








Network Emulation as a Service (NEaaS): Towards a Cloud-Based Network Emulation Platform

Junyu Lai^{1,2}, Jiaqi Tian¹, Dingde Jiang¹, Jiaming Sun¹,
and Ke Zhang¹

¹ School of Aeronautics and Astronautics, University of Electronic Science and Technologies of China, Xiyuan Ave no. 2006, Chengdu, China
{laijy, tianjq, jiangdd, sunjm, zhangk}@uestc.edu.cn
² Science and Technology on Communication Networks Laboratory, Shijiazhuang, China

Abstract. Network emulation is an essential method to test network architecture, protocol and application software during a network's entire life-cycle. Compared with simulation and test-bed methods, network emulation possesses the advantages of accuracy and cost-efficiency. However, legacy network emulators are typically restricted in scalability, agility, and extensibility, which builds barriers to prevent them from being widely used. In this paper, we introduce the currently prevalent cloud computing and related technologies including resource virtualization, NFV (network functional virtualization), SDN (software-defined networking), traffic control and flow steering to the network emulation domain. We design and implement an innovative cloud-based network emulation platform, aiming at providing users Network Emulation as a Service (NEaaS), which can be conveniently deployed on both public and private clouds. We carried out performance evaluation and discussion on this platform. It turns out, the platform can significantly outperform most legacy network emulators regarding to the scalability, agility, and extensibility, with much lower emulation costs.

Keywords: Network emulation · Cloud computing · NFV (network functional virtualization) · SDN (software-defined networking) · Flow steering

1 Introduction

Modern networking systems are getting far more complicated than before. It is already difficult to rely on theoretical methods to analyze network performance. Therefore, network testing technologies are of great importance to the design and implementation of network architecture, protocols and upper layer applications. There are mainly three network testing methods.

Firstly, computer simulation is a technique whereby a software program models the behavior of a network by calculating the interaction between the different network entities (nodes, links, etc.). Most simulators use discrete event simulation, and the simulation method cannot be very precise due to inaccurate models, although its

running speed is usually fast and the cost is pretty low. Secondly, a live test-bed is a platform for conducting rigorous and transparent testing of network protocols and applications. It is a prototype of the target network, and is the most accurate method with the highest cost. Thirdly, network emulation is a technique for the performance testing of original protocols and applications over a virtual network, which is different from computer simulation where purely mathematical models are applied. A network emulator appears to be a real network, and is with medium cost and high accuracy.

This paper focuses on network emulation, which has been researched for more than half century. However, due to technology constraints, traditional network emulators have been restricted in scalability, agility, and extensibility for a long time, which significantly influenced their applications in a larger scale. This research introduces the currently prevalent cloud computing and related ICT technologies including resource virtualization, NFV, SDN, traffic control and flow steering to the network emulation domain, aiming at eliminating the above restrictions. More preciously, the major contribution of this paper is to design and implement an innovative cloud-based network emulation platform, to provide users Network Emulation as a Service (NEaaS). The NEaaS can be deployed either public or private cloud to satisfy diverse user needs.

The remaining part of this paper goes as follows. The related work in industry and academia are briefly reviewed in Sect. 2, followed by the innovative design of the cloud-based network emulation platform in Sect. 3, and its implementation details in Sect. 4. Performance evaluation of the proposed emulation platform is presented in Sect. 5. Finally, Sect. 6 concludes the paper and provides the outlook.

2 Related Work

In industry, in 2003, Wellington and Kubischta [1] from General Dynamics presented an approach for integrating and testing wireless systems in the laboratory with real-time emulation of ad hoc radio networks. In 2006, Yousefi'zadeh et al. [2] from Boeing Company collaborated with University of California reported the addition of emulation functionality to the NEWS testbed capturing fading wireless link effects. In 2007, Bonney et al. [3] from Architecture Technology Corporation, developed a hardware-in-the-loop emulator known as ABSNE that creates a controllable, repeatable, virtual network environment. Also in 2007, Beuran et al. [4] from NICT of Japan, had presented the design of QOMET, a wireless LAN emulator, with a versatile two-stage scenario-driven design. The details of the improved model and additional functionality were given in 2008 [5], and in 2015 [6], the authors again presented a framework for evaluating wireless network performance through emulation, using a hybrid design. In 2008, Ahrenholz et al. [7] from Boeing Phantom Works, presents CORE (Common Open Research Emulator), a real-time network emulator that allows rapid instantiation of hybrid topologies composed of both real hardware and virtual network nodes. Also in 2008, Nickelsen et al. [8] from Aalborg University and FTW described how to create reproducible test conditions by emulating the wireless links. In 2015, Soles et al. [9] from University of West Florida, collaborated with Northrop Grumman Aerospace Systems, researched on the need to provide an emulation method for studying the interaction among diverse hardware and software components.

In academia, Giovanardi et al. [10–12] from University of Ferrara described the emulation facility in the Simple Ad hoc simulator (SAM), which is able to emulate many unicast routing protocols with a real exchange of signaling and data packets. In 2007, Maier et al. [13] from University of Stuttgart focused on scalable network emulation problems, and present a comparison of different virtual machine (VM) implementations (Xen, UML) and their virtual routing approach (NET). In 2009, Mehta et al. [14], described a new emulation architecture that is scalable, modular, and responds to real time changes in topology and link characteristics. In 2014, Balasubramanian et al. [15] from Vanderbilt University, described a rapid development and testing framework for a distributed satellite system. In 2015, To et al. [16] from Universidad Galileo, presented the Dockemu tool for emulation of wired and wireless networks.

Although there have already been a plenty of emulators as introduced above, most of them are serving for dedicated purposes, and are to some extent, restricted in scalability, agility, and extensibility. In the last decade, cloud computing and its related ICT technologies have developed rapidly, which motivates the researchers to leverage these promising technologies to be applied in developing network emulators in an innovative manner.

3 Innovative Cloud-Based Network Emulation Platform

3.1 Vital Feature Requirements on Modern Network Emulator

Current networking systems appear to be with a large number of nodes, wired or wireless links, and dynamic topologies, which introduces new requirements on modern network emulators. Firstly, scalability. Emulator should support the number of emulated nodes from several to tens of thousands, without changing the hardware and software architecture. Secondly, agility. Building of a target emulation scenario should be fast enough to satisfy the user QoE. Thirdly, extensibility. Emulator should be extensible to hold newly appeared nodes and links in a convenient manner. Fourthly, low-cost. Without sacrificing emulation accuracy, emulator can emulate larger scale networks.

3.2 Network Node Emulation Scheme

The principle of network node emulation is to utilize the VMs created and allocated by the cloud platform to imitate the nodes in target networks. More precisely, to emulate a router node, the cloud platform calls the underlying hypervisor to create a VM instance, and then allocates this VM to the emulator. Considering the fact that the targeted router itself is a computer, it would be straight forward to adopt the allocated VM to emulate that router.

3.3 Network Link Emulation Scheme

Emulating target network links relies on using the virtual network links of the cloud platform. The cloud virtual network consists of virtual links which connect multiple VMs. Therefore, to emulate a specific link between two nodes in the target network, the emulator sets a virtual link between the two corresponding VMs. The physical and mac layer characteristics of the target link shall be accurately mapped to the network layer attributes of the virtual network link.

3.4 Network Topology Emulation Scheme

Both wired and wireless network topology will change as time goes. For wired network, the reason could be the failures happened on some certain links or nodes. While for wireless network, it may be the consequence of link break caused by node mobility, nodes failure, etc. To emulate network topology is to dynamically control and adjust the virtual network links between the VMs in a fast enough manner to satisfy the emulation needs.

3.5 Architecture of the Cloud-Based Network Emulation Platform

An architecture of the cloud-based network emulation platform is designed and presented in Fig. 1, which is divided into four layers: resource virtualization layers, cloud computing layers, emulation core layers, and emulation interface layers.

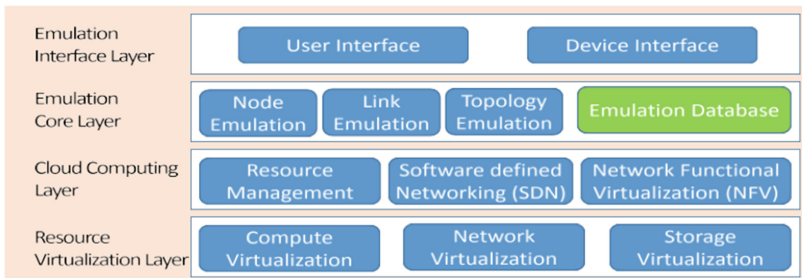


Fig. 1. Architecture of the cloud-based emulation platform.

Resource Virtualization Layer. Its functionality is to abstract, virtualize and pool all sorts of underlying hardware resources. The major modules are included: (1) Compute virtualization. This module creates VMs that acts like a real computer with an operating system; (2) Network virtualization. The module combines hardware network resources and network functionality into a single, software-based administrative entity; (3) Storage virtualization. This module presents a logical view of all the physical storage resources, treating all storage media in the system as a single pool of storage.

Cloud Computing Layer. It is a cloud Operating System, responsible for providing resources in different forms according to the upper layer's requirements in real-time. Primarily, this layer consists of three modules: (1) Resource management. The module allocates and frees compute, network, and storage resources to satisfy the emulation needs; (2) SDN. This module contains two type of entities, i.e., SDN controller and virtual switch located in physical machine. Together with the network virtualization module of the lowest layer, this module can provide traffic control and flow steering functionalities, which are the key features to implement link and topology emulations; (3) NFV. The module embodies the principle of node emulation. Each node in the target network can be emulated by a generic VM with NFV enhancement, which includes dedicated functionality implementations of the target node by means of software.

Emulation Core Layer. It contains three modules covering node, link and topology emulations, respectively, plus with one emulation database storing emulation status and parameter values: (1) Node emulation. By calling the lower layer's resource management and NFV module, this layer can accomplish the tasks of node emulation; (2) Link emulation. The resource management and SDN modules were utilized to control the accessibility among arbitrary nodes, together with network ports settings on the emulated nodes to define the characteristics of the corresponding links; (3) Topology emulation. This module still relies on the lower layer's resource management and SDN modules, and considers all the nodes and links, which form the emulated network topology to be consistent with the target network; (4) Emulation database. The database to record and store the parameter values of the emulated nodes, links and topologies. The users can store an emulation scenario by means of writing the status of all its elements into the database, and later on, the scenario can again be recalled according to the database.

Emulation Interface Layer. Two types of interfaces are considered in the platform: (1) User Interface. It provides users the graphic interface to accomplish a series of typical emulation operations, such as creation of nodes, links, as well as topologies, management of emulation scenarios, etc.; (2) Device Interface. The emulation platform supports the connections to real network nodes. It could be either a single node or an external network consisting of an amount of real nodes.

4 Implementation Details

The hardware components of the cloud based emulation platform is illustrated in Fig. 2. It consists of a number of COTS computers and switches. More precisely, all the computers are X86 based (i.e., AMD 1700X, 64G RAM, 500G SSD), five of which are emulation nodes, the rest two are emulation and SDN controller, respectively.

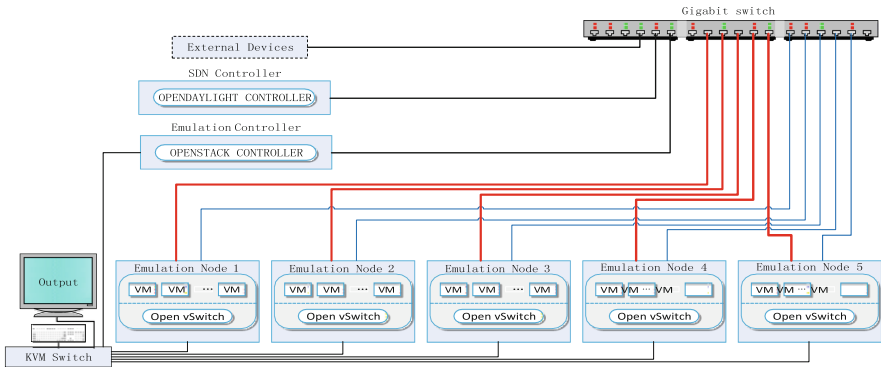


Fig. 2. Hardware components of the cloud-based emulation platform.

4.1 Implementation of Resource Virtualization Layer

Considering the balance among efficiency, generality, and cost, Linux Kernel based Virtual Machine (KVM) is adopted as the hypervisor. KVM requires a CPU with hardware virtualization extensions. A wide variety of guest operating systems work with KVM, including many flavors and versions of Linux, BSD, Solaris, Windows, OS X, Android, Solaris 10, etc. which support the emulation nodes installed with diverse operating systems. KVM is installed and configured in each emulation node to virtualize and pool all the compute, storage, and network resources. In particular, VMs with diverse profiles are created on top of host machines by KVM, according to emulation needs.

4.2 Implementation of Cloud Computing Layer

OpenStack, OpenDaylight (ODL), and Open vSwitch (OVS) are employed to implement the three modules in this layer. OpenStack is a free and open-source software for cloud computing, mostly deployed as IaaS, whereby virtual servers and other resources are made available to users. OpenStack consists of interrelated components that control diverse hardware pools of compute, storage, and networking resources. Users either manage it through a web-based dashboard, through command-line tools, or through RESTful web services. OpenStack has a modular architecture with various project names for its components. The goal of the ODL project is to promote SDN and NFV, with a clear focus on network programmability. ODL is a modular open platform for customizing and automating networks of any size and scale. In the emulation platform, Openstack and OVS are deployed on the emulation and the emulation controller nodes. ODL is installed on the SDN controller node. By using ODL plug-in to replace OpenStack Neutron's original ones, the designed functionalities can be achieved.

4.3 Implementation of Emulation Core Layer

The emulation platform is B/S model based. The primary functions, such as node, link and topology emulations, are implemented in the web server side, together with the database record the emulation status. Both of the web server and the database are deployed on the emulation controller. Users can utilize any browser to access that web server, and to accomplish their emulation tasks.

The web server includes node, link and topology emulation module. The node emulation module is responsible for the creation, configuration, and deletion of the emulated nodes. It mainly calls the OPENSTACK dashboard API to accomplish these tasks. The link emulation module can then configure the emulated node’s network ports to imitate the ports of the target nodes. Linux Traffic Control is the major tool adopted to set the bitrate, delay and packet loss attributes of the network ports. The SDN controller is also commanded by the link emulation module via ODL Controller API to creation the virtual links connecting the emulated nodes. On the basis of node and link emulation, the topology module supports the emulation for both the static topology which keeps unchanged during the emulation, and the dynamic topology which changes according to user settings or predefined trace files. All the above modules programmed in Java at the server side, follows the MVC (Model, View and Controller) design pattern.

As afore mentioned, a database module is designed to store emulation status. MySQL is chosen to be the DBMS. Hieratical tables are built to record the emulation parameter values and the relations.

4.4 Implementation of Emulation Interface Layer

This layer contains user interface and device interface modules, serving for different purposes. The user interface is the front-end of the web server. Technologies, such as HTML5, jQuery, Ajax, jTopo, etc. are employed to construct the web-based user interface. Users can conveniently create and configure emulation scenarios. The web pages designed for emulation tasks and their logical relations are given in Fig. 3. To realize the united emulation between the emulation platform and a part of the real network, the device interface module is developed. A real network device can be directly attached to the emulation platform, and an inner agent, corresponding to the external device will be created automatically, and will be connected to the external device via a virtual L2 link. Figure 4 illustrates the implementation principles.

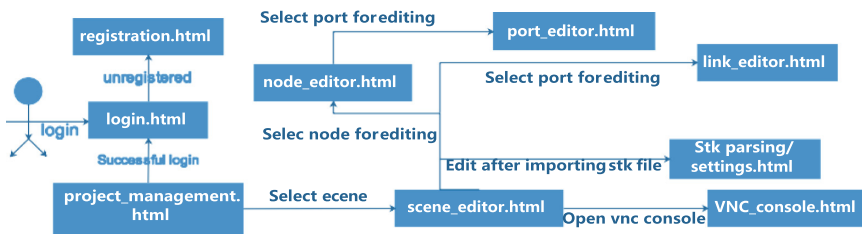


Fig. 3. Web pages designed for emulation tasks and their logical relations.

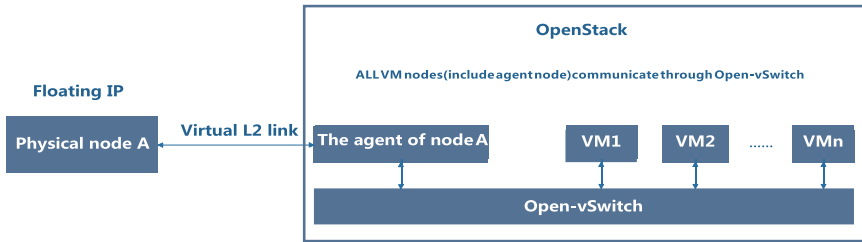


Fig. 4. Implementation principles for device interface module.

5 Performance Evaluation

5.1 Performance Metrics Introduction

The proposed emulation platform can conquer most legacy emulator's defects in scalability, agility and extensibility. Therefore, the number of emulated nodes supported by the platform is chosen as the metric for scalability, while the creation time of an emulation scenario is selected as the measure of agility. The platform can create different types of emulated nodes and links only restricted by the VM and channel templates, and thus has a very good extensibility.

5.2 Evaluation and Discussion

For the scalability evaluation, the number of emulated nodes a single emulation node can support has been tested; the quantity of the emulated nodes supported by the platform shall be the summation of the nodes all the physical nodes can emulate. Experiments show that around 50 emulated nodes can simultaneously run on each single emulation node, and for 5 emulation nodes of the same profile, 250 emulated nodes shall be supported. The number can be linearly scaled up by simply increasing the number of physical emulation nodes.

For the agility evaluation, a scenario with 100 emulation nodes is investigated. The experiments of creating 10 emulated nodes on a single machine is carried out, and it turns out 27 s is consumed, which means around 2.7 s are needed for creating one emulation node. Since the platform has 5 physical emulation nodes, and each nodes can create VMs independently, the time spent on building the scenario is around 5.4 s, excluding the detailed configuration time for each node.

To summarize, the proposed platform's performances on scalability, agility and extensibility are much better than most legacy emulators.

6 Conclusion

Network emulation is regarded as the most promising network testing method due to its balance on cost and accuracy. This paper focused on solving network emulation's inherent shortcomings in scalability, agility and extensibility. In particular, the paper

designed and implemented an innovative cloud-based network emulation platform aiming at providing users NEaaS. Performance evaluation and discussion illustrated that the proposed platform can effectively outperform legacy network emulators regarding to scalability, agility, and extensibility.

The potential research work we have planned for the next step includes the following two points: Firstly, the cost of VM-based emulation is still high, light-weighted virtualization technologies, i.e., Docker, will be investigated to replace VMs. Secondly, the target network's heterogeneous nodes of diverse hardware architectures, such as ARM, Sparc, and Power PC, are currently emulated by X86 architected VMs, which is inaccurate to some extent. Emulation for heterogeneous nodes will be further studied.

Acknowledgement. This work is partially supported by the Science and Technology on Communication Networks Laboratory(Grant No. XX17641X011-03), the 54th Research Institute of China Electronics Technology Group Corporation, and the National Natural Science Foundation of China (Grant No. 61402085 and 61872051).

References

1. Wellington, R.J., Kubischta, M.D.: Wireless network emulation for distributed processing systems. In: IEEE Military Communications Conference, 2003. MILCOM 2003, Boston, MA, USA, vol. 1, pp. 475–480 (2003). <https://doi.org/10.1109/MILCOM.2003.1290149>
2. Yousefi'zadeh, H., Li, X., Furmanski, W., Lofquist, D.B.: Emulation of fading wireless link effects in NEWS wired testbed. In: MILCOM 2007 - IEEE Military Communications Conference, Orlando, FL, USA, pp. 1–7 (2007). <https://doi.org/10.1109/MILCOM.2007.4455162>
3. Bonney, J., Bowering, G., Marotz, R., Swanson, K.: Hardware-in-the-loop emulation of mobile wireless communication environments. In: 2008 IEEE Aerospace Conference, Big Sky, MT, USA, pp. 1–9 (2008). <https://doi.org/10.1109/AERO.2008.4526345>
4. Beuran, R., Nguyen, L.T., Latt, K.T., Nakata, J., Shinoda, Y.: QOMET: a versatile WLAN emulator. In: 21st International Conference on Advanced Information Networking and Applications (AINA 2007), Niagara Falls, ON, Canada, pp. 348–353 (2007). <https://doi.org/10.1109/AINA.2007.116>
5. Nickelsen, A., Jensen, M.N., Matthiesen, E.V., Schwefel, H.: Scalable emulation of dynamic multi-hop topologies. In: 2008 The Fourth International Conference on Wireless and Mobile Communications, pp. 268–273 (2008). <https://doi.org/10.1109/ICWMC.2008.44>
6. Beuran, R., Tariq, M.I., Miwa, S., Shinoda, Y.: Wireless network performance evaluation through emulation: WiMAX case study. In: 2015 International Conference on Information Networking (ICOIN), Cambodia, pp. 265–270 (2015). <https://doi.org/10.1109/ICOIN.2015.7057894>
7. Ahrenholz, J., Danilov, C., Henderson, T.R., Kim, J.H.: CORE: a real-time network emulator. In: MILCOM 2008 - 2008 IEEE Military Communications Conference, San Diego, CA, USA, pp. 1–7 (2008). <https://doi.org/10.1109/MILCOM.2008.4753614>
8. Ramneek, Choi, W., Seok, W.: Wireless network mobility emulation over wired testbeds: a review. In: 2015 17th International Conference on Advanced Communication Technology (ICACT), Seoul, South Korea pp. 431–435 (2015). <https://doi.org/10.1109/ICACT.2015.7224832>

9. Soles, L.R., Reichherzer, T., Snider, D.H.: Creating a cost-effective air-to- ground network simulation environment. In: Southeast Conference 2015, Fort Lauderdale, FL, USA, pp. 1–5 (2015). <https://doi.org/10.1109/SECON.2015.7132897>
10. Giovanardi, A., mazzini, G.: Emulation architecture implementation and design. In: 2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, Reston, VA, USA, pp. 723–728 (2006). <https://doi.org/10.1109/SAHCN.2006.288537>
11. Giovanardi, A., Mazzini, G.: Ad hoc routing protocols: emulation vs simulation. In: 2005 2nd International Symposium on Wireless Communication Systems, Siena, Italy, pp. 140–144 (2005). <https://doi.org/10.1109/ISWCS.2005.1547673>
12. Giovanardi, G., Mazzini, G., Veronesi, R.: Network emulation in the SAM simulator. In: 2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications, Berlin, Germany, pp. 1302–1306 (2005). <https://doi.org/10.1109/PIMRC.2005.1651651>
13. Maier, S., Grau, A., Weinschrott, H., Rothermel, K.: Scalable network emulation: a comparison of virtual routing and virtual machines. In: 2007 12th IEEE Symposium on Computers and Communications, Las Vegas, NV, USA, pp. 395–402 (2007). <https://doi.org/10.1109/ISCC.2007.4381529>
14. Mehta, D., Jaeger, J., Faden, A., Hebert, K., Yazdani, N., Yao, H.: A scalable architecture for emulating dynamic resource allocation in wireless networks. In: MILCOM 2009 - 2009 IEEE Military Communications Conference, Boston, MA, USA, pp. 1–7 (2009). <https://doi.org/10.1109/MILCOM.2009.5379801>
15. Balasubramanian, D., Dubey, A., Otte, W.R., Emfinger, W., Kumar, P.S., Karsai, G.: A rapid testing framework for a mobile cloud. In: 2014 25th IEEE International Symposium on Rapid System Prototyping, New Delhi, India, pp. 128–134 (2014). <https://doi.org/10.1109/RSP.2014.6966903>
16. To, M.A., Cano, M.: DOCKEMU – a network emulation tool. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, pp. 593–598. <https://doi.org/10.1109/WAINA.2015.107>