Postprint of Task Partitioning for Network Simulation Integrating Virtualization and Digital Simulation

Authors: Wu Wenyan, Jiang Xin, Wang Xiaofeng, Liu Yuan

Date: 2018-05-18T00:00:00+00:00

Abstract

To enhance network simulation performance, this study investigates a network simulation task partitioning method for an architecture that integrates virtualization and digital simulation. By comprehensively leveraging the respective advantages of virtualization and digital simulation, the network topology is partitioned into virtualization topology regions and digital simulation topology regions. Based on given physical resources, a fusion-based partitioning of these two regions is performed with the dual objectives of load balancing and remote communication volume minimization. Experimental results demonstrate that, compared with random algorithms and balanced load balancing algorithms, the proposed method reduces remote communication volume by 33.7% and 25.1% on average, respectively, while improving load balancing degree by 56.3% and 38.0% on average, respectively. The proposed method can effectively reduce remote communication volume and enhance load balancing degree.

Full Text

Task Dividing of Network Emulation for Fusion of Virtualization and Digital Simulation

Wu Wenyan[1] , Jiang Xin[2], Wang Xiaofeng[1] , Liu Yuan[1] 1. Jiangnan University, a. School of Internet of Things Engineering; b. School of Digital Media, Wuxi, Jiangsu 214122, China 2. Jiangnan Institute of Computing Technique, Wuxi, Jiangsu 214083, China

Abstract: To improve network emulation performance, this paper investigates a network emulation task partitioning method for the architecture that fuses virtualization and digital simulation. By comprehensively considering the respective advantages of virtualization and digital simulation, the network topology is divided into virtualization topology areas and digital simulation topology areas.

Combined with given physical resources, the fusion partitioning of these two areas aims to achieve load balancing and minimize remote communication traffic. Experimental results demonstrate that compared with random algorithms and uniform load balancing algorithms, this method reduces remote communication traffic by 33.7% and 25.1% on average, respectively, while improving load balancing degree by 56.3% and 38.0% on average, respectively. This method can effectively reduce remote communication traffic and enhance load balancing degree.

Keywords: virtualization; digital simulation; fusion emulation; task dividing; remote traffic; degree of load balancing

0 Introduction

Network technology research and information system security evaluation platforms provide strong support for computer technology and network security assessment, with network emulation technology serving as their cornerstone. However, how to further improve the large-scale capability and high fidelity of network research experiments remains an urgent problem. In response to this challenge, network emulation technology that fuses virtualization and digital simulation has emerged as a new trend in network research. For instance, literature [1] integrated distributed network simulation technology with virtualization-based network emulation technology to implement an experimental platform combining simulation and emulation, focusing on synchronization issues between network simulation and network emulation. Literature [2] addressed the scalability, extensibility, and fidelity issues of router emulation in network experiments, proposing a design method for a virtualized routing emulation platform by combining virtualization technology and digital simulation technology. Literature [3,4] designed a network emulation quasi-experimental testbed by virtualizing physical resources and deploying functional network nodes and simulated communication link entities at the virtual machine level. Literature [5] proposed a network test platform integrating virtualized emulation systems and parallel discrete-event network simulators, designing a globally synchronized algorithm to manage the transition between virtual time and simulation time.

Facing the performance requirements of large-scale network emulation, achieving reasonable and effective network emulation task partitioning methods to ensure effective mapping between the topology to be partitioned and the computing cluster, reducing communication overhead among computing clusters based on rational utilization of physical resources, and satisfying load balancing are critical. Literature [6] comprehensively considered specific homogeneous computing environments and network models to be simulated, taking load balancing, communication overhead, and synchronization overhead minimization as partitioning objectives. Literature [7] focused on analyzing how to effectively evaluate the packet forwarding volume of each network node in the network topology to be simulated, and studied load balancing methods for packet forwarding simulation tasks in distributed network emulation platforms based on

this analysis. Literature [8] studied network simulation topology partitioning methods based on abstraction reduction to balance packet forwarding simulation tasks, reduce the number of remote links, and improve network topology partitioning efficiency. Literature [9] proposed a load balancing partitioning algorithm based on polynomial time complexity, which can minimize communication consumption while balancing loads. Literature [10] proposed a virtual network mapping algorithm based on virtual topology pre-configuration and reusable technology to improve mapping fairness, dividing the virtual network mapping process into two steps: topology pre-configuration and mapping. This algorithm outperforms similar algorithms in terms of physical network resource utilization, revenue/cost ratio, and virtual network acceptance fairness. Literature [11] proposed an intelligent cloud testing platform automatic topology mapping implementation method by combining ATF (automation test framework) topology mapping in practice to strengthen centralized management of testing resources. This topology mapping algorithm has good scalability and confidentiality, with significant improvements in discovery efficiency, accuracy, and effectiveness. Literature [12] addressed energy consumption issues in network virtualization environments by establishing a network topology-consistent and energy-efficient virtual network mapping model based on network topology attributes and energy consumption characteristics of physical nodes and network devices in the underlying physical network. This model maps adjacent virtual nodes to adjacent physical nodes with smaller energy consumption increases while coordinating the use of minimum energy consumption routing algorithms to map virtual links to physical paths with minimum energy consumption, effectively increasing the number of dormant physical nodes and physical links and significantly reducing the resource cost and system energy consumption of virtual network mapping.

The above network emulation task partitioning methods consider both physical resource utilization and emulation performance, but they are all based on single virtualization or digital simulation technology and do not consider the partitioning of network topology that fuses virtualization and digital simulation. However, in actual network environments, a network topology often contains more than one type of network node, and different network mapping methods are used for different network nodes during network emulation experiments. Therefore, fusion network topology partitioning methods become particularly important.

1.2 Performance Analysis of Virtualization and Digital Simulation Fusion Network Emulation

Virtualization and digital simulation fusion network emulation technology deploys fused network topology based on the OpenStack cloud platform. This paper uses KVM [13] (kernel-based virtual machine) virtualization technology to deploy virtualized nodes. KVM is deployed in the Linux kernel and can directly interact with hardware in any scenario without modifying the virtualized operating system, facilitating convenient control of virtualization processes. NS3

[14-16] simulation technology is used to deploy digital simulation nodes. NS3 is a highly distinctive new network simulator with completeness, open-source characteristics, ease of use, and extensibility, making it superior to most existing mainstream network simulators.

Figure 2 [Figure 2: see original paper] shows the throughput comparison chart between KVM nodes and NS3 nodes in OpenStack cloud platform experiments. The experimental results demonstrate that KVM node throughput is far superior to NS3 node throughput. Figure 3 [Figure 3: see original paper] shows the comparison of the number of KVM nodes (with 4G memory and 2 logical CPUs) and NS3 nodes that can be launched on a computing node configured with 16G memory and 24 logical CPUs. The experimental results show that NS3 node scalability is far superior to KVM nodes.

In summary, KVM offers high throughput, but the number of KVM nodes that can be launched on a computing node is limited by memory size and logical CPU count, so a computing node can launch relatively few KVM nodes (on the order of tens). While NS3 has lower throughput, the number of NS3 nodes that can be launched is also limited by logical CPU count, but memory size has almost no impact on the number of launchable nodes. A high-performance computing node can simultaneously run thousands of NS3 simulation nodes. Therefore, KVM fidelity is superior to NS3, but NS3 scalability is superior to KVM. Consequently, virtualization and digital simulation fusion network emulation technology can effectively solve the fidelity and scalability issues of emulation networks.

1.1 Virtualization and Digital Simulation Fusion Network Emulation Architecture

As shown in Figure 1 [Figure 1: see original paper], the virtualization and digital simulation fusion network emulation architecture is based on the OpenStack cloud platform. The basic architecture of OpenStack includes one control node, one network node, and several computing nodes. The control node is responsible for system management and control functions, the network node provides network services such as DHCP (dynamic host configuration protocol) services and routing services, and computing nodes provide resources for establishing virtual hosts. Conducting network emulation oriented toward the fusion of virtualization and digital simulation scales on the OpenStack cloud platform, the key to achieving fused network emulation lies in solving problems such as virtual-real interconnection, routing configuration, and operational protocols among virtualization, digital simulation, and the OpenStack platform. Virtual-real interconnection technology can achieve seamless connectivity among virtualization, digital simulation, and the OpenStack platform, while routing configuration and the TCP/IP protocol suite are the foundation for normal packet storage and forwarding during fusion emulation.

In the network emulation topology to be partitioned in this paper, non-terminal routing nodes are mapped by virtualization nodes, while terminal routing nodes

and host nodes are mapped by digital simulation nodes. Moreover, the purpose of virtualization and digital simulation fusion network emulation task partitioning is to achieve load balancing and minimize communication overhead throughout the entire computing environment. The communication overhead here mainly refers to communication overhead between remote links, because in distributed network emulation, the network topology is divided into smaller topologies. Links between the original network topologies are separated and assigned to different computing nodes as remote links, significantly increasing overhead during packet forwarding.

1.3 Problem Description of Fusion Network Emulation Task Partitioning

A network topology contains different node types, which can be generally divided into routing nodes and host nodes. Routing nodes can be further divided into terminal routing nodes and non-terminal routing nodes. Terminal routing nodes in a network topology, after removing links connected to host nodes, are only connected to one other node in the topology (i.e., degree = 1). Different mapping methods can be used for different node types in a network topology. The algorithm in this paper mainly considers applicable mapping methods for different node types from two aspects: throughput and scalability.

Based on the advantages and disadvantages of KVM nodes and NS3 nodes, this paper proposes a virtualization and digital simulation fusion network emulation task partitioning method. According to actual network conditions, terminal routing nodes and host nodes in the network topology to be partitioned have relatively small communication traffic and load, while non-terminal routing nodes have relatively large communication traffic and load, and the number of host nodes far exceeds the number of routing nodes. Based on these characteristics, terminal routing nodes and host nodes in the network topology to be partitioned are mapped using NS3, while non-terminal routing nodes are mapped using KVM.

The virtualization and digital simulation fusion network emulation task partitioning method fully considers the differences among different nodes and the communication requirements of links composed of different nodes. Based on saving physical resources as much as possible, it comprehensively considers load balancing of computing nodes and remote link communication traffic among computing clusters from the perspective of the overall computing environment, and uses this as the basis for rapid and accurate partitioning of network emulation topology.

2 Virtualization and Digital Simulation Fusion Network Emulation Task Partitioning Method

2.1 Distributed Network Emulation Task Partitioning

The principle of network emulation task partitioning is to divide the network emulation topology into multiple subnets, with each subnet' s emulation tasks assigned to different computing nodes for execution, while communication between subnets is realized through real links between computing nodes. As shown in Figure 4 [Figure 4: see original paper], this is a simple distributed network emulation task partitioning model that describes the network emulation task partitioning problem in a homogeneous computing environment. In Figure 4, N represents the total number of topology nodes in the network topology to be emulated, and W represents the sum of loads of each topology node in the network topology to be emulated (the load of a topology node is the weight value of that node). Assuming there are M computing nodes in the homogeneous computing environment, represented by computing nodes 1~M. After partitioning the total network topology, computing nodes 1~M are assigned n ~n topology nodes respectively, and the sum of loads of assigned topology nodes are w ~w respectively. The number of topology nodes assigned to each computing node reflects the routing emulation task volume allocated to that computing node, while the sum of loads of topology nodes assigned to each computing node reflects the packet forwarding emulation task volume allocated to that computing node.

2.2 Virtualization and Digital Simulation Fusion Network Emulation Task Partitioning Algorithm

The steps of the virtualization and digital simulation fusion network emulation task partitioning algorithm are as follows. During partitioning, only terminal routing nodes and non-terminal routing nodes are partitioned; host nodes are partitioned into the same results as the terminal routing nodes they are connected to.

Assume there is a network topology with N routing nodes, numbered 1~N. Create matrix Topo[N][N] to store the relative load value information of the network topology. For example, if there is a link between routing node i and routing node j, the relative load value of this link is stored in Topo[i-1][j-1]; if there is no link, the value of Topo[i-1][j-1] is 0. Topo[i-1][i-1] stores the relative load value of routing node i, which is the sum of the load values of links connected to this routing node.

Input: Weighted undirected graph G(V, E), where vertex set V = {v , ···, v , ···, v }, vertex weight w (v), and edge weight w (v , v).
Number of terminal routing nodes: Z.
Number of non-terminal routing nodes: F.
Computing capability of computing nodes (packet forwarding simulation capability): K.

Virtualization capacity of computing nodes (maximum number of virtual machines that can be launched): Q.
Number of computing nodes (number of subnets): M.
Output: Partitioning result {G , ···, G , ···, G }.

Algorithm Steps:

1. Initialize matrix Topo[N][N]. Topo[i-1][j-1] stores the edge weight value of the link between routing nodes i and j, and Topo[i-1][i-1] stores the vertex weight value of routing node i.

2. Traverse the entire network topology to find terminal routing nodes in the topology, record the Z value, set the vertex weight of terminal routing nodes to 1, set the edge weight of edges connecting terminal routing nodes to 1, and finally record them in matrix Topo[N][N] to ensure that the relative load values of terminal routing nodes do not affect the partitioning results of non-terminal routing nodes during initial METIS partitioning.

3. Traverse the entire network topology to find non-terminal routing nodes, record the F value. For each non-terminal routing node, record the edge weight of each connected edge in matrix Topo[N][N]. The sum of edge weights is the vertex weight of this routing node, which is also recorded in matrix Topo[N][N].

4. Determine the number of topology partitions, i.e., the number of computing nodes. Let A = F/K (A takes an integer value; if there is a decimal, remove the decimal and add 1). Let B = F/Q (B takes an integer value; if there is a decimal, remove the decimal and add 1). The number of topology partitions is M = max{A, B}.

5. Initialize matrix R[M][N] to record partitioning results.

6. Obtain initial partitioning based on METIS. METIS can obtain an initial partitioning with relatively balanced packet forwarding simulation task loads and smaller remote communication overhead based on the data in matrix Topo[N][N], according to the packet forwarding simulation capability K of computing nodes and the number of computing nodes M. Set loop = 0.

7. Conduct verification. If the sum of routing node weights on each computing node does not exceed the computing capability K of computing nodes (threshold 1) and the number of routing nodes on each computing node is less than or equal to the maximum number of virtualized nodes Q that can be launched on a computing node (threshold 2), input the partitioning result into R[M][N] and proceed to step 8. If any of the above two conditions is not satisfied, proceed to step 11.

8. Traverse the topology to partition terminal routing nodes into the partitioning results of the non-terminal routing nodes they are connected to, and proceed to step 16. This is done because terminal routing nodes have

relatively small loads that will not significantly affect the load sum of computing nodes or cause load imbalance or exceed the K value (threshold 1). Moreover, terminal routing nodes are digital simulation nodes not limited by the Q value (threshold 2), and this approach can also reduce remote communication overhead among computing clusters.

9. Input partitioning results of computing nodes whose sum of routing node weights does not exceed threshold 1 and whose number of routing nodes equals threshold 2 into R[M][N]. Set the vertex weights and edge weights of routing nodes in qualified partitioning results to 1 and record them in Topo[N][N]. If there are edges between routing nodes in Topo[N][N] and routing nodes in R[M][N], the edge weights of connected edges become 1, and the vertex weights of these routing nodes are recalculated. If there are new changes in Topo[N][N], indicating new qualified partitioning results, return to step 6 for re-partitioning; otherwise, proceed to step 10.
10. Set loop = loop + 1, return to step 6 until loop = 5 (i.e., perform partitioning with unchanged results 5 times) and there are still unqualified partitioning results, then proceed to step 11 for fine-tuning.
11. Select the set of topology nodes Vc on the subnetwork that exceeds the threshold (any threshold). Nodes in this set must have links with some node in the computing node to be received (below all thresholds). If no link exists, Vc is the set of all topology nodes on the subnetwork that exceeds the threshold.
12. For Vc, select the topology node with the smallest vertex weight to form set V' c (there may be multiple topology nodes with the smallest vertex weight). The reason for selecting the topology node with the smallest vertex weight is to minimize the impact of topology node migration on the load balance of packet forwarding simulation tasks.
13. Select the topology node vm in V' c that has the smallest sum of link weights with other topology nodes in the subnetwork to be migrated (exceeding the threshold) (if multiple nodes meet the condition, randomly select one). This selection ensures that the new remote communication overhead added after topology node migration is minimized.
14. Select vm for node migration.
15. Return to step 7 for verification.
16. Output partitioning result {G , ···, G , ···, G }.

Algorithm 1: Uniform Load Balancing Algorithm (ULB)
For a given computing environment, METIS partitioning can obtain partitioning results with relatively balanced loads for packet forwarding simulation tasks and routing simulation tasks. The METIS partitioning algorithm oriented toward load balancing of packet forwarding simulation tasks and routing simulation tasks is called the ULB algorithm.

Definition 1: Remote Link Traffic (RLT). Used to evaluate the remote link communication traffic of packet forwarding simulation tasks on each computing node during network topology emulation operation. The smaller the RLT value, the smaller the remote link communication traffic of computing nodes obtained based on the algorithm.

Definition 2: Total Remote Link Traffic (TRLT). The cumulative sum of remote link communication traffic of each computing node is the total remote link traffic of the computing cluster. The smaller the TRLT value, the smaller the total remote link traffic of the computing cluster obtained based on the algorithm.

$$\mathrm{TRLT} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \mathrm{Topo}[i][j]$$

Definition 3: Algorithm Load Balancing Degree (DLB). Used to evaluate the comprehensive load balancing situation of routing simulation tasks on each computing node during network topology emulation operation. The closer this value is to 0, the more balanced the routing simulation task load of the computing environment. It is described by the following formula:

$$\mathrm{DLB} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{x_i - \bar{x}}{\bar{x}} \right|$$

Since the experimental topology is a virtualization and digital simulation fusion network topology, and the throughput of digital simulation nodes is far lower than that of virtualized nodes, this paper mainly considers the impact of virtualized routing nodes on the load balancing of routing simulation tasks. Therefore, in the formula, n represents the scale of the computing cluster used, x represents the number of virtualized routing nodes deployed on each computing node, and $\bar{x}$ represents the expected number of virtualized routing nodes deployed on computing nodes. The smaller the value calculated by the load balancing degree formula, the higher the load balancing degree; conversely, the larger the value, the lower the load balancing degree.

3 Experimental Validation and Analysis

Experimental Environment: One OpenStack platform consisting of 1 control node, 1 network node, and 8 computing nodes. All computing nodes are R730 servers with CPU type of 2 Intel E5620 processors, 24 logical CPUs, 16G memory configuration, and throughput of 22000M/s.

Topology Scenarios: Specifically, three groups of network topologies are generated using topology generation tools. The first group has 2026 network nodes,

including 26 routing nodes numbered 1~26, among which there are 22 non-terminal routing nodes and 4 terminal routing nodes, with each terminal routing node connected to 500 host nodes. The second group has 4027 network nodes, including 27 routing nodes numbered 1~27, among which there are 19 non-terminal routing nodes and 8 terminal routing nodes, with each terminal routing node connected to 500 host nodes. The third group has 2530 network nodes, including 30 routing nodes numbered 1~30, among which there are 25 non-terminal routing nodes and 5 terminal routing nodes, with each terminal routing node connected to 500 host nodes. Based on the above experimental environment, the three groups of generated network topologies are deployed in this homogeneous computing environment.

The algorithm in this paper is a virtualization and digital simulation fusion network emulation task partitioning algorithm. Due to the uniqueness of this algorithm, host nodes have no impact on remote link communication traffic, so the partitioning of network topology focuses on routing nodes in the topology.

Algorithm 1: Uniform Load Balancing Algorithm (ULB)
For a given computing environment, METIS partitioning can obtain partitioning results with relatively balanced loads for packet forwarding simulation tasks and routing simulation tasks. The METIS partitioning algorithm oriented toward load balancing of packet forwarding simulation tasks and routing simulation tasks is called the ULB algorithm.

Based on the given experimental environment and network topology, the virtualization and digital simulation fusion network emulation task partitioning algorithm is used for topology partitioning. According to the partitioning results, deployment is implemented on the OpenStack platform. To verify that the virtualized and digital simulation fusion network topology deployed on the OpenStack platform can communicate like a real network topology, ping commands are executed between different hosts in the fusion network topology. The results show that hosts in the fusion network topology can ping each other. From the test results, the algorithm proposed in this paper can quickly partition a given fusion network topology, and based on the partitioning results, the fusion network topology can be deployed on the OpenStack platform with high fidelity.

The total remote link traffic of the fusion algorithm' s running results is compared with the partitioning results of the random algorithm and the ULB algorithm. Figure 5 [Figure 5: see original paper] shows the comparison of total remote link traffic of computing clusters obtained by the fusion algorithm, random algorithm, and ULB algorithm for the three groups of network topologies. Specifically, for the first group, TRLT = 16206 for the fusion algorithm, TRLT = 30064 for the random algorithm, and TRLT = 23230 for the ULB algorithm. For the second group, TRLT = 12296 for the fusion algorithm, TRLT = 19646 for the random algorithm, and TRLT = 17368 for the ULB algorithm. For the third group, TRLT = 27570 for the fusion algorithm, TRLT = 33634 for the random algorithm, and TRLT = 33010 for the ULB algorithm. Calculations show that compared with the random algorithm and ULB algorithm, the fusion

algorithm reduces the total remote link traffic of partitioning results by 33.7% and 25.1% on average, respectively.

The experimental results demonstrate that the fusion algorithm is an optimized network task partitioning method that can reduce the total remote link traffic among computing clusters. Figures 6 [Figure 6: see original paper]~8 [Figure 8: see original paper] show the remote link traffic comparison charts of each computing node for the three groups of network topologies based on different algorithms. The comparison between Figure 5 [Figure 5: see original paper] and Figures 6~8 shows that although the remote link traffic of each computing node obtained through fusion algorithm partitioning may not necessarily be smaller than that of random algorithm or ULB algorithm partitioning results, the total remote link traffic of the computing cluster is significantly smaller than the latter two. Moreover, because the algorithm in this paper is a virtualization and digital simulation fusion network emulation task partitioning algorithm, the partitioning results obtained based on this algorithm can save computing resources. As shown in Figure 6, the random algorithm and ULB algorithm partitioning results both use 1 more computing node than the fusion algorithm partitioning result. As shown in Figure 7 [Figure 7: see original paper], the random algorithm and ULB algorithm partitioning results both use 2 more computing nodes than the fusion algorithm partitioning result. As shown in Figure 8 [Figure 8: see original paper], the random algorithm and ULB algorithm partitioning results both use 1 more computing node than the fusion algorithm partitioning result.

In addition to remote link traffic, this paper also calculates the load balancing degree of the computing cluster. For the first group, DLB = 0.22 for the fusion algorithm, DLB = 0.41 for the random algorithm, and DLB = 0.39 for the ULB algorithm. For the second group, DLB = 0.16 for the fusion algorithm, DLB = 0.58 for the random algorithm, and DLB = 0.37 for the ULB algorithm. For the third group, DLB = 0.53 for the fusion algorithm, DLB = 1.08 for the random algorithm, and DLB = 0.61 for the ULB algorithm. The comparison is shown in Figure 9 [Figure 9: see original paper]. Calculations show that compared with the random algorithm and ULB algorithm, the fusion algorithm improves the load balancing degree of partitioning results by 56.3% and 38.0% on average, respectively.

The superiority of the virtualization and digital simulation fusion network emulation task partitioning algorithm lies not only in optimizing METIS partitioning results through node migration to reduce the load balancing degree of the fusion algorithm, but also in partitioning terminal routing nodes into the partitioning results of non-terminal routing nodes connected to them, eliminating the impact of terminal routing nodes on remote link traffic and reducing the remote link traffic of the fusion algorithm. Additionally, the running time for partitioning the above three groups of network topologies using the virtualization and digital simulation fusion network emulation task partitioning method is less than 1 second. This shows that the time complexity of the algorithm is not high and

can be applied to larger-scale network topologies.

4 Conclusion

Network emulation is currently the main method for network research, and virtualization and digital simulation fusion network emulation is a new trend in network research. This paper focuses on the virtualization and digital simulation fusion network emulation task partitioning algorithm, which can effectively reduce remote link communication traffic among computing clusters and improve the load balancing degree of computing clusters while saving computing resources. In future research, we will focus on the application of virtualization and digital simulation fusion network emulation task partitioning methods in network security experiments.

References

[1] Jin Du, Zheng Yi, Nicol D M. A parallel network simulation and virtual time-based network emulation testbed [J]. Journal of Simulation, 2014, 8(3): 206-219.

[2] Huang Minhuan, Zhang Yaoxue, Xu Fei, et al. Design of routing emulation experimental platform based on virtualization technology [J]. Journal of System Simulation, 2014, 26(8): 1672-1677.

[3] Huang Jinsong, Yang Yi, Wang Wennai. A network emulation quasi-experimental testbed based on virtualization platform [J]. Computer Technology and Development, 2015, 08(9): 208-212.

[4] He Xinhua, Jin Guozhu, Wang Qiong. Application of virtualization technology in simulation experiments [J]. Ordnance Equipment Engineering Journal, 2011, 32(8): 71-73.

[5] Gu Yun, Fujimoto R. Applying parallel and distributed simulation to remote network emulation [C]// Proc of Winter Simulation Conference. 2007: 1328-1337.

[6] Xu Ding, Ammar M. BencHMAP: Benchmark-based, hardware and model-aware partitioning for parallel and distributed network simulation [C]// Proc of International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. Washington DC: IEEE Computer Society, 2004: 455-463.

[7] Willinger W, Taqqu M S, Sherman R, et al. Self-similarity through high-variability: statistical analysis of ethernet LAN traffic at the source level [J]. IEEE/ACM Trans on Networking, 1997, 5(1): 71-86.

[8] Peschlow P, Honecker T, Martini P. A flexible dynamic partitioning algorithm for optimistic distributed simulation [C]// Proc of International Workshop on Principles of Advanced and Distributed Simulation. 2007: 273-280.

[9] Grande R E D, Boukerche A, Ramadan H. Distributed re-arrangement scheme for balancing computational load and minimizing communication delays in HLA-based simulations [J]. Concurrency & Computation Practice & Experience, 2013, 25(5): 626–648.

[10] Wang Cong, Yuan Ying, Peng Sancheng, et al. Fair virtual network mapping algorithm based on topology pre-configuration [J]. Journal of Computer Research and Development, 2017, 54(1): 212-220.

[11] Wang Liang, Han Liangang, Xie Xihai. Research on topology mapping algorithm implementation under intelligent cloud testing [J]. Application of Electronic Technique, 2017, 43(3): 116-119.

[12] Peng Limin. Topology-consistent green virtual network mapping algorithm [J]. Small Microcomputer System, 2016, 37(5): 1079-1083.

[13] Qumranet A K, Qumranet Y K, Qumranet D L, et al. kvm: the Linux virtual machine monitor [J]. Proc Linux Symposium, 2012, 10(7): 104-112.

[14] Kumar S, Paul S, Amar A K. Communication in vehicular cloud network using ns3 [J]. International Journal of Control Theory & Applications, 2017, 12(4): 159-167.

[15] Riley G F, Henderson T R. The ns-3 network simulator [J]. Modeling & Tools for Network Simulation, 2010, 20(5): 15-34.

[16] Font J L, Iñigo P, Domínguez M, et al. Analysis of source code metrics from ns-2 and ns-3 network simulators [J]. Simulation Modelling Practice & Theory, 2011, 19(5): 1330-1346.

Note: Figure translations are in progress. See original paper for figures.