from SAVI testbed and test them on the SAVI testbed. For instance, if an end user wants to allocate virtual machines based on CPU core temperature of physical servers, the SDI manager can provide an API to provide a list of available physical servers and measured properties including CPU core temperature. Based on the given information, the end user can develop his or her own resource allocation algorithm and apply it to SAVI TB through the resource allocation API given by the SDI manager.

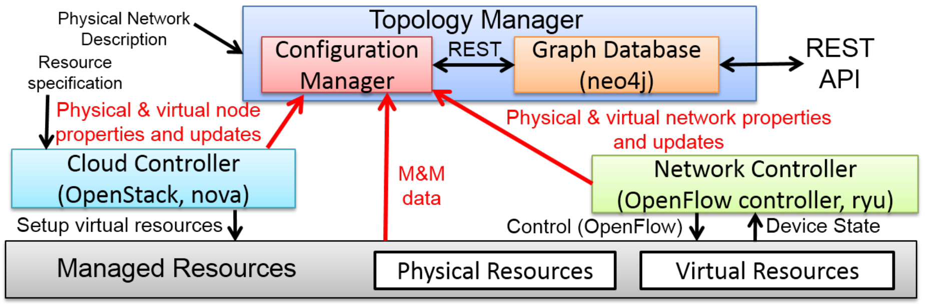

**Fig. 4.** Topology manager design

For example, Fig. 3 shows a network control module which is a predefined module and enables SDN applications over the SDI manager. The module runs one or more network control applications (e.g., learning switch, topology discovery, FlowVisor control, etc.) using controllers A, B, $\cdots$ , and X which provides receiving and handling APIs. We have implemented an application (SDIApp in Fig. 3) running on the OpenFlow controller which forwards certain OpenFlow events to the network control module. In [7] we discussed the network control module, including how to control OpenFlow-enabled networks, manage virtual networks, and delegate control to user-defined OpenFlow controllers.

### 3.4 Topology Manager

Fig.4 shows a high-level design for the topology manager where a configuration manager monitors the state and relationship between converged heterogeneous resources through a cloud controller and a network controller, and stores the monitored data to a graph database. In addition, the topology manager provides answers for the queries to the data from an SDI manager. The cloud controller and network controller each provide physical and virtual computing or networking resource properties, as well as associated monitoring and measurement data to the configuration manager.

The configuration manager builds a model by analyzing states and relationships of all monitored cloud and networking resources. We have used a graph for the model because all resources and their relationships can be represented by a set of vertices and edges with flexibility and simplicity. All physical and virtual resources are represented by a vertex and their relationship is represented by an edge. For example, physical computing server, physical network interface, physical network link, virtual machine, virtual network interfaces, virtual network link, physical network switch, physical network port, virtual network switch, virtual network port, physical access point, and any other heterogeneous resources can be a vertex in a graph model. Each vertex has its own properties such as ID, name, and associated monitoring data. In addition, the graph model includes a set of subgraphs that represent a physical or a virtual network topology. Thus the configuration manager can store not only the state and relationship of converged heterogeneous resources, but also the physical and virtual network topology.

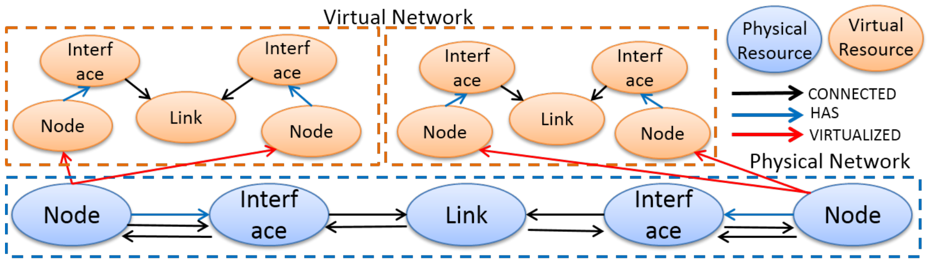

**Fig. 5.** Graph model example for cloud and networking resources

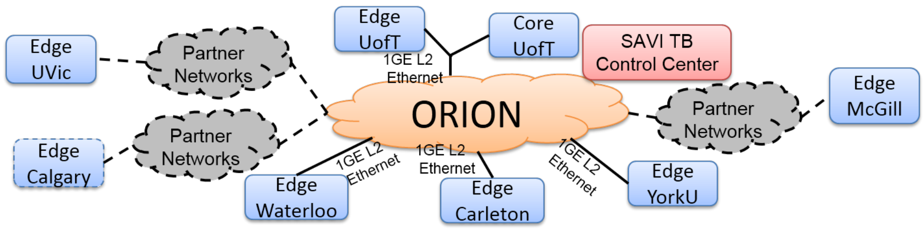

**Fig. 6.** Current deployment of SAVI testbed in Canadian universities

Fig. 5 shows a graph model example. In the SAVI TB, we have three types of network elements: Node, Interface, and Link. In the graph, a vertex represents one of the network elements with some dynamic properties. We also have three types of associations: `CONNECTED`, `HAS`, and `VIRTUALIZED`. In the graph, an edge represents the relationship. `CONNECTED` is used for one network element connected to another network element with a specified medium. `HAS` represents that a network element has another network element. `VIRTUALIZED` represents that a network element virtualizes another network element. In Fig. 5, physical resources are composed of a node, an interface and a link, and their relationships represent a physical network. Virtual resources can be virtualized on the physical resources and their relationships represent a virtual network.

We use a graph database for storing the graph model built by the configuration manager. The graph database is whiteboard friendly meaning that we can use the language of node, property, and relationship to describe our domain, so there is no need for a complicated object and relationship mapping tool to implement it in the database. We use the neo4j graph database, a popular open source graph database[10].

The topology manager provides topology information via REST APIs. It includes: 1) SAVI Node (physical server, switch, hardware sources, etc.), 2) Interface (network interface, switch port, etc.), 3) Link information, and 4) Topology.

## 4 Current Deployment and Testing

Fig. 6 shows the current SAVI node and network topology deployed in seven Canadian universities. One core node and seven Edge nodes provide cloud computing and heterogeneous resources. Currently, the SAVI testbed has 550+ CPU cores, 10+ FPGA systems, 6+ GPU systems, 50+ TB storage, 10/1GE OpenFlow-enabled switches, and wireless access points. SAVI nodes in Ontario are connected by ORION (Ontario Research and Innovation Optical Network) with a 1GE L2 ethernet link, and elsewhere connectivity is through CANARIE (Canadian Research and Education Networks). The main SAVI testbed control center is located in the University of Toronto and provides resource management services for all infrastructures. Currently, we have a project to federate with GENI in the USA.

To provide an example of real operation in the SAVI testbed, we share our experience from a hands-on tutorial for 60+ researchers on July 2013. The tutorial introduced users to the topology management service and our SDI manager to query and manage the physical and virtual network infrastructure in SAVI testbed. The topology service provides the entire network topology to users through either RESTful APIs, a CLI client, or a Python library using a graph database. We showed how to use the service to get the topology from SAVI testbed and how to apply the information for VM resource scheduling. By default, each testbed tenant has a main network which is connected to a router to enable Internet access. However, users can define their own private networks isolated from the main network and the Internet. The tutorial showed how to create a private virtual network as well as a subnet for that network. Afterwards, users learned how to control the private network using their own SDN controller. In addition, SAVI researchers showed how to deploy and manage an application on the SAVI testbed using the Cross-Cloud Application Management Platform (XCAMP) [11]. Other applications and experiments running on the testbed involve big data analysis, multimedia services, resource scheduling, virtual data center embedding, and cloud deployment are running on the SAVI testbed.

Finally we describe a wireless use case for SDI. The SAVI testbed includes OpenFlow-enabled wireless access points with the added capability to virtualize WiFi. As an example of using SDI in the provisioning and control of end-to-end services, consider a user who wishes to deploy a wireless service for clients. The user first queries the SDI manager, using its open APIs, for information regarding the capacity and capabilities of existing computing resources on the various smart edges. The information returned allows the user to allocate, again via APIs on the SDI manager, virtualized servers on physical machines chosen based on some combination of user-chosen metrics (i.e. compute capabilities, free RAM, proximity to clients, etc.). The user decides that customized network access control is needed, and thus instructs the SDI manager to delegate control of a slice of the network to the user's own OpenFlow controller (which could be running in another VM on the smart edge). In order to connect clients, the user instructs the SDI manager to virtualize the wireless access points to enable a unique ESSID, seen by client devices as an independent WiFi network. Clients who connect via this ESSID will automatically be associated with the slice of network controlled by the user, and their traffic will be controlled accordingly based on the users OpenFlow controller. Other open APIs which enable monitoring and measurement allow the user to view the state and current utilization of their computing and network resources, in turn empowering the user to adjust the capacity and availability of their service as desired.

## 5 Concluding Remarks

In this paper we have presented an SDI architecture and resource management system for infrastructures consisting of converged heterogeneous virtualized resources. SDI promises to provide flexibility, performance, and cost effectiveness, especially in the smart edge of a multi-tiered cloud. As a practical operational example, we presented the design and implementation of the SAVI testbed based on the SDI concept. SAVI provides integrated resource management services through an SDI manager and topology manager. We also presented the current deployment of the SAVI testbed and shared our experiences running applications on it.

## References

1. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.:Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.
2. Kang, J.M., Bannazadeh, H., Leon-Garcia, A.: SAVI Testbed:Control and Management of Converged Virtual ICT Resources. 2013 IFIP IEEE Intl. Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, May, 2014, pp. 664–667.
3. Kang, J.M., Bannazadeh, H., Rahimi, H., Lin, T., Faraji, M., Leon-Garcia, A.: Software-Defined Infrastructure and the Future Central Office. International Conference on Communications (ICC), budapest, Hungary, June, 2013, pp. 225-229.
4. Lu, H., Shtern, M., Simmons, B., Smit, M., Litoiu, M.:Pattern-based Deployment Service for Next Generation Clouds. In IEEE 9th World Congress on Services, Cloud Cup, Santa Clara, CA, June 28, 2013, pp. 464-471.
5. Bannazadeh, H., Leon-Garcia, A., et al.: Virtualized application networking infrastructure. 6th International ICST Conference, TridentCom 2010, Berlin, Germany, May 18-20, 2010, Revised Selected Papers, pp. 363-382, Springer. Editors: T. Magedanz, A. Gavras, N.H. Thanh, and J.S. Chase
6. “Openstack,” http://www.openstack.org.
7. Lin, T., Kang, J.M., Bannazadeh, H., Leon-Garcia A.: Enabling SDN Applications on Software-Defined Infrastructure. IEEE IFIP Network Operations and Management Symposium (NOMS 2014), Krakow, Poland, May, 2014. (Accepted to appear)
8. Sherwood, R., Gibby, G., Yapy, K.-K., Appenzellery, G., Casado, M., McKeown, N., Parulkary, G.:Flowvisor: A network virtualization layer, OpenFlow, Tech. Rep. OPENFLOW-TR-2009-1, October 2009.
9. Tomonori, F.: Introduction to ryu sdn framework, Open Networking Summit, April 2013, http://osrg.github.io/ryu/slides/ONS2013-april-ryu-intro.pdf.
10. “Neo4j Graph Database,” http://www.neo4j.org.
11. Shtern, M., Simmons, B., Smit, M., Lu, H., Litoiu, M.: Introducing the Cross-Cloud Application Management Platform (XCAMP). Submitted to the 2014 International Conference on Software Engineering.