





A Hybrid Virtualization Approach to Emulate Heterogeneous Network Nodes

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 Avenue no. 2006, Chengdu, China
{laijy, tianjq, jiangdd, sunjm, zhangk}@uestc.edu.cn

² Science and Technology on Communication Networks Laboratory, Shijiazhuang, China

Abstract. In the last decade, various resource virtualization technologies have been widely applied in ICT industry, particularly the cloud computing domain. These virtualization technologies can squeeze out hardware potential and consequently can save expenditure. Virtualization technologies are used in the network emulation domain to emulate network nodes, which could be quite heterogeneous in terms of hardware architecture. Currently, many network emulators utilize x86 based virtual machines (VMs) to emulate target network nodes of heterogeneous architectures, i.e. ARM, SPARC, PPC, etc., which may introduce incompatibility to the original system and application software of the target nodes, and will consequently jeopardize the emulation fidelity. This paper focuses on alleviating the emulation incompatibility caused by node heterogeneity. Firstly, this emulation incompatibility problem is investigated and analyzed. Then, a hybrid virtualization approach to emulate heterogeneous nodes is elaborated and implemented in a cloud-based network emulation system. A case study of applying the proposed approach to emulate a space-ground integrated network (SGIN) is conducted. Functional verification and performance evaluation experiments lead to the results, which show the hybrid approach can effectively dispose of the incompatibility problem with an affordable performance degradation.

Keywords: Resource virtualization · Network emulation · Heterogeneous nodes · Incompatibility · Space-ground integrated network

1 Introduction

Modern networks are getting increasingly more complicated. Mathematic models can not accurately evaluate network performance anymore. Consequently, various network testing methods are regarded as more decent functional verification and performance evaluation solutions for network architectures, protocols, and upper layer applications. Computer simulation, live test-bed, and network emulation are the major three networking testing methods, among which network emulation is the focus of this work. Network emulation is a technique for testing the real protocols and applications over a

virtual network, which varies from network simulation where abstractly mathematical models of traffic, network, channels and protocols are applied. A network emulator's major task is to emulate nodes and links with medium cost and high fidelity.

Regarding to network node emulation, physical machines are traditionally utilized to represent the target network nodes, with high cost and low scalability. As various resource virtualization technologies are booming in ICT industry, it is nature to use VMs to emulate network nodes. In real network, heterogeneous nodes, such as x86, ARM, PowerPC, Sparc, and MIPS architected, coexist. Many legacy emulator adopts x86 architected VM to emulate all the nodes, which introduces incompatibility to the original protocol and application software of the target nodes. For the sake of high fidelity emulation, it is vital to elaborate a practical approach to accurately emulate heterogeneous nodes in target networks. The contribution of this work contains the following three points.

- A hybrid virtualization approach to emulate heterogeneous network nodes;
- Practically implementing the proposed hybrid virtualization approach in a cloud-based network emulation system;
- Conduct a case study on SGIN emulation, to illustrate the effectivity of the proposed hybrid virtualization approach.

This paper proceeds as follows. Section 2 discusses the state-of-the-art, followed by the briefing of the involved cornerstone technologies in Sect. 3. The hybrid virtualization approach to emulate heterogeneous nodes is derived and implemented in Sect. 4. Then, a case study is presented in Sect. 5. Finally, Sect. 6 concludes the paper and provides the outlook.

2 Related Work

Resource virtualization technologies have gradually been adopted to emulate network nodes to dig up hardware potential, and to lower emulation cost. In industry, Boeing Phantom Works in 2008 [1], presented 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. CORE used FreeBSD jail mechanism to form lightweight VMs. In 2015, Northrop Grumman Aerospace Systems [2], researched on the need to provide a method for studying the interaction among diverse hardware and software components and identifying potential network bottle necks in air-to-ground networks. VMware ESXi is adopted to emulate 10 virtual machines on a single physical server. In the same year, KISTI of Korea [3], presents a critical analysis on existing wired testbeds with respect to the wireless network emulation. Among the investigated testbeds, EMWIN and MobiNet utilize VM technologies to emulate real network nodes. In academia. In 2007, Maier et al. [4] from University of Stuttgart focused on scalable network emulation problems, and present a comparison of different virtual machine implementations (Xen, UML) and their virtual routing approach (NET). In 2009, Mehta et al. [5], described a virtualization technology based emulation architecture that is scalable, modular, and responds to real time changes in topology and link characteristics. In 2014, Balasubramanian et al. [6] from

Vanderbilt University, described a rapid development and testing framework for a distributed satellite system. QEMU (Quick Emulator) virtualization technology is used to launch a configurable number of instances. In 2015, Antonio et al. [7] from Universidad Galileo, presented the Dockemu tool for emulation of wired and wireless networks. The tool glues together state of the art technologies of virtualization, Linux Bridging and NS-3.

Most of the existing emulation schemes rely on x86 architected VMs or containers to emulate heterogeneous nodes, which introduces incompatibility problems to the protocol and application software running on the physical nodes.

3 Cornerstone Technologies

3.1 Virtual Machine

A VM is an emulation of a computer system, and was originally defined as an efficient, isolated duplicate of a real computer. The physical hardware running the VM is generally referred to as the host, and the VMs emulated on that machine are generally referred to as the guests. A host can emulate several guests, each of which may emulate different operating systems and hardware platforms. The software or firmware that creates VMs on the host hardware is called a hypervisor. Typically, the virtualization technologies adopted by a VM include Full virtualization and Para-virtualization.

3.2 KVM

KVM (Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, `kvm.ko`, that provides the core virtualization infrastructure and a processor specific module, `kvm-intel.ko` or `kvm-amd.ko`. KVM supports multiple VMs simultaneously running unmodified Linux or Windows images. By itself, KVM does not perform any emulation. Instead, it exposes the `/dev/kvm` interface to a userspace host. On Linux, QEMU is one such userspace host. QEMU uses KVM when available to virtualize guests at near-native speeds, but otherwise falls back to software-only emulation. Currently, KVM has been ported to S/390, PowerPC, IA-64, ARM, etc., and can support a wide variety of guest operating systems, including Linux, BSD, Solaris, Windows, OS X, Android, etc.

3.3 QEMU

QEMU is a free and open-source emulator that performs hardware virtualization. It emulates the machine's processor through dynamic binary translation and provides a set of different hardware and device models for the machine, enabling it to run a variety of guest operating systems. It also can be used with KVM to run virtual machines at near-native speed, by taking advantage of hardware extensions such as Intel VT. QEMU supports the emulation of various architectures, including: x86, x86-64, MIPS64, SPARC, ARM, SH4, PowerPC, RISC-V, etc. QEMU has two operating modes:

- User mode emulation. QEMU runs single programs that were compiled for a different instruction set. System calls are thunked for endianness and for 32/64 bit mismatches. Fast cross-compilation and cross-debugging are the main targets;
- Full system emulation. QEMU emulates a full computer system, including peripherals. It can be used to provide virtual hosting of several virtual computers on a single computer. QEMU can boot many guest operating systems, including Linux, Solaris, Microsoft Windows, DOS, and BSD; it supports emulating several instruction sets, including x86, MIPS, 32-bit ARMv7, ARMv8, PowerPC, SPARC, ETRAX CRIS and MicroBlaze.

3.4 NFV

NFV (Network functions virtualization) is a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services. A virtualized network function, or VNF, may consist of one or more virtual machines running different software and processes, on top of standard high-volume servers, switches and storage devices, or even cloud computing infrastructure. In network emulation domain, target networks nodes can be regarded as a set of VNFs, and can be emulated by means of NFV.

4 A Hybrid Virtualization Approach to Emulate Heterogeneous Nodes in Target Network

4.1 General Assumptions

Network nodes of various architectures coexist in practical networks. To emulate these heterogeneous nodes, x86 architected VMs are traditionally employed, which however may introduce incompatibility to target nodes of architectures other than x86. For example, using an x86 based VM to emulate a practical ARM node, the system and application software designed for the target node cannot be directly installed on the x86 based VM due to incompatibility. Thus, it is assumed that a target network contains two types of nodes, x86 and non-x86 architected, respectively.

4.2 Principle of the Hybrid Approach

In order to solve the incompatibility problem and to emulate the target network with a sufficiently high fidelity, an innovative hybrid virtualization approach is derived. More precisely, two different visualization technologies are applied simultaneously, which are QEMU-KVM technology for x86 nodes emulation and QEMU-System for non-x86 nodes emulation. Both of them work in the full system emulation mode.

QEMU-KVM is a fork of QEMU supporting using KVM acceleration when the target architecture is the same as the host architecture. Although the fork has already been merged into the QEMU upstream, the term QEMU-KVM is still adopted here to refer the technology to emulate X86 nodes. To use KVM, just pass `-enable-kvm` to

QEMU. QEMU-System technology implements full platform virtualization, including one or several processors and various peripherals. It runs without a host kernel driver and yet gives acceptable performance. It uses dynamic translation to native code for reasonable speed. Therefore, it can be used to emulate non-x86 nodes in the target network. For example, one can use the executable `qemu-system-sparc` to simulate the sparc architected machines. Figure 1 illustrates the principle of the hybrid virtualization approach.

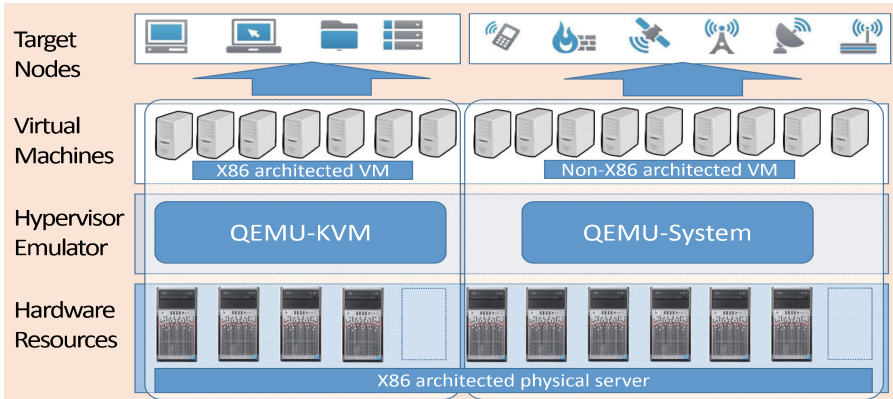


Fig. 1. The principle of the hybrid virtualization approach.

4.3 Practical Implementation

The hybrid virtualization approach is practically implemented in an innovative cloud-based network emulation system, which introduces currently prevalent cloud computing and related ICT technologies including resource virtualization, NFV, SDN, traffic control and flow steering to the network emulation domain, so to provide users Network Emulation as a Service (NEaaS). The emulation system can be deployed on either public or private cloud to satisfy diverse user needs. The network emulation principle is to utilize the VMs created and allocated by the cloud platform to imitate network nodes. Emulating network links relies on using the virtual network links in the cloud platform. To emulate network topology is to dynamically control and adjust the virtual network links between VMs in a fast enough manner to satisfy the emulation needs. The architecture of the cloud-based network emulation system is designed and presented in Fig. 2, which is divided into four layers, namely, resource virtualization, cloud computing, emulation core, and emulation interface layers. The key points to deploy the proposed hybrid virtualization approach in the aforementioned cloud-based network emulation system, is to modify the lowest two layers, namely resource virtualization layer and cloud computing layer.

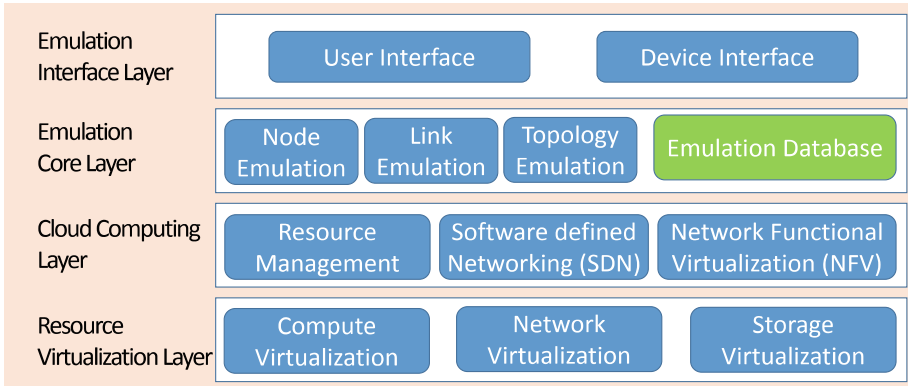


Fig. 2. Architecture of the cloud-based emulation platform.

In the resource virtualization layer, the xml file adopted by Libvirt to depict and to create heterogeneous VMs should be established at first. Then, for some heterogeneous architectures, OS Kernel and Initrd files should be loaded externally when creating VM instances. In the cloud computing layer, Legacy Openstack lacks the ability to manage heterogeneous VMs. Therefore, the first step of the implementation is to create heterogeneous VM images which can be recognized by Openstack. Then, front end configuration should be carried out, including updating the supported architecture list of each compute node in the MySQL database. In the meanwhile, Openstack API i.e., nova-metadata is called to configure the hardware information of heterogeneous VM images. At last, the back-end to manage the heterogeneous VMs is also implemented, with the logical demonstrated in Fig. 3, where `os_kernel` and `os_initrd` are the parameters passing addresses to the aforementioned xml files.

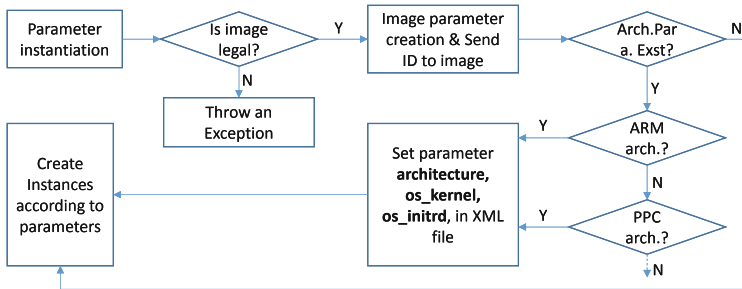


Fig. 3. Implementation schematic diagram.

5 Case Study: Applying the Hybrid Approach to Emulate a Space-Ground Integrated Network

5.1 Emulation Scenarios and Assumptions

A SGIN is considered as the target network to be emulated. The topology of a typical SGIN is presented in Fig. 4. As is shown, in the space section, three Sparc-architected GEO satellite nodes connected with each other forms a ring as the backbone network. Each of the GEO satellite node covers a wide area of earth surface, and connects with a large number of heterogeneous terminal nodes on the ground. To simplify the emulation scenario, this case study only concerns three ground terminal nodes (each with a different architecture) for each GEO node, namely one ARM, one PPC, and one x86 terminal node. In practice, there might be thousand or tens of thousands terminal nodes served by each single GEO satellite. Besides, a gateway station node providing access to other ground networks is connected with one of the GEO satellites; it is assumed to be x86 architecture based. The emulated SGIN network is built in the cloud-based emulation system, with applying the proposed hybrid virtualization. Particularly, the x86 nodes are emulated by QEMU-KVM technology, while the sparc nodes, ARM nodes and PPC nodes utilize QEMU-System technology.

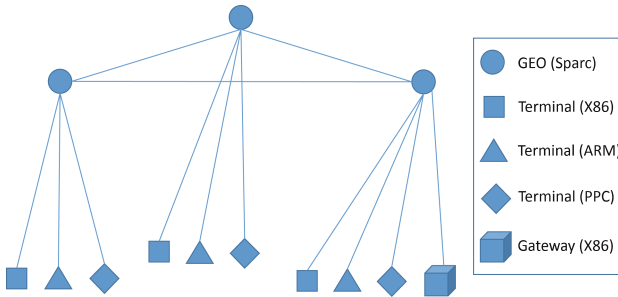


Fig. 4. Schematic diagram.

5.2 Functional Verification and Performance Evaluation

Functional verifications are carried out as follows. Firstly, connectivity between two arbitrarily chosen nodes is tested by using ping tool. The results show no difference between scenarios with and without applying the proposed hybrid scheme. Secondly, real-time video streaming function is also verified in the emulated network. Two heterogenous nodes are randomly chosen, and on which VLC server and client are deployed, respectively. A H.264 encoded video file is streamed from the server to the client, and again, no big difference is observed between scenarios with and without using the proposed approach.

Comparative performance evaluations are carried on for the scenarios with and without using the proposed approach. The VM profiles for the two different scenarios are given in Table 1. Both computation and networking performance are considered in this case study. On one hand, for computation performance evaluation, Unix Bench is adopted. Table 2 presents the testing results. On the other hand, for the networking performance evaluation, the iperf and ping tools are utilized to measure TCP and UDP bandwidth, packet loss, transmission delay and Jitter. The results are given in Table 3.

Table 1. The VM profiles.

CPU	AMD R1700X	RAM	64 GB DDR4 2400
Network Card	Intel 1000 Mbps * 4	Storage	SSD 480 GB
QEMU	Version 2.5.0	Libvirt	Version 1.3.1
OpenStack	Mitaka	Unixbench	Version 5.1.3
OS	Ubuntu 14.04 LTS	Kernel	4.4.0-96-generic

Table 2. Computational performance.

Metric	Arch.											
	Qemu-kvm				Qemu-system							
	X86				ARM				PowerPC			
Ncpu	1	2	3	4	1	2	3	4	1	2	3	4
TCP BW (G/s)	47.3	49.1	51.3	52.6	1.4	1.4	1.5	1.5	1.9	1.9	2.0	2.1
UDP BW (G/s)	0.81	0.81	0.80	0.81	0.2	0.2	0.2	0.2	0.2	0.1	0.1	0.1
Packet loss (%)	0.46	0.75	0.61	0.44	0	0	0	0	0	0	0	0
Time delay (ms)	0.36	0.29	0.38	0.28	0.7	0.7	0.7	0.7	0.8	0.8	0.9	0.8
Jitter	0.09	0.05	0.10	0.07	0.2	0.2	0.2	0.2	0.3	0.2	0.2	0.2

5.3 Results Discussion

According to the above experimental results, it is no surprised that the heterogeneous VMs emulated by the QEMU-System technology are with much lower computation and networking performance, attributed to the fact that the software-based instruction translation is much slower than the hardware-assistant virtualization. Considering the fact that only part of the target nodes will be emulated by QEMU-System technology, and the fact that QEMU-KVM starts to support virtualize more architectures, the cost of the proposed hybrid approach is still affordable.

Table 3. Network performance.

Metric	Arch.											
	Qemu-kvm				Qemu-system							
	X86				ARM				PowerPC			
Ncpu	1	2	3	4	1	2	3	4	1	2	3	4
String manipulation	204.4	200.6	583.8	745.2	166.7	169.6	170.8	168.8	181.9	193.4	219.0	215.2
Floating point processing	31.2	249.2	221.9	193.1	55.8	67.1	73.6	82.4	24.1	25.1	23.2	24.8
R/W	34.8	34.1	88.6	110.2	11.5	13.3	13.9	14.4	55.9	56.4	56.8	56.7
File replication (small)	976.7	980.1	1567.9	1492.6	75.6	72.2	54.6	78.4	40.7	36.1	37.6	31.6
File replication (middle)	1601.6	1585.5	2526.1	2380.7	61.1	118.4	87.0	87.8	117.0	65.0	87.3	52.8
File replication (large)	3249.5	3229.4	5718.6	5511.6	169.6	137.6	172.2	194.5	151.2	147.5	143.0	138.3
Process communication	1049.3	1036.3	3043.2	3886.9	98.9	97.8	96.5	101.6	33.8	36.3	35.3	25.1
Process context switching	730.9	617.0	1889.6	2398.3	19.2	21.0	21.3	22.4	22.2	21.8	22.4	23.7
Process creation	137.3	138.3	342.1	497.1	56.8	61.9	68.5	59.7	57.8	62.4	64.3	67.8
Single script operation	93.2	90.1	240.0	298.6	33.0	33.2	33.2	32.4	100.5	102.1	113.1	15.2
Multi script operation	86.3	74.8	220.9	271.2	28.7	28.0	26.5	24.6	90.5	91.6	90.2	93.1
System call	578.1	569.5	1585.9	3023.5	255.7	256.9	252.4	255.4	31.5	32.3	23.2	31.8
Score	300.9	343.9	769.4	876.9	59.4	64.3	63.8	64.8	57.2	60.1	59.0	63.1

6 Conclusion

Network emulation is regarded as the most promising network testing method attributed to its low cost and high fidelity features. Most existing network emulators rely on x86 based VMs or containers to imitate network nodes of heterogeneous architectures, such as ARM, PowerPC, Sparc, MIPS, etc., which could bring incompatibility problems to the protocol and application software running on the real nodes. This paper focused on solving this incompatibility problem and further improving emulation fidelity. To that end, the paper designed an innovative hybrid virtualization approach to emulate the heterogeneous nodes in target networks. The proposed hybrid approach was also implemented in a practical cloud-based network emulation platform. A Case study on emulating an SGIN illustrated that the hybrid virtualization approach effectively and efficiently eliminates the incompatibility problem in practical scenarios with an affordable performance degradation.

The planned work for the next step mainly focus on the cost of VMs. Compared to the currently booming light-weighted container technologies, such as Docker, VM's cost is still too high, which promotes the authors to investigate replacement of the VMs by containers for more efficient network emulations.

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 & No. 61872051).

References

1. Ahrenholz, J., Danilov, C., Henderson, T.R., Kim, J.H.: CORE: a real-time network emulator. In: 2008 IEEE Military Communications Conference, MILCOM 2008, San Diego, CA, USA, pp. 1–7 (2008)
2. Soles, L.R., Reichherzer, T., Snider, D.H.: Creating a cost-effective air-to-ground network simulation environment. In: SoutheastCon 2015, Fort Lauderdale, FL, USA, pp. 1–5 (2015)
3. Ramneek, T., 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 (2015)
4. 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)
5. Mehta, D., Jaeger, J., Faden, A., Hebert, K., Yazdani, N., Yao, H.: A scalable architecture for emulating Dynamic Resource Allocation in wireless networks. In: 2009 IEEE Military Communications Conference, MILCOM 2009, Boston, MA, USA, pp. 1–7 (2009)
6. 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)
7. To, M.A., Cano, M.: DOCKEMU – a network emulation tool. In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, Gwanju, South Korea, pp. 593–598 (2015)