DOI: 10.3969/j. issn. 1673-5188. 2017. 03. 001
http://kns.cnki.net/kcms/detail/34.1294.TN.20170824.1054.002.html, published online August 24, 2017

A Large-Scale NFV-Based Emulation Platform for Smart Identifier Network

LI Haifeng, LI Taixin, and ZHANG Hongke
(Institute of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China)

Abstract

Network emulation is significant because of its ability to study network architecture running on real operating system and the whole protocol stack. However, conservatively allocating a physical network equipment for a corresponding emulation network is costly, scale-limited and rigid. In this paper, based on network function virtualization (NFV), we present a large-scale emulation platform for the Smart Identifier Network (SINET). We pool and virtualize all hardware resources by lightweight virtualization technology. Controllers and orchestrators are designed to manage emulation and collect monitored SINET emulation results. The controllers are used to implement dynamically synchronously management for large-scale emulation network and link characteristics. The orchestrators are utilized to achieve overload avoidance and green computing with the Xiao's dynamic resource allocation algorithm. We implement flexibly programmable and dynamically configurable emulation.

Keywords

NFV; SINET; emulation; virtualization; control; orchestration

1 Introduction

The current problems of the Internet are a natural consequence of its architecture, which was designed to address the simple data communication needs 40 years ago [1]. The basic requirement of the Internet at that time was merely forwarding data packets among limited trust hosts. With the growth of the Internet, enormous technological innovations in both the application and network layers have emerged, which gives rise to new requirements from the architecture, such as support for trust, security, mobility and low energy consumption. However, the Internet was never designed to meet these requirements. To help evolve the Internet, there are enormous patches made for the current Internet architecture, such as Transport Layer Security (TLS), Content Delivery Network (CDN) and Network Address Translation (NAT). Unfortunately, these patches cannot fundamentally solve the existing shortcomings, and even make the current Internet more and more complex and bulky [2].

An alternative way to address new requirements is designing a new clean-state architecture for the Internet. Along this way, there have been a large number of research efforts that are proposing architectures for the Future Internet [2]–[5]. In this context, the Smart Identifier Network (SINET) has emerged as a promising architecture for the Future Internet [5]. SINET completely removes the restrictions from the triple bindings, including resource and location binding, user and network binding, and control and data binding. Besides, SINET creates three layers vertically and two domains horizontally to fundamentally solve the current Internet problems. Therefore, it can meet new requirements raised by the tremendous growth of the Internet and introduction of new scenarios.

Recently, great progress has been made in SINET in many respects [6], [7]. SINET has been applied for 5G communications [6], and it provides a specific SCN-R scheme to promote efficient and reliable communications in High-Speed Railway (HSR) scenarios. The concept of SINET was also used to deal with mobility-caused outdated mappings [7]. It designed a timer-based pointer forwarding (TBPF) approach to avoid the triangular routing problem. However, as a new clean-state architecture, SINET confronts several technological problems that have not been investigated thoroughly, and still has a long way ahead.

To further promote SINET development, an emulation platform for SINET is urgently needed. In contrast to simulators, emulation platforms provide more realistic environment for developing protocols and techniques in the new network architecture [8], [9]. Typically, an emulation platform uses real hardware resources to virtualize various network equipment and network environments. This means that researchers can use real operating systems, software and protocol stacks to run their experimental network protocols, in order to achieve actual (not simulated) performance measures.

In this paper, we propose an emulation platform for SINET, based on network function virtualization (NFV) technologies [10]. The emergence of NFV simplifies the design of emulation platforms. Besides, NFV endows an emulation platform with more programmable and flexible ability. On the basis of NFV technologies, we pool and virtualize all hardware resources, which include computing, network and special hardware resources. The pooled virtualized resources are managed by controllers and orchestrators of the emulation platform to instantiate SINET emulation.

Our goal is to implement a directly programmable, dynamically configurable and green computing emulation platform for SINET. Programmability is to assure that we can emulate SINET protocols as flexibly as general simulation (e.g., ns-3) can. Dynamical configuration means that emulation topologies and link characteristics can be dynamically and synchronously configured and managed during the whole emulating process. Finally, green computing aims at effectively preventing overload while saving energy used.

To implement preceding goals, we employ the state-of-art techniques to build the NFV-based SINET emulation platform. First, we adopt the separation technique of control and data planes. We decouple the control network and emulation network, enabling the emulation network directly programmable and the underlying physical infrastructure to be abstracted for SINET emulation applications. Second, we design a controller to monitor and manage emulation topology and link characteristics. By running a distributed sleep and busy waits algorithm, the controller implements dynamical and synchronous configuration for large-scale emulation network and link characteristics. Finally, we adopt Xiao's dynamic resource allocation technique [16] into our SINET emulation platform. With this algorithm, we can achieve both overload avoidance and green computing.

The remainder of this paper is organized as follows. Section 2 overviews SINET architecture and related emulation platforms. Section 3 gives some details of our emulation platform including overall structure, emulation control and orchestration. Conclusions are given in Section 4.

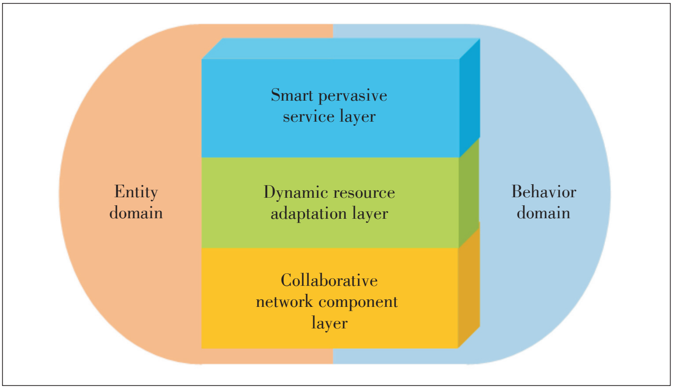
2 Related Work

This section introduces the architecture of SINET and related emulation platforms.

2.1 SINET Architecture

The architecture of SINET (**Fig. 1**) is divided into three vertical layers and two horizontal domains.

The three-layer structure consist of the smart pervasive service layer, dynamic resource adaptation layer, and collaborative network component layer. The smart pervasive service layer is in charge of the naming, registration, and management of


Smart pervasive service layer
Entity domain
Dynamic resource adaptation layer
Behavior domain
Collaborative network component layer

▲**Figure 1. SINET reference model.**

various services, including video streaming, web surfing, instant chatting, and file downloading. The dynamic resource adaptation layer is used to manage and organize network function groups, which are in charge of the optimal decision, task allocation, and resource scheduling. Each function group consists of a set of network nodes or components with similar functions such as mobility support, security enhancement, in-path caching, and energy saving. The collaborative network component layer enables network nodes such as routers, content servers, sensors, and interfaces to carry out a specific task, including routing, data transmission/caching, mobility proxy, and power saving (in sleep mode).

The two domains are the entity domain and behavior domain. The entity domain aims at providing identifiers of entities such as network services, function groups, and network nodes. The behavior domain describes properties of different entities, including service characteristics such as service type and service cache and provider signatures.

With three layers and two domains, SINET dynamically gets network status and intelligently meets service requirements. Network function groups and network components in a function group can be properly selected. Hence, SINET can provide smart pervasive service. Meanwhile, network strategies, such as behavior matching, behavior clustering and network complex behavior game are adopted to achieve dynamic adaptation and cooperative scheduling for network resources. In this way, SINET greatly increases the utilization of network resources, further reduces the consumption of energy and improves user experience.

2.2 Related Emulation Platform

There are a number of emulation platforms for different network architecture and purposes [8], [9], [12], [14], [15]. Emulab [8] is well-known for its scalability and Linux Container virtualization (LXC). The design principles in Emulab have a significant influence on the following design of emulation platforms. However, with advances in technology, Emulab techniques proposed in 2008 would be outdated. PlanetLab [9] is a global overlay network for developing and accessing broad-coverage network services. PlanetLab uses the slice technique to run multiple services concurrently and continuously. This slice technique is similar to network slice in 5G, which can be viewed as network virtualization technology. Currently, this technique has been rapidly developed and fully utilized in cloud computing and NFV. In parallel to network virtualization technologies, compute virtualization technologies have been developed and tend to mature, such as LXC, Linux-Vserver, and Docker [11]. In particular, LXC is used as OS-level virtualization technology in Mininet [12]. Mininet is a lightweight emulation platform which is designed specifically for soft defined networking (SDN) [13]. Mininet enables a laptop to produce hundreds of emulating nodes by LXC. Handigol [14] improves Mininet with resource provisioning, isolation and monitoring system, and enormous typical experiments have been reproduced to prove that Mininet has good fidelity.

Besides, we [15] proposed an OpenStack-based Delay Tolerant Networking (DTN) Emulation Platform (EmuStack). EmuStack integrates OpenStack with Linux Traffic Control (TC) tools for managing and emulating the virtual link characteristics that include variable bandwidth, delay, loss, jitter, re-ordering and duplication. Using Docker container technology and network namespaces, EmuStack can support a large-scale topology (including hundreds of nodes) with only several physical nodes.

3 Emulator System Design

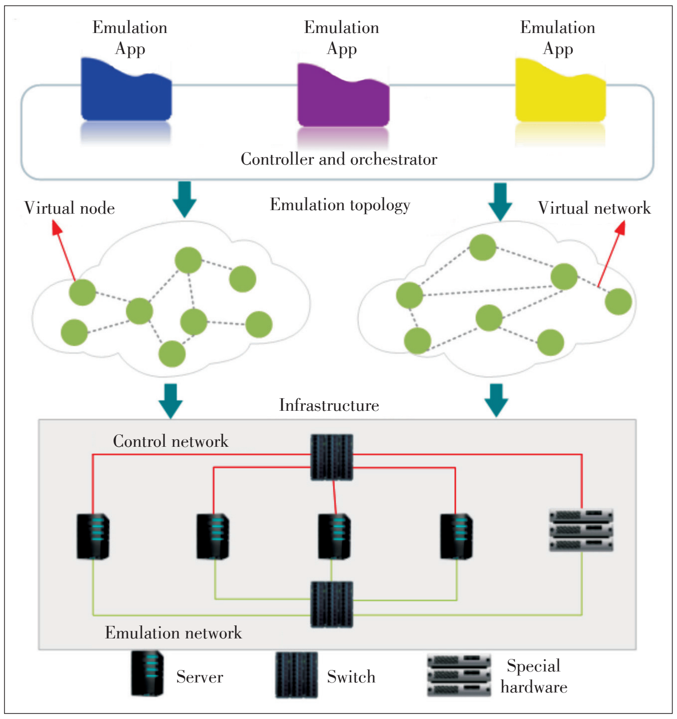
This section describes the overall architecture of NFV-based emulation platform for SINET, and discusses emulation control and orchestration, which are critical for the SINET emulation platform.

3.1 Design Overview

Fig. 2 presents the overall platform design structure. NFV is employed to manage the hardware infrastructure, which is composed of standard X86 servers, high-performance switches and special hardware. In particular, the special hardware is used to accurately emulate complex and professional network environments, such as deep space link and high-speed traffic. It is uniformly described, and can work with X86 servers to carry three types of nodes, namely, emulation nodes, controller nodes and orchestrator nodes.

Emulation nodes consist of physical emulation nodes and virtual emulation nodes. The physical emulation node is special hardware or a standard x86 server which can be virtualized massively to virtual emulation nodes. Virtual emulation nodes can be created by different computing virtualization technologies, such as Docker, Xen and Kernel-based Virtual Machine (KVM). A hybrid deployment is allowed where different computing virtualization technologies can be employed concurrently with multiple physical emulation nodes.

Controller and orchestrator nodes run on standard X86 servers. In the SINET emulation platform, there are at least one controller and orchestrator, which are responsible for managing SINET emulation, monitoring and collecting experimental results. In details, the controller is in charge of creating and managing compute and network resources. On the basis of users' configuration, the controller invokes the virtualized infrastructure manager (VIM) to instantiate virtual emulation nodes and network. After starting an emulation, the controller monitors and collects emulating data. As to the orchestrator, it supervises usage of hardware resources and dynamically orchestrates the whole hardware resources including CPU, memory and network, hence, achieving emulation overload avoidance and green computing.



▲Figure 2. The overall design structure of emulator.

Emulation nodes, controller nodes and orchestrators are equipped with multiple Network Interface Cards (NICs), which are connected with each other by switches to construct the physical network. The physical network is divided into a control network and an emulation network, as shown in the bottom part of Fig. 2. This design enables the network to become directly programmable and dynamically configurable. The control network carries control traffic that consists of network and emulation management information. The emulation network transfers emulation data. Emulation data varies greatly with different SINET experiments, which probably consumes enormous bandwidth of the emulation network. In practice, physical emulation network is the primary limit in the SINET emulation platform. Therefore, we equip each physical emulation node with multiple emulation NICs to carry the large-amount emulation traffic. These NICs are bridged to an Open vSwitch

(OVS), whose internal device is assigned an emulation network IP address. With these bridging technique, we extend bandwidth of emulation network to *n* times than one NIC, where *n* is the number of emulation NICs.

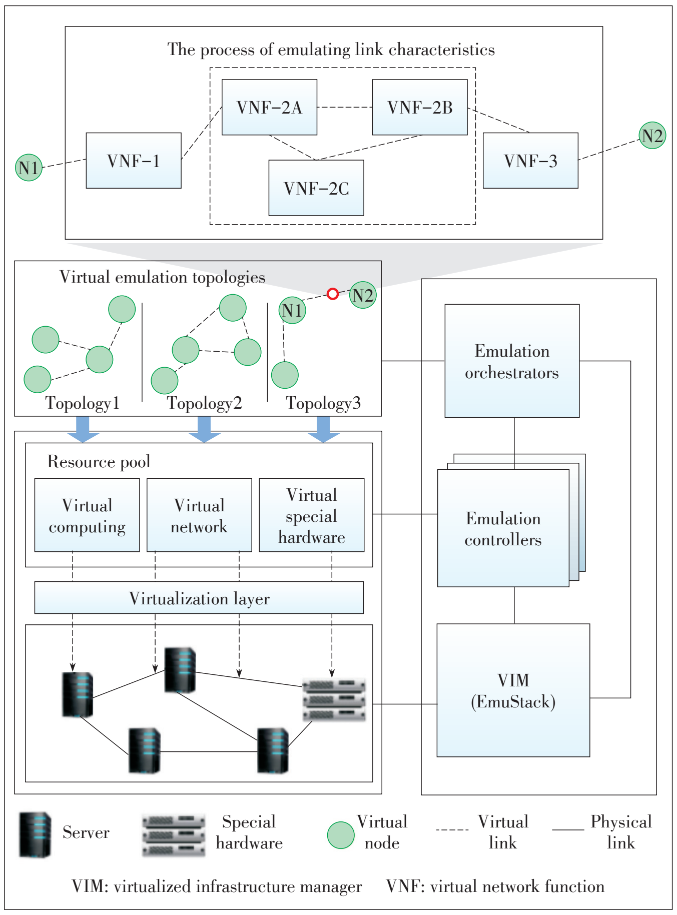
The middle part of Fig. 2 presents the solution to generating virtual emulation topologies in the SINET emulation platform. Virtual emulation topology is composed of virtual emulation nodes and network. Virtual emulation nodes are spawned by the orchestrator with the custom operating system images equipped with SINET experimental protocol software. The virtual emulation network is overlay network built on top of the physical emulation network. SDN and encapsulation techniques are employed to virtualize physical emulation network. The SDN has multiple kinds of network technologies designed to make the network more flexible and agile. The SINET emulation platform uses (Open vSwitch) OVS as SDN switch, and integrates the SDN controller into the emulation controller, hence, achieving the ability to dynamically program and configure virtual emulation network. Additionally, encapsulation techniques consist of Generic Network Virtualization Encapsulation (GNVE) protocol and virtual extensible LAN (VXLAN). These techniques are employed to isolate the user emulation traffic from other users, and accordingly offer each user an oblivious context by which different users can create respective virtual emulation network on the same underlying infrastructure and do not interfere with each other.

Virtual emulation topology is managed by user control emulation applications (Apps). According to his emulating purpose and scenario, a user writes an APP to dynamically control virtual emulation nodes and network. In order to further evaluate SINET performance, the emulation platform offers abundant link characteristic emulations, including link bandwidth, latency, jitter, packet loss, duplication and re-ordering. An App invokes EmuStack [15] to control dynamically these link characteristics. By setting Hierarchical Token Bucket of EmuStack within the network namespace, a controller App can dynamically and independently manage every virtual link characteristic for each virtual emulation topology.

3.2 Emulation Control and Orchestration

Fig. 3 shows emulation control and orchestration. In the SINET emulation platform, the entire hardware resource is virtualized into the resource pool. The controller and orchestrators are employed to control and orchestrate this resource pool.

The emulation controllers are primarily responsible for dynamically and synchronously controlling virtual emulation topology instantiated from the resource pool. For the large-scale SINET emulation platform, it is a challenge. First, there are hundreds of nodes that have stochastic communication delay and background system load, which makes it difficult and complex to control them. Second, different from the simulator (e.g., ns-3) based on discrete event and run in virtual time, the SINET emulation platform runs in real time. Therefore, the controller cannot pause a emulation node’ clock to wait for other pending events, which further raises the problems such as synchronization in control.



▲**Figure 3. The diagram of emulation control and orchestration.**

In order to overcome these problems, we employ a distributed pre-allocating control algorithm. In detail, each physical emulation node (compute node) runs as a local control agent, which is responsible for controlling the emulating topologies that are instantiated locally. Before starting a emulation, the controller dispatches control information to the local control agent in advance. Control information primarily consists of management commands of SINET experimental protocols, local link characteristics and topological state of different period. Especially, a solution time stamp is sent with control information, which is used as the experimental start time for each local control agent.

The local control agent employs the sleep and busy waits algorithm to exactly and effectively execute emulating events. When there are amounts of time remained before starting experiment, local control agent goes on sleep-waits mode. Sleep-waits cause local control agent to yield the processor for some amount of time. This specified amount of time is actually converted to an operating system specific granularity, even though it can be passed to nanosecond resolution. For the SINET emulation platform, every X86 server run Linux. In Linux, the granularity is namely Jiffy. Typically, this resolution is insufficient for our needs (generally, ten milliseconds), so we round down and sleep for some smaller number of Jiffies. After hardware real time is close the solution time stamp of experimental start, local control agent is in busy-waits mode and awakened. At this time, we have some residual time to wait. This time is generally smaller than the minimum sleep time, so we busily wait for the remainder of the time. This means that the thread of local control agent just sits in a loop, and consumes cycles until the experimental time arrives.

For all events in emulation scheduling queue, we all adopt the sleep and busy waits algorithm to wait for the time of emulation events. After the previous event combination of sleep and busy waits, the elapsed real-time (wall) clock should agree with the emulation time of the next event, and then the emulation proceeds.

Orchestrators are primarily responsible for dynamically orchestrating hardware resource to ensure there are adequate CPU, memory, network and special hardware resource available to provide users emulation. For the large-scale SINET emulation platform, dynamically large-scale orchestration is also a challenge. First, the resource needs of virtual emulation nodes are heterogeneous due to the diverse SINET experiments, and vary with time as the workloads shrink and grow. It is difficult to forecast the utilization of resource pool when multiple users emulate experimental protocols concurrently. Second, the capacity of physical emulation nodes (standard x86 servers) can also be heterogeneous since multiple generations of servers co-exist in the emulation platform. Thus this heterogeneity of hardware further increases complexity of orchestrating resource.

To solve aforementioned orchestrating problems, we apply Xiao's dynamic resource allocation algorithm to the SINET emulation platform to orchestrate emulation resource in [16]. We aim at effectively avoiding overload but minimizing the number of servers used. In practice, there is an inherent tradeoff between these two goals. To avoid overload, we should keep the utilization of physical emulation node low to reduce the possibility of overload in case the resource needs of virtual emulation nodes increase later. To minimize amounts of used servers, we should keep the utilization of physical emulation nodes reasonably high to utilize efficiently their energy. Fortunately, Xiao's resource allocation algorithm nearly optimally tackled this tradeoff. We apply it to the SINET emulation platform, and adopt Xiao's skewness concept to measure the uneven usage of a server. By minimizing skewness, we can enhance the overall utilization of server. Meanwhile, the load prediction algorithm in [16] is employed to achieve the future resource usages. It helps to reduce the placement churn significantly.

Besides, orchestrators can work either with controller or Virtualized Infrastructure Manager (VIM), and provide management of virtual network functions (VNFs) to emulate more link characteristics. VNFs run on industry-standard X86-based servers like virtual emulation nodes, which provides a well-defined functional behavior, such as forwarding, routing and firewall. Typically, emulation link characteristics including delay, error ratio and link rate are emulated by controller with EmuStack. However, there are still several link characteristics which cannot be realized by EmuStack. For example, it cannot emulate more than 300 s link delay with EmuStack. However, if we emulate SINET for deep space communication, 300 s delay would not meet emulating requirement. To emulate this longer link delay, a VNFs chaining would be orchestrated to endow a virtual link with the longer delay. The VNFs chaining is generally composed of multiple VNFs as shown in the upper part of Fig. 3.

4 Conclusion

In this paper, we present a NFV-based emulation platform for Smart Identifier Network (SINET). We first introduce SINET architecture, and then outline the design of the SINET emulation platform. We design controllers and orchestrators for the SINET emulation platform. With the controllers running a distributed sleep and busy waits algorithm, we achieve dynamical and exactly synchronous emulation management for large-scale SINET emulation. In order to achieve overload avoidance and green computing, we apply the Xiao's dynamical resource allocation algorithms to Orchestrators. Thus, we ensure there are always ample CPU, memory and network resources to provide SINET emulations for different users, while minimizing the number of underutilized servers.

References

[1] G. Xylomenos, C. N. Ververidis, V. A. Siris, et al., "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014. doi: 10.1109/SURV.2013.070813.00063.

[2] H. Luo, Z. Chen, J. Cui, et al., "CoLoR: an information-centric internet architecture for innovations," *IEEE Network*, vol. 28, no. 3, pp. 4–10, 2014. doi: 10.1109/MNET.2014.6843226.

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, Jul. 2012. doi: 10.1109/MCOM.2012.6231276.

[4] L. Zhang, A. Afanasyev, J. Burke, et al., "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, Jul. 2014. doi: 10.1145/2656877.2656887.

[5] H. Zhang, W. Quan, H. Chao, and C. Qiao, "Smart identifier network: a collaborative architecture for the future internet," *IEEE Network*, vol. 30, no. 3, pp. 46–51, 2016. doi: 10.1109/MNET.2016.7474343.

[6] H. Zhang, P. Dong, W. Quan, and B. Hu, "Promoting efficient communications for high-speed railway using smart collaborative networking," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 92–97, Dec. 2015. doi: 10.1109/MWC.2015.7368829.

[7] H. Zhang, H. Luo, and H. Chao, "Dealing with mobility-caused outdated mappings in networks with identifier/locator separation," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 199–213, 2015. doi: 10.1109/

TETC.2015.2449664.

[8] M. Hibler, R. Ricci, L. Stoller, et al., "Large-scale virtualization in the emulab network testbed," in *USENIX Annual Technical Conference*, Boston, USA, 2008, pp. 113-128.

[9] B. Chun, D. Culler, T. Roscoe, et al., "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3-12, Jul. 2003. doi: 10.1145/956993.956995.

[10] H. ZHAO, Y. XIE, and F. Shi, "Network function virtualization technology: progress and standardization," *ZTE Communications*, vol. 12, no. 2, pp. 3-7, Jun. 2014. doi: 10.3969/j.issn.1673-5188.2014.02.001.

[11] M. G. Xavier, M. V. Neves, F. D. Rossi, et al., "Performance evaluation of container-based virtualization for high performance computing environments," in *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Belfast, UK, 2013, pp. 233-240. doi: 10.1109/PDP.2013.41.

[12] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Monterey, USA, 2010, article No. 19. doi: 10.1145/1868447.1868466.

[13] Z. Sun, J. Li, and K. Yang, "Software-defined networking," *ZTE Communications*, vol. 12, no. 2, pp. 1-2, 2014.

[14] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *ACM Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012. doi: 10.1145/2413176.2413206.

[15] H. Li, H. Zhou, H. Zhang, and W. Shi, "EmuStack: an OpenStack-based DTN network emulation platform (extended version)," *Mobile Information Systems*, vol. 2016, article ID 6540207, 2016. doi: 10.1155/2016/6540207.

[16] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107-1117, Jun. 2013. doi: 10.1109/TPDS.2012.283.

Manuscript received: 2017-05-26

Biographies

LI Haifeng (haifengli@bjtu.edu.cn) received the B.E. degree in communication and information systems from the Beijing Jiaotong University, Beijing, China, in 2013. He is currently working toward his Ph.D. degree at the School of Electronic and Information Engineering, Beijing Jiaotong University. He has participated in several national research programs of China such as "973" Program and "863" Program. His research interests include future Internet, Delay Tolerant Networking, High Performance Computing and Cloud Computing.

LI Taixin (14111040@bjtu.edu.cn) received the B.S. degree in telecommunications engineering from Beijing Jiaotong University in 2013, and then entered National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. Currently, he is pursuing the Ph.D. degree in telecommunications and information system at BJTU, China. His research interests include next generation internet, network services and satellite network.

ZHANG Hongke (hkzhang@bjtu.edu.cn) received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China (formerly known as Chengdu Institute of Radio Engineering) in 1988 and 1992, respectively. From September 1992 to June 1994, he was a Postdoctoral Research Associate at Beijing Jiaotong University (formerly known as Northern Jiaotong University). In July 1994, he joined Beijing Jiaotong University, where he is a Professor. He currently directs a National Engineering Lab on Next Generation Internet in China. He is a Senior member of IEEE, and has published more than 100 research papers in the areas of communications, computer networks, and information theory. He is the author of eight books written in Chinese and the holder of more than 30 patents. Dr. Zhang received the Zhan Tianyou Science and Technology Improvement Award in 2001, the Mao Yisheng Science and Technology Improvement Award in 2003, the First Class Science and Technology Improvement Award of the Beijing government in 2005, and other various awards. He is now the Chief Scientist of a National Basic Research Program ("973" Program).