SURE 静岡大学学術リポジトリ
Shizuoka University REpository

A Virtual WLAN Device Model for High-Fidelity Wireless Network Emulation

著者	Kawai Takaaki, Kaneda Shigeru, Takai Mineo, Mineno Hiroshi
journal or publication title	ACM Transactions on Modeling and Computer Simulation
volume	27
number	3
page range	17
year	2017-08
出版者	Association for Computing Machinery, Inc.
権利	(C) ACM 2017. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in ACM Transactions on Modeling and Computer Simulation, http://dx.doi.org/10.1145/3067664.
URL	http://hdl.handle.net/10297/00024382

doi: 10.1145/3067664

A Virtual WLAN Device Model for High Fidelity Wireless Network Emulation

TAKAAKI KAWAI, Graduate School of Informatics, Shizuoka University, Japan.
SHIGERU KANEDA, Space-Time Engineering, LLC.
MINEO TAKAI, Osaka University, Japan / University of California, Los Angeles.
HIROSHI MINENO, College of Informatics, Academic Institute, Shizuoka University, Japan.

The recent popularization of mobile devices has increased the amount of communication traffic. Hence, it is necessary both in academia and industry to research load distribution methods for mobile networks. An evaluation environment for large scale networks that behaves like a practical system is necessary to evaluate these methods, and either a physical environment or simulation environment can be used. However, physical and simulation environments each have their advantages and disadvantages. A physical environment is suitable for practical operation because it is possible to obtain data from a real environment. In contrast, the cost for a large number of nodes and the difficulty of field preparation are its disadvantages. Reproducing radio propagation is also a challenge. Network simulators solve the disadvantages of the physical environment by modeling the entire evaluation environment. However, they do not exactly reproduce the physical environment because the nodes are abstracted. This paper presents an evaluation environment that combines a network simulator and virtual machines with virtual wireless local area network (LAN) devices. The virtual machines reproduce the physical environment with high fidelity by running the programs of the physical machines, and the virtual wireless LAN devices make it possible to emulate wireless LAN communication using default operating system drivers. A network simulator and virtual machines also reduce the cost for nodes, ease the burden of field preparation, and reproduce radio propagation by modeling the evaluation environment. In the evaluation, the proposed method decreased the difference from the physical environment to 5% in terms of transmission control protocol throughput. In the case of user datagram protocol, the proposed method decreased the difference from the physical environment down to 1.7%. The number of virtual machines available on a host machine and the practical use of the proposed method are also discussed.

CCS Concepts: •**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Additional Key Words and Phrases: Wireless LAN simulation, Wireless LAN emulation, Virtual device, Virtual machine

ACM Reference Format:
Takaaki Kawai, Shigeru Kaneda, Mineo Takai, and Hiroshi Mineno, 2016. A Virtual WLAN Device Model for High Fidelity Wireless Network Emulation. *ACM Trans. Model. Comput. Simul.* 9, 4, Article 39 (March 2016), 22 pages.
DOI: 0000001.0000001

1. INTRODUCTION

The amount of communication traffic has increased because of the recent popularization of mobile devices. It is hence critical to research load distribution methods to

This study was supported by Grant-in-Aid for Scientific Research (B) 26280028 and 15H02697, Japan.
Author's addresses: T. Kawai, Graduate School of Informatics, Shizuoka University, Japan; S. Kaneda, Space-Time Engineering, LLC; M. Takai, Osaka University, Japan / University of California, Los Angeles; H. Mineno, College of Informatics, Academic Institute, Shizuoka University, Japan.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2016 ACM. 1049-3301/2016/03-ART39 $15.00
DOI: 0000001.0000001

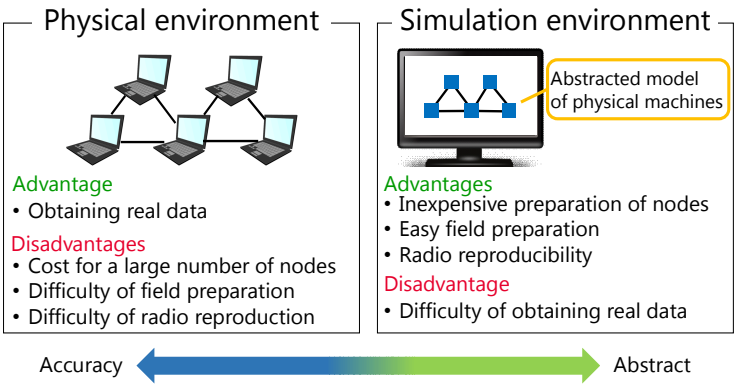

Fig. 1. Advantages and disadvantages of physical and simulation environments

handle this increased communication traffic on mobile networks. For example, a traffic distribution method that utilizes multiple channels such as Wi-Fi and WiMAX [Aijaz et al. 2013] needs to be verified to determine if its approach is practical. A physical or simulation environment can be used to evaluate such large scale networks, but each have their advantages and disadvantages, as shown in **Figure 1**. A physical environment is suitable for practical operation because real data can be obtained from the real environment. However, the cost for a large number of nodes are required and field preparation is challenging. Difficulty of reproducing radio propagation is also a disadvantage in the case of wireless communication. This reproducibility is important so that the defects of the algorithm can be discriminated from radio interference. In contrast, a simulation environment solves the disadvantages of the physical environment by modeling the whole evaluation environment. However, it does not simulate a physical environment with high fidelity because the nodes are abstracted. The definition of fidelity in this paper is the degree to which the condition and behavior of a physical environment are reproduced. Abstracted nodes cannot run real programs; therefore, they lead to differences between the physical and simulated evaluation results.

The TOSSIM [Levis et al. 2003] has addressed the disadvantages of both environments by combining a network simulator and an emulator in the field of sensor networks. However, the simulation-physical environment trade-off is still an issue in the field of wireless local area networks (LANs). The combination of a network simulator and virtual machines has been proposed to improve the fidelity of node models to physical machines [Weingärtner et al. 2011; Space-Time Engineering, LLC. 2015a]. In this context, a virtual machine is software that emulates physical hardware. Virtual machines simplify the implementation of node model fidelity by running any operating system (OS) without modification. Wireless LANs have been emulated on virtual machines using dedicated drivers that create a wireless LAN interface on a guest OS (an OS on a virtual machine) [Weingärtner et al. 2011]. However, the drivers for each specific OS must be implemented, incurring costs in terms of work and time. In addition, the difference in drivers leads to different behavior in the emulation and practical environments.

The Scenargie network simulator [Space-Time Engineering, LLC. 2015a] utilizes device virtualization to emulate communication with default OS drivers. Scenargie works with QEMU virtual machines [QEMU 2015] to use virtual wired LAN devices. However, Scenargie and QEMU do not support the virtualization of wireless LAN devices.

This paper proposes an evaluation environment that combines a network simulator and virtual machines. In addition, the authors implement virtual wireless LAN devices to emulate wireless LAN communication on the virtual machines. The contributions of this paper are the following:

— The utilization of virtual machines enables multiple nodes on a single PC and reduces the cost for nodes.
— Radio propagation in the simulation environment reduces the necessity for large-scale experimental fields.
— Radio propagation simulation successfully reproduces radio propagation.
— Virtual machines achieve fidelity to the physical environment by running native programs, i.e., binary programs runnable on physical machines. In particular, a virtual wireless LAN device for a virtual machine makes it possible to emulate wireless LAN communication using default OS drivers.

In addition to above ones, native wireless LAN drivers can be debugged in proposed environment. The native wireless LAN drivers can run on the virtual wireless LAN device, hence behaviors of the drivers in practical environment can be tested by emulation.

The rest of this paper proceeds as follows. Section 2, reviews related work, and Section 3 presents an overview of the proposal. In Section 4, the detail of prototype implementation is described. Section 5 describes the common evaluation conditions for Section 6 and 7. Sections 6 and 7 evaluate the system in terms of fidelity to the physical environment and available number of virtual machines. Finally, an example of its practical use is provided in Section 8, and the conclusions are presented in Section 9.

2. RELATED WORK

2.1. Requirements

Physical and simulation environments have complementary advantages and disadvantages, as discussed in Section 1. To address these issues in a test environment, the following requirements should be satisfied.

(1) Easy and inexpensive preparation of nodes: a large number of nodes can be prepared inexpensively for large scale experiments.
(2) Easy field preparation: field preparation for experiments is easy so that wide-area wireless LANs can be evaluated in limited fields.
(3) Radio reproducibility: radio propagation is reproducible so that the defects of the algorithm can be discriminated from radio interference.
(4) Utilization of native programs: an evaluation can be performed with native programs so that the physical environment is correctly reproduced and node model creation is easy.

The authors classified methods to achieve these requirements into two types: those that improve the physical environment and those that improve the simulation environment.

2.2. Methods improving the physical environment

The physical environment already utilizes native programs, so the main issues are requirements 1, 2, and 3. A physical machine can be used for multiple virtual nodes for easy preparation of nodes and fields [Networks and Communication Systems Branch 2015; Ahrenholz et al. 2008; Zhang and Li 2002]. For example, EMANE [Networks and Communication Systems Branch 2015] emulates multi-node communication on a physical machine with Linux containers. Linux containers create virtual hosts on an OS by

allocating individual process space and network space. Other approaches provide a large-scale communication environment to share among users [Miyachi et al. 2011; Nakata et al. 2007; National Institute of Information and Communications Technology 2015; Raychaudhuri et al. 2005]. StarBED[3] [National Institute of Information and Communications Technology 2015] installs a physical large-scale network with more than 1,000 PCs. Users can create arbitrary networks easily by changing the topologies and OSs on the shared physical network. However, correctly simulating physical radio propagation is an issue because these methods emulate communication delay and packet loss with a wired LAN frame filter.

ORBIT [Raychaudhuri et al. 2005] installs large-scale networks with physical radio and 64 nodes on a ceiling. Users can install networks easily by sharing the physical network through web interfaces. However, the reproducibility of the radio propagation is an issue because physical nodes can cause random radio interference with other nodes in the environment.

Other methods control the behavior of protocols. Dummynet [Rizzo 1997] emulates the communication conditions such as bandwidth and delay by changing the length of the queue between the transmission control protocol (TCP) and internet protocol (IP) layers. However, Dummynet does not emulate a physical wireless LAN with high fidelity because radio propagation is not emulated. Another method emulates delay, bandwidth, and packet loss rates using a packet filter based on data from physical wireless LAN communications [Noble et al. 1997]. However, it is difficult to conduct an evaluation of practical wireless LAN systems because this method requires the protocol in a physical machine to be modified.

There is a method that emulates radio propagation [Mano and Saruwatari 2014]. This method emulates radio propagation in a physical device using I/Q data, which describes sign waves used to generate radio waves by an analog circuit. However, it is difficult to reduce the hardware cost because this method requires specific hardware.

2.3. Methods improving the simulation environment

Network simulators achieve requirements 1, 2, and 3 by modeling the entire evaluation environment. Hence, the main issue is requirement 4. The well-known network simulators are OPNET [Riverbed Technology 2015], OMNeT++ [Varga and Hornig 2008], NS-2 [ns-2 2015], GloMoSim [Zeng et al. 1998], and QualNet [SCALABLE Network Technologies 2015b]. These network simulators are suitable for evaluations in ideal conditions because the models of the entire environment are able to specify detailed parameters such as radio propagation and protocol behavior. However, it is difficult to conduct an evaluation that is similar to one on physical machines because the nodes are abstracted.

There are methods to utilize native protocol stacks in simulation environments to improve the modeling of physical machines [Barr et al. 2005; Krop et al. 2007]. For example, JiST/SWANS [Barr et al. 2005] utilize native application programs in a simulation environment by synchronizing the behavior of Java virtual machines and the discrete time of a network simulator. However, all protocols are abstracted except for those of the target layer. Therefore, the system's fidelity to physical machines remains an issue.

EXata [SCALABLE Network Technologies 2015a] combines a network simulator and physical machines to utilize whole native programs in a simulation environment. The network simulator simulates radio propagation, and physical machines are associated with the network simulator as nodes. EXata improves the physical environment simulation accuracy by utilizing native OSs. Using the entire native programs of physical machines by combining a network simulator with emulated nodes has also been proposed [Staub et al. 2009; Werthmann et al. 2014; Henderson et al. 2008; Erazo et al.

Table I. Characteristics of the methods improving physical and simulation environments

Requirement	Physical environment	Simulation environment
(1) Easy and inexpensive preparation of nodes	Depends on methods	Satisfied
(2) Easy field preparation	Depends on methods	Satisfied
(3) Radio reproducibility	Depends on methods	Satisfied
(4) Utilization of native programs	Satisfied	Depends on methods

2015]. For example, VirtualMesh [Staub et al. 2009] combines OMNeT++ and virtual machines on Xen [Barham et al. 2003]. OMNeT++ simulates radio propagation and Xen runs a native OS. However, these methods do not accurately simulate physical wireless LAN networks because wireless LAN interfaces are not emulated.

There are methods that transmit wireless LAN frames in virtual machines to accurately simulate physical wireless LANs [Weingärtner et al. 2011; Xia et al. 2011]. Weingärtner et al. [2011] proposed creating wireless LAN interfaces on a guest OS by installing a dedicated driver. However, drivers for specific guest OSs must be implemented, which incurs cost in terms of work and time. In addition, the difference in the drivers leads to differences in behavior between the emulation environment and practical environment. Xia et al. [2011] proposed a system where guest OSs can handle wireless LAN communication using the default drivers of guest OSs. However, this method is not suitable for emulation because it assumes wireless LAN frames will be transmitted through physical wireless LAN devices.

The Scenargie network simulator [Space-Time Engineering, LLC. 2015a] utilizes default OS drivers by device virtualization and works with QEMU virtual machines [QEMU 2015] to utilize a virtual wired LAN device. However, Scenargie and QEMU do not support the virtualization of wireless LAN devices.

2.4. Role of the proposal

Table I summarizes the characteristics of the methods that improve physical or simulation environments. Improving the physical environment achieves requirement 4. However, these methods do not satisfy all of the other requirements. The methods that improve the simulation environment satisfy requirements 1, 2, and 3. However, the accurate simulation of physical machines is an issue because it is difficult to utilize the entire programs of physical nodes.

This study focuses on methods that improve simulation environments because this approach has fewer issues. In this paper, an evaluation environment for wireless LANs satisfying all requirements by combining a network simulator and virtual machines is proposed. A virtual wireless LAN device for a virtual machine to emulate wireless LAN communication is also implemented.

3. HIGH FIDELITY EMULATION ENVIRONMENT

The main advantage of the proposed system is that a virtual wireless LAN device emulates wireless LAN communication using native drivers. In this system, the host machine is the physical machine running the virtual machines, the host OS is the OS on the host machine, and the simulation node is the abstracted node model implemented in a network simulator.

Figure 2 presents the system concept. A network simulator simulates radio propagation, virtual machines run as node models with a native OS, and the network simulator has simulation nodes. Simulation nodes combine the virtual wireless LAN device and the radio propagation model of the network simulator. Virtual machines connect to the corresponding simulation nodes as their entities. The applications on guest OSs transmit wireless LAN frames from the virtual wireless LAN device. The network simulator receives the wireless LAN frames and forwards them to other virtual wireless

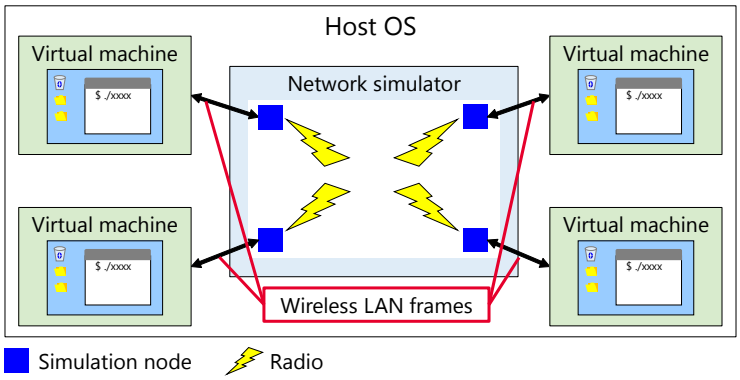

Fig. 2. System concept

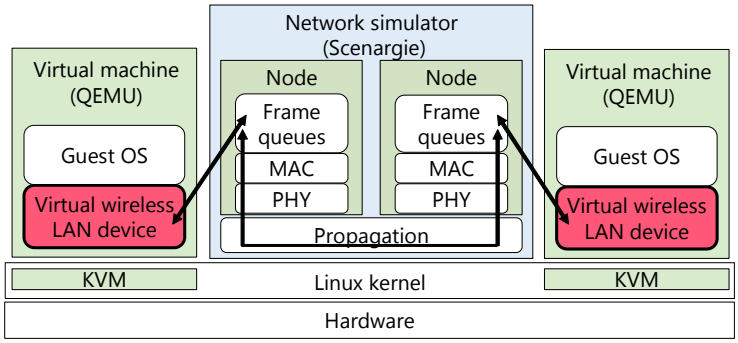

Fig. 3. Implementation of the High Fidelity Emulation Environment (HiFEE)

LAN devices based on the radio propagation simulation. Applications on the virtual machines utilize a virtual wireless LAN device through the default OS driver as if there were a physical wireless LAN device. The virtual machines and virtual wireless LAN device accurately emulate the physical environment. The virtual machines also reduce the burden of model implementation.

Figure 3 shows the prototype implementation called the High Fidelity Emulation Environment (HiFEE). Scenargie 1.8 [Space-Time Engineering, LLC. 2015a] was chosen as the network simulator and QEMU 2.1.2 [QEMU 2015] as the virtual machine. However, the system concept can be implemented on various network simulators and virtual machines. This prototype models the AR9160 (QUALCOM Atheros) as the virtual wireless LAN device. The guest OS is a native program that creates the protocol stack from the application layer to the media access control (MAC) management of the MAC layer. MAC management consists of the functions for handling wireless LAN association and beacon frame creation. Scenargie models the processing from the MAC control to radio propagation. Rate control, modulation, and carrier sense multiple access/collision avoidance (CSMA/CA) are processed by the MAC control.

The following functions are modeled in HiFEE.

— IEEE 802.11g communication.
— Virtual wireless LAN devices can run in infrastructure, ad-hoc, master, and monitor modes.

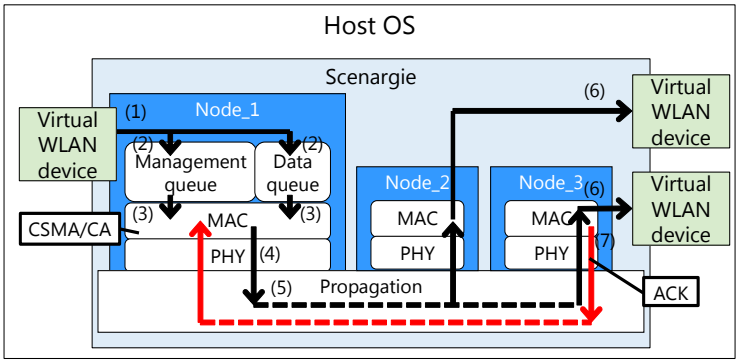

Fig. 4. Protocol stack in Scenargie

— Scenargie send the simulation results of the received signal strength indicator (RSSI) and noise floor to the virtual wireless LAN device.
— QEMU runs based on actual time. Simulation times in Scenargie are synchronized to actual time with 100 ms accuracy.

The infrastructure mode is used for connection to access points (APs). The master mode runs the virtual wireless LAN devices as an AP. Ad-hoc mode enables the wireless LAN devices to connect with each other to create ad-hoc networks. The monitor mode is for monitoring wireless LAN frames in the air.

4. DETAIL OF PROTOTYPE IMPLEMENTATION

4.1. Modeling in Scenargie

Scenargie models the processing from the MAC control to radio propagation using the protocol stack illustrated in **Figure 4** as follows.

(1) The virtual wireless LAN device transmits wireless LAN frames to a simulation node.
(2) Scenargie enqueues the received frames. The management queue is for MAC management frames, and the data queue is for MAC data frames. Scenargie automatically changes the frame transmission timing, which consists of Short Inter-Frame Space (SIFS) and Distributed Inter-Frame Space (DIFS), based on the queue types.
(3) The MAC layer dequeues the wireless LAN frames from the management and data queues. The MAC layer simulates CSMA/CA.
(4) The physical (PHY) layer forwards the wireless LAN frames from the MAC layer to the propagation layer.
(5) The propagation layer forwards the wireless LAN frames to other simulation nodes when these nodes can receive radio in the simulated field.
(6) Other nodes receive the wireless LAN frames through the propagation, PHY, and MAC layers.
(7) The destination node of the frames transmits acknowledgement (ACK) frames from the MAC layer to the transmitter.

4.2. Virtual wireless LAN device

The AR9160 virtual wireless LAN device satisfies the behaviors required by the ath9k driver, which is included in backports-3.18.1-1 [Linux kernel backports 2015] as a prototype. Backports are the patches used to install the latest version of wireless LAN drivers on old versions of Linux.

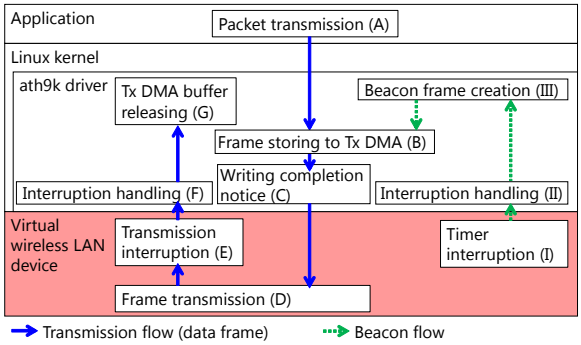

Fig. 5. Frame transmission

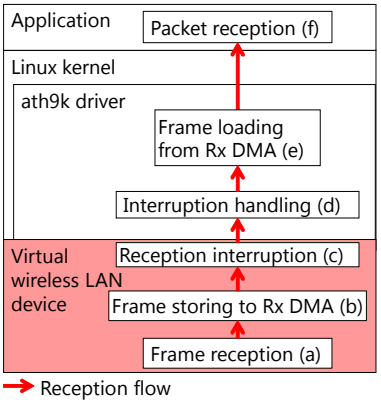

Fig. 6. Frame reception

The main processes of the virtual wireless LAN device are frame transmission and frame reception. The frame transmission consists of data frame transmission and beacon frame transmission. **Figure 5** shows the frame transmission procedures (A)–(G), which are performed as follows.

(A) An application in a guest OS transmits packets, and MAC headers are added to the packets in the MAC layer.
(B) An ath9k driver receives the MAC frames. The ath9k driver stores the received frames in the transmission Direct Memory Access (DMA), which is a memory space shared by the ath9k driver and virtual wireless LAN device.
(C) The ath9k driver writes bits to the registers of the virtual wireless LAN device to give notice of transmission preparation completion.
(D) The virtual wireless LAN device loads the frames from the transmission DMA and forwards them to Scenargie.
(E) The virtual wireless LAN device issues a transmission interruption.
(F) The ath9k driver receives the transmission interruption.
(G) The ath9k driver releases the frame in the transmission DMA to prepare for the next transmission.

The ath9k driver transmits beacon frames when the mode of a wireless LAN interface is in master mode or ad-hoc mode. When a wireless LAN interface changes its mode to master mode or ad-hoc mode, the ath9k driver sends the timer interruption interval for beacon transmission to the virtual wireless LAN device. The timer interruption periodically calls the beacon frame transmission procedures (I)–(III) in Figure 5, which consist of the following steps.

(I) The virtual wireless LAN device periodically issues timer interruptions based on the beacon transmission interval written in a register.
(II) The ath9k driver receives the timer interruptions.
(III) The ath9k driver generates beacon frames according to the timer interruption processing. The beacon frames are transmitted using steps (B)–(G) of Figure 5.

Figure 6 shows the frame reception procedure. The virtual wireless LAN device receives frames according to steps (a)–(f) in Figure 6, detailed as follows.

(a) Scenargie forwards the wireless LAN frames to QEMU. The virtual wireless LAN device receives the frames through the QEMU reception handler.

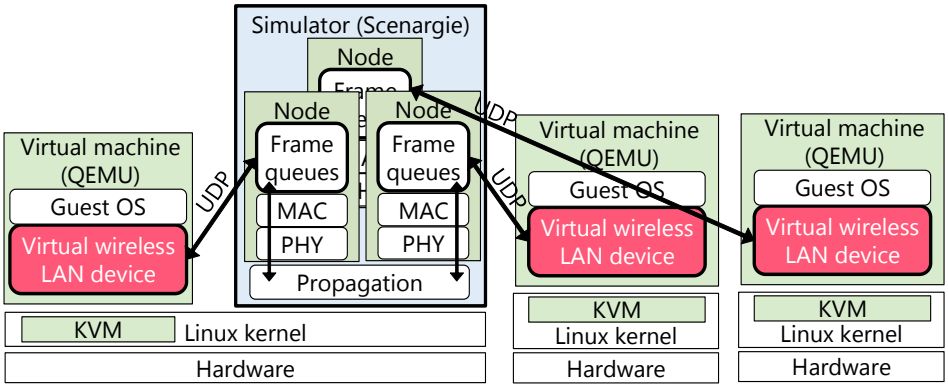

Fig. 7. Load distribution by multiple host machines

Table II. Characteristics of TCP and UDP

Protocol	Advantages	Disadvantages
TCP	• Communication reliability	• Low connection flexibility • Throughput could decrease because of the duplication of flow control on host and guest OSs • Overhead caused by frame header size
UDP	• Connection flexibility • No duplication of flow control • Smaller frame header size	• Low communication reliability • Reception of malicious packets

(b) The virtual wireless LAN device stores the frames to reception DMA, which is a memory space shared by the ath9k driver and virtual wireless LAN device.
(c) The virtual wireless LAN device issues a reception interruption.
(d) The ath9k driver receives the reception interruption.
(e) The ath9k driver loads the frames from the reception DMA using reception interruption processing.
(f) The ath9k driver passes the frames to the upper layer of the protocols.

4.3. Connection between Scenargie and QEMU

HiFEE supports load distribution by running QEMU virtual machines on multiple host machines, as shown in **Figure 7**. Scenargie and QEMU virtual machines connect with each other using user datagram protocol (UDP) to support the utilization of virtual machines on remote hosts. It is difficult to run a large number of virtual machines on a single host machine, therefore, HiFEE supports the use of remote hosts. Other protocols such as TCP are also available for the connection between Scenargie and QEMU. The characteristics of TCP and UDP are compared in **Table II**. HiFEE uses UDP because of its connection flexibility and transmission rate. The connection flexibility means that Scenargie and QEMU automatically connect with each other, even after Scenargie or QEMU have been restarted. Users of HiFEE will restart Scenargie and QEMU repeatedly to coordinate simulation scenarios or settings for guest OSs in practical evaluations. If Scenargie and QEMU connect with each other by TCP, the connection cannot be restored automatically because TCP connection requires a running server. In contrast, in the case of UDP, Scenargie and QEMU can restore the connection automatically after restarting Scenargie or QEMU. UDP is also suitable regarding transmission rate because it does not cause the duplication of flow controls on host and guest OSs. TCP causes the duplication of flow controls similar to the so-called

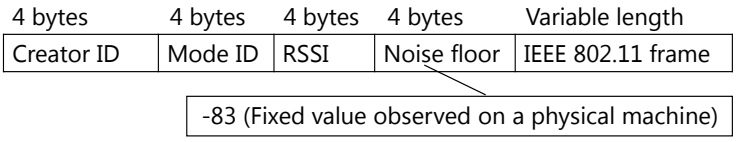

Fig. 8. Frame format for the communication between Scenargie and QEMU

Table III. Spec of host machine

CPU	Intel CORE i7 3.6 GHz
RAM	16 GB
Storage (HDD)	1 TB
OS	CentOS 6.4

Table IV. Spec of physical node

Product name	Scenargie Comm Node
CPU	AMD Geode LX800 500 MHz
RAM	256 MB
Storage (Compact flash)	16 GB
Wireless LAN device	Atheros AR9160B, AR9106A (external radio device)
Wireless LAN driver	backports-3.18.1-1
OS	CentOS 6.4

TCP over TCP problem. Wireless LAN frames from guest OSs are treated as payloads in a host OS. These payloads are controlled by multiple flow controls in the host and guest OSs in the case of TCP. The duplication of flow control decreases the throughput on guest OSs. Another advantage of UDP is that the header size is smaller than that of TCP. Smaller headers reduce the communication overhead outside of guest OSs.

Scenargie can send various simulation results to the virtual wireless LAN devices, such as the RSSI and noise floor. **Figure 8** shows the frame format for the communication between Scenargie and QEMU, which is added by Scenargie and QEMU to the head of wireless LAN frames. Scenargie does not propagate additional headers for precise evaluation in simulation processing. The creator ID and Mode ID are reserved for future function expansion. Scenargie writes the RSSI simulation result and a fixed value for the noise floor in the header. The noise floor is the value observed on a physical wireless LAN device immediately after the boot of a physical machine. Scenargie also can change the noise floor value dynamically when noise floor simulation is required. Frame processing in Scenargie and QEMU will be extended to send more simulation results to virtual wireless LAN devices in the future.

5. COMMON EVALUATION CONDITIONS

This study evaluated the HiFEE in terms of the fidelity to a physical environment and the available number of QEMU virtual machines. Here the authors describe the common evaluation conditions used in this research. Scenargie and HiFEE ran on the host machine listed in **Table III**. The firewall was disabled on the host machine to avoid the influence on the bandwidth. QEMU virtual machines modeled physical nodes specified by the values in **Table IV**. The physical nodes have an external AR9106A radio device. However, the virtual wireless LAN device does not emulate the AR9106A because the ath9k driver does not handle AR9106A and the behavior of the AR9106A could not be determined from the driver. Instead, Scenargie simulates the MAC control of the wireless LAN device.

The Dot Eleven Module radio propagation model is used [Space-Time Engineering, LLC. 2015b]. This model is available in Scenargie. Scenargie simulates radio propagation using the IEEE 802.11g model, which is included in its Dot Eleven Module.

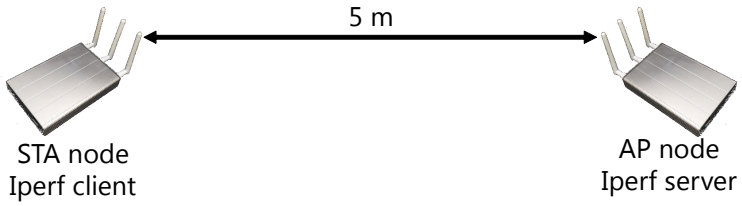

Fig. 9. Evaluation topology in the physical environment

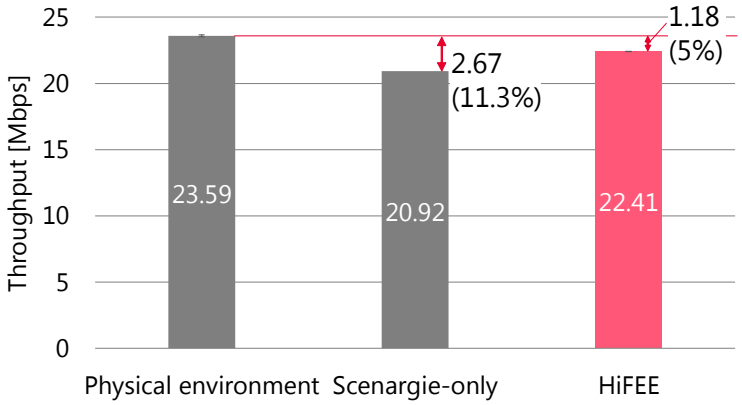

Fig. 10. TCP throughput

In the MAC layer, parameter "64QAM_0.75" and "static" are used for modulation and rate control, respectively. Other parameters are the same as the default values of the Scenargie Dot Eleven Module.

TCP and UDP throughput between guest OSs was measured using iperf [Dugan et al. 2015] for the performance evaluation of the HiFEE. One guest OS runs iperf as an iperf server, which is a packet receiver, and the other guest OSs run iperf as an iperf client. The iperf client is a packet transmitter and measures the throughput to an iperf server. Finally, the iperf client bandwidth option was set to 30 GB for UDP. All reported throughput values are the average of ten trials.

6. ACCURACY OF HIFEE

Both TCP throughput and UDP throughput were used as indexes to evaluate the fidelity to the physical environment. TCP and UDP throughputs were measured between two nodes in the physical environment, Scenargie-only simulation, and HiFEE. The Scenargie-only is the case that Scenargie simulates the whole environment without the QEMU virtual machines. The results of the Scenargie-only simulation and HiFEE were compared to the physical environment results to evaluate the improvement in accuracy. **Figure 9** shows the topology used for the throughput measurement. A station (STA) node and an AP node 5 m apart connect with each other. The physical environment evaluation was conducted in a field of grass with low interference. The Scenargie-only and HiFEE evaluations reproduced the physical environment.

Figure 10 shows the TCP throughput. The throughputs for the physical environment, Scenargie-only, and HiFEE tests were 23.59, 20.92, and 22.41 Mbps, respectively. The throughput difference between the physical environment and Scenargie-only tests was 2.67 Mbps, which is 11.3% of the throughput of physical environment. The throughput difference between the physical environment and HiFEE tests was

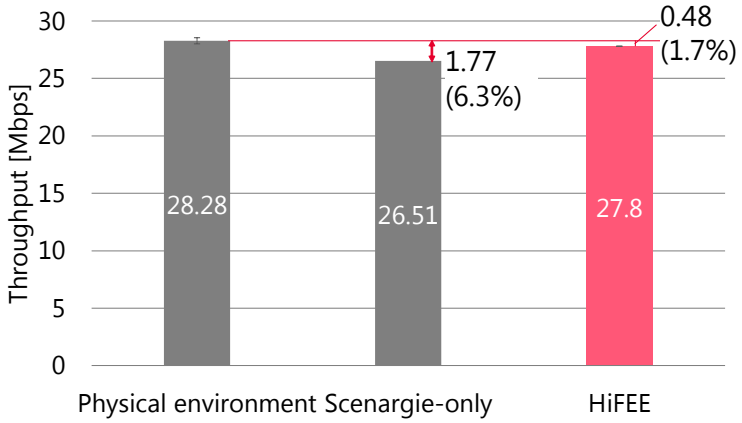

Fig. 11. UDP throughput

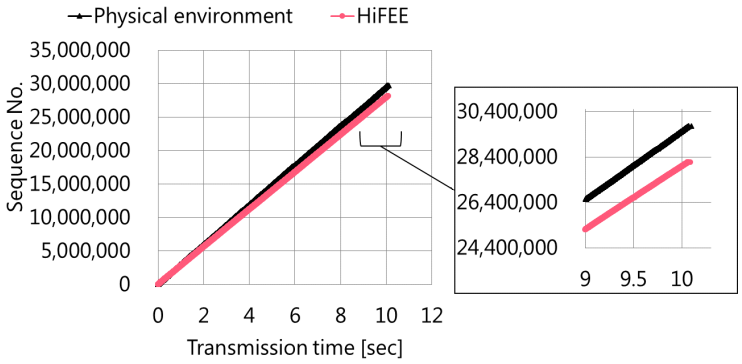

Fig. 12. Change in sequence number of TCP communication

1.18 Mbps, which is 5% of the throughput of physical environment. HiFEE decreased the throughput difference from the physical environment. Therefore, it can be concluded that HiFEE improves the fidelity to physical environment.

HiFEE also improves fidelity to the physical environment in the case of UDP, as shown in **Figure 11**. The throughputs of the physical environment, Scenargie-only, and HiFEE tests were 28.28, 26.51, and 27.8 Mbps, respectively. The throughput difference between the physical environment and Scenargie-only tests was 1.77 Mbps, which is 6.3% of the throughput of physical environment. The throughput difference between the physical environment and HiFEE tests was 0.48 Mbps, which is 1.7% of the throughput of physical environment. Therefore, HiFEE also improves the fidelity to the physical environment when UDP is used.

The change in sequence numbers of the physical environment and HiFEE was evaluated in detail. Two QEMU virtual machines used iperf to communicate each other by TCP through Scenargie, just as in the evaluation of TCP throughput. Tshark packet capture software [Wireshark 2015] then recorded the sequence numbers of the packets transmitted by the iperf client.

Figure 12 shows the result of the change in sequence number, which was similar for both HiFEE and the physical environment. The gap in the sequence numbers increased gradually as time proceeded. It was assumed that the cause of the gap was in the virtual wireless LAN device or Scenargie; however, the difference of rate con-

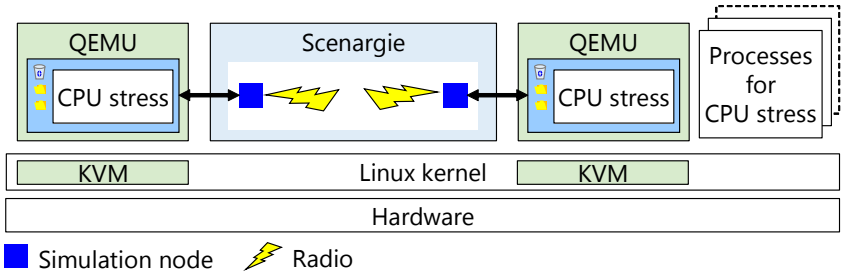

Fig. 13. Topology for TCP communication with CPU stress

trol was found to be the cause of the gap. The TCP throughput between two QEMU virtual machines connected directly without Scenargie was evaluated to analyze the behavior of the virtual wireless LAN device. The result showed that virtual wireless LAN device achieved 24.72 Mbps, which is enough to emulate ideal wireless LAN communication. To analyze the behavior of Scenargie, the TCP throughput between two QEMU virtual machines without virtual wireless LAN devices was also evaluated. The Scenargie Emulation Module can emulate wireless LAN communication using a virtual wired LAN device instead of virtual wireless LAN devices. It emulated a topology that was the same as the one in Figure 9 and achieved 22.4 Mbps, which is similar to the result in HiFEE. Therefore, the models in Scenargie are believed to cause the throughput gap between the physical environment and HiFEE. The physical AR9160 transmits frames using a device specific rate control while Scenargie uses a static rate control modulated by "64QAM_0.75." However, AR9160 supports an open source rate control called minstrel_ht. For more rigorous emulation, users can use minstrel_ht and implement the minstrel_ht model in Scenargie.

7. AVAILABLE NUMBER OF QEMU VIRTUAL MACHINES

7.1. Factors limiting the available number of QEMU virtual machines

The available number of QEMU virtual machines are limited on a host machine by the following factors.

— Stress on a CPU
— Memory consumption
— Bandwidth between Scenargie and a QEMU virtual machine
— Traffic concentration on Scenargie

Sections 7.2 to 7.5 describe the influence of each factor in detail.

7.2. Stress on a CPU

The influence of CPU stress by QEMU virtual machines on emulated communication was evaluated. **Figure 13** shows the topology for the evaluation. Two QEMU virtual machines communicated with each other by TCP through Scenargie to measure the throughput. The decrease in throughput between the QEMU virtual machines was measured when the CPU stress on a host machine increased. The CPU stress on the host machine should be made by QEMU virtual machines; however, the stress command was used to avoid any influence from traffic concentration on Scenargie. The stress command is a tool to impose a load on CPU or memory. When a QEMU virtual machine does not affect the other links in HiFEE, the QEMU virtual machine can be considered like a software process on a host machine such as the stress command. One stress command process consumes one logical core of the CPU on the host machine.

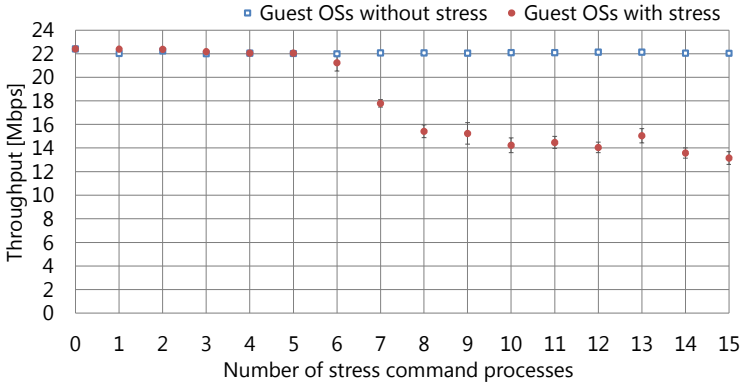

Fig. 14. Throughput of TCP communication with CPU stress

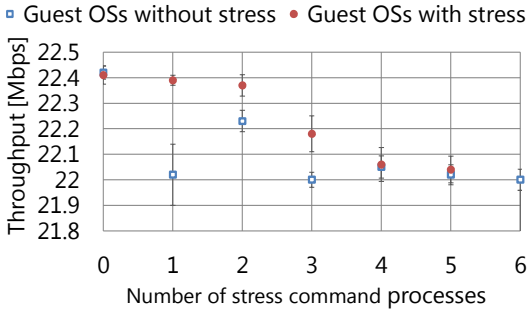

Fig. 15. Throughput of TCP communication with CPU stress (enlarged)

The host machine has eight logical cores created by multi-threading, which is a technology that treats a physical core as multiple cores. The evaluation considered the case when each guest OS had one stress command process and the case when the guest OSs had no CPU stress.

Figure 14 shows the throughput when the CPU stress increased on the host machine. When the guest OSs had no CPU stress, the throughput did not decrease substantially. However, when the guest OSs had CPU stress and there were six or more stress command processes on the host OS, the throughput decreased. It seems that the utilization of all the CPU capacity caused the throughput to decrease. All logical cores were utilized up to 100% by the five stress command processes, one Scenargie process, and two QEMU virtual machine processes. When the number of stress command processes was six or more, the throughput decreased because of insufficient processing ability. In contrast, when the number of stress command processes was five or less, the throughput did not decrease significantly. Therefore, users can add QEMU virtual machines until the processes on a host machine utilize all logical cores up to 100%.

The case when there were up to five stress command processes was analyzed in detail. **Figure 15** shows an enlarged portion of Figure 14. Even if the number of processes was five or less, the throughput decreased gradually when the guest OSs were stressed. This was caused by processes sharing the physical cores of the CPU. The host OS assigns processes to multiple logical cores. When the QEMU virtual machines, Scenargie, and other processes share logical cores created from the same physical core, the processing ability will be insufficient.

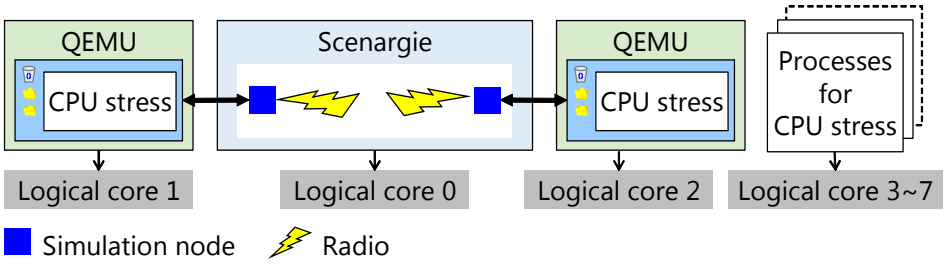

Fig. 16. Environment to assign processes to different logical cores

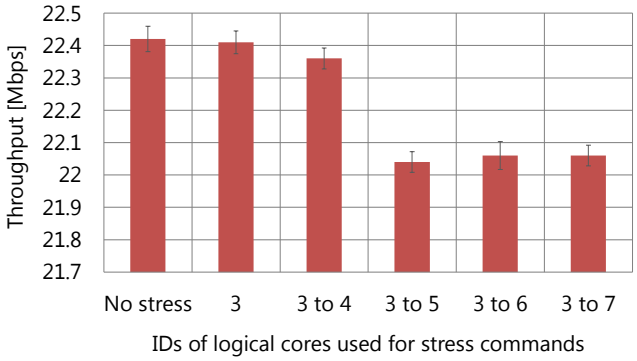

Fig. 17. Throughput when processes are fixed to different logical cores

To confirm the effects of processes sharing the same physical CPU core, the processes of Scenargie, the QEMU virtual machines, stress commands, and other processes were fixed to different logical cores. **Figure 16** shows the evaluation environment. Here, HiFEE emulated TCP communication between two nodes, and the QEMU virtual machines were given a stress command. In addition, the host machine had eight logical cores from logical core 0 to 7 created by four physical cores. The pair of logical core sharing same physical cores are logical core 0 and 4, 1 and 5, 2 and 6, and 3 and 7. Scenargie ran on logical core 0, and the QEMU virtual machines ran on logical cores 1 and 2. The number of stress command processes were increased one-by-one and ran on logical cores 3 to 7. Background services of the host OS were assigned to logical core 7 as much as possible.

Figure 17 shows the result of the throughput when each process was fixed to run on different logical cores. When the stress command process ran only on logical core 3, the throughput did not decrease because logical core 3 did not share the physical core that ran the QEMU virtual machines or Scenargie. However, when the stress command processes utilized other cores, the throughput decreased because logical cores 4 to 6 shared the physical cores that ran Scenargie or the QEMU virtual machines.

To confirm the influence from other logical cores, another experiment was conducted such that only one stress command process ran on one logical core in the same topology, as shown in Figure 16. **Figure 18** shows the throughput results when there is a stress command process running on a logical core. The throughput decreased when the stress command process ran on logical core 5. Logical cores 5 and 1 shared the physical core running the iperf server QEMU virtual machine. It seems that an iperf server was especially susceptible to another logical core sharing the same physical core. To confirm this influence, the iperf server and iperf client places were swapped, and **Figure 19**

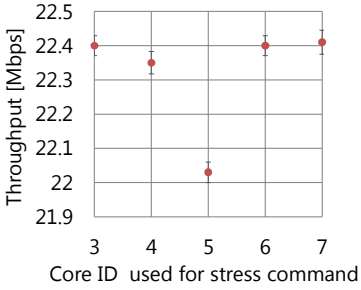

Fig. 18. Throughput when there is a stress process on a logical core

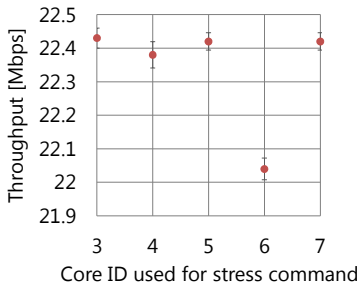

Fig. 19. Throughput after swapping the iperf server and iperf client places

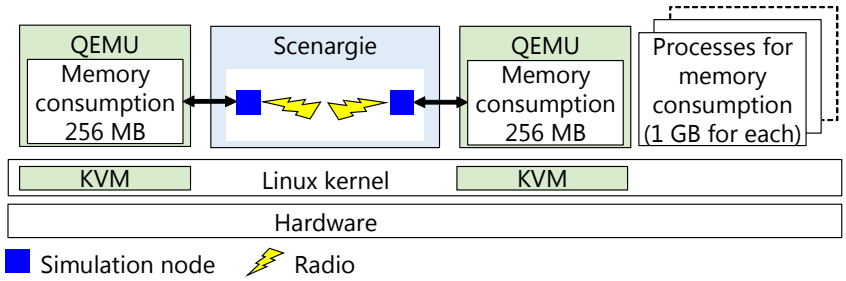

Fig. 20. Topology for TCP communication with memory consumption

shows the throughput results, where logical core 6 affected the throughput instead of logical core 5. Logical cores 6 and 2 shared the physical core running the iperf server QEMU virtual machine. Thus, it can be concluded that one logical core affects the other logical core that shares the same physical core. Hence, users of HiFEE should use different physical cores to run Scenargie and QEMU virtual machines for rigorous evaluation.

7.3. Memory consumption

The influence of memory consumed by the QEMU virtual machines on emulated communication was measured. **Figure 20** shows the topology for the evaluation, where two nodes made by the QEMU virtual machines communicate with each other by TCP to measure throughput. In addition, stress command processes consume memory on the host machine. QEMU virtual machines should be the main consumers of memory, however, a stress command was used to avoid the influence of traffic concentration on Scenargie. When a QEMU virtual machine does not affect the other links in HiFEE, the QEMU virtual machine can be considered a consumer of memory on the host machine, similar to a stress command. Each stress command process consumed 1 GB memory, and the throughput decrease shows the influence of this memory consumption. Both the case when the guest OSs ran the stress command for memory consumption and when they did not were evaluated. The memory consumption in the guest OSs was 256 MB, which is the memory capacity of the QEMU virtual machines.

Figure 21 shows the throughput when the memory consumption increased. When the guest OSs had memory stress, the throughput decreased when the memory consumed by the stress command processes reached 14 GB. When the guest OSs did not have memory stress, the throughput also decreased when the memory consumed by the stress command processes reached 15 GB. The cause of throughput decrease was insufficient memory on the host machine, which has 16 GB of memory capacity. When

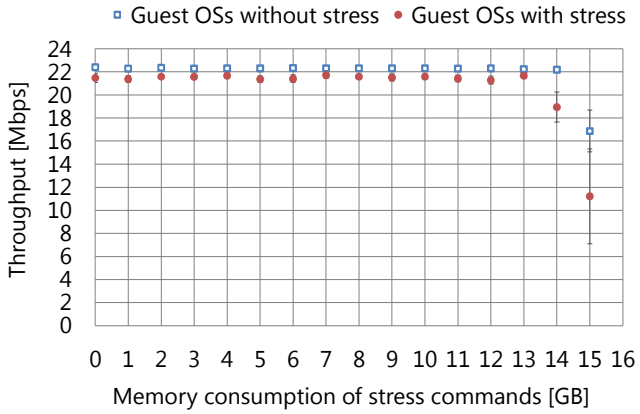

Fig. 21. Throughput of TCP communication with memory consumption

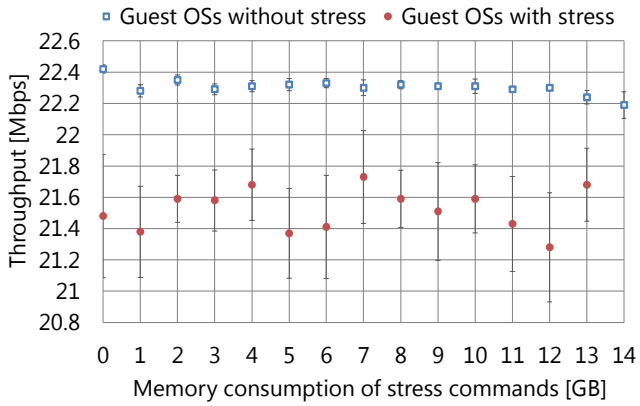

Fig. 22. Throughput of TCP communication with memory consumption (enlarged)

the stress command processes consumed 14 GB, the memory capacity was not large enough to run Scenargie, the memory-stressed QEMU virtual machines, and the background services of the host machine. The memory insufficiency also happened at 15 GB when the QEMU virtual machines did not run the memory consumption process. It can be concluded that memory consumption does not influence the throughput on guest OSs unless it exceeds the memory capacity of the host machine. **Figure 22** shows the enlarged results when the memory consumption on the host OS was 14 GB or less. The throughput decrease was not proportionate to the amount of memory consumption. Therefore, users can add QEMU virtual machines until the processes utilize up to 100% of the memory on a host machine.

7.4. Bandwidth between Scenargie and a QEMU virtual machine

To evaluate the influence of bandwidth on emulated communication, the bandwidth between Scenargie and QEMU virtual machine was controlled. **Figure 23** shows the topology used for bandwidth limitation. Two nodes created by the QEMU virtual machines communicated with each other through Scenargie by TCP to measure the throughput. The QEMU virtual machines were run on different host machines to control the bandwidth, and the tc command limited the outbound bandwidth of each host machine.

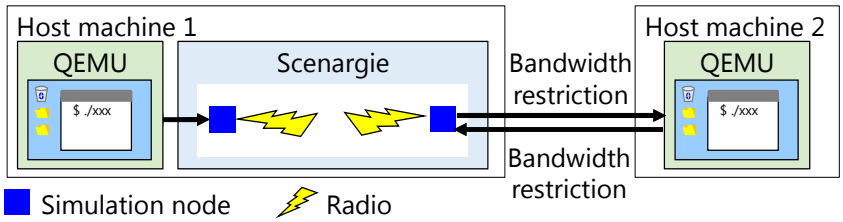

Fig. 23. Topology for bandwidth limitation

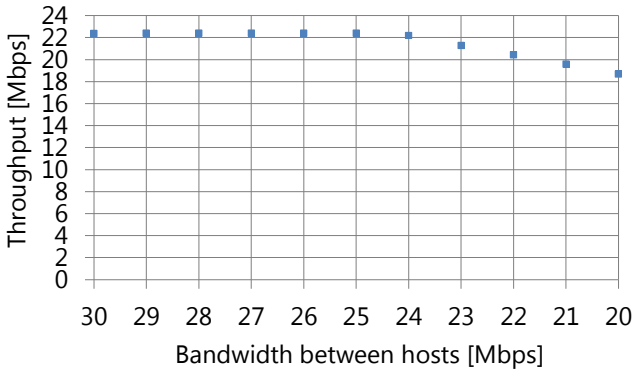

Fig. 24. Throughput of TCP communication for limited bandwidth

Figure 24 shows the change in throughput caused by the limited bandwidths. When the bandwidth between the Scenargie and QEMU virtual machine was less than 24 Mbps, the throughput decreased. The ideal throughput of wireless LAN communication is about 23 Mbps. However, the QEMU virtual machines add a UDP header and a HiFEE dedicated header to the wireless LAN frames. Therefore, the Scenargie and QEMU virtual machine require a bandwidth of over 24 Mbps per connection to emulate wireless LAN communication with the attached headers.

7.5. Traffic concentration on Scenargie

TCP throughput was measured when there were multiple links in the simulation scenario to evaluate the influence of traffic concentration. **Figure 25** shows the topology used for the evaluation. Scenargie has three links that do not cause radio interference with each other. If there is no communication overhead caused by traffic concentration, all links would achieve maximal throughput. The pair of nodes communicated with each other by TCP to measure throughput. One node ran as an AP node and the other node ran as a STA node. The QEMU virtual machines were on different physical cores to avoid any influence from other processes. Two host machines were used to prepare enough physical cores.

Figure 26 shows the TCP throughput measured in the STA nodes of each link. The result shows that there is no significant throughput decrease when the number of links increased. The worst throughput decrease was the case of link 1. The throughput of STA 1 was 22.38 Mbps for one link. When the number of links was three, the throughput of STA 1 was 22.13 Mbps. The change in the throughput was 0.25 Mbps, which is 1.1% of the throughput for only one link. Therefore, it can be concluded that the influence from traffic concentration on Scenargie is sufficiently small.

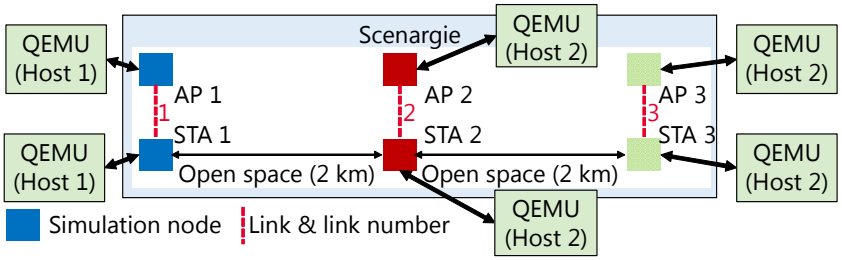

Fig. 25. Topology for multi-link communication

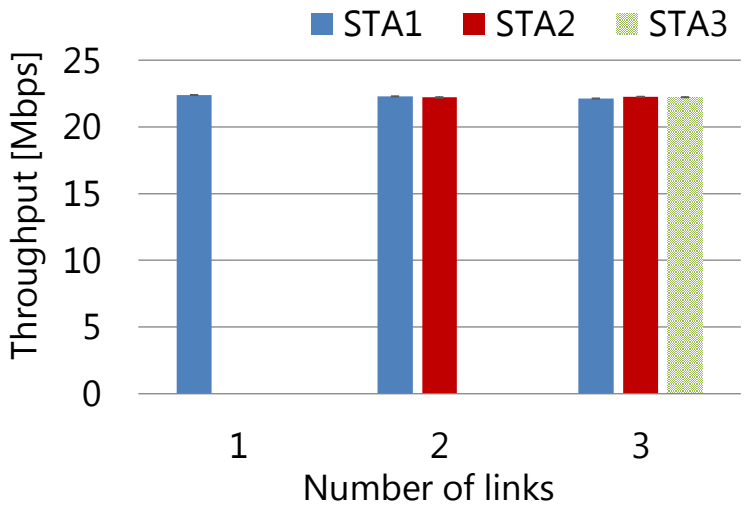

Fig. 26. Throughput of TCP communication when there are multiple links

8. EXAMPLE OF PRACTICAL USE

To demonstrate practical use of HiFEE, the performance of the open80211s mesh network [open80211s 2015] was evaluated. Open80211s is an implementation of IEEE 802.11s for Linux. Another evaluation of open80211s on a testbed was reported by Hiertz et al. [Hiertz et al. 2010]. This experiment was reproduced and the multi-hop throughput was evaluated. Hiertz et al. reported that the testbed had 12 nodes that were all in radio range of each other. In addition, a manual address filter created the topology. In this study, the topology shown in **Figure 27** was created to reproduce the evaluation environment as faithfully as possible. HiFEE emulates only seven nodes because the authors had only two host machines that could provide seven physical cores for the QEMU virtual machines in total. The nodes in HiFEE utilize IEEE 802.11g, while Hiertz et al. did not report the type of wireless LAN. Scenargie used "64QAM_0.75" modulation and "static" rate control as the parameter of the MAC layer. The specifications of the host machines and QEMU virtual machines are the same as those listed in Tables III and IV, respectively. The firewall was disabled on the host machine to avoid the influence on the bandwidth. The average TCP throughput of ten trials was measured by iperf.

Figure 28 shows the graph of the throughput traced from Hiertz et al., and **Figure 29** shows the throughput measured in HiFEE. The throughput decrease in Figure 29 shows similar tendencies to that of Figure 28. This result indicates that HiFEE

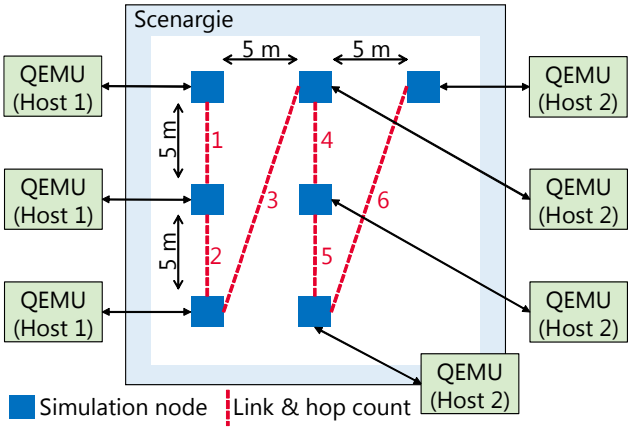

Fig. 27. Topology for evaluation of open80211s

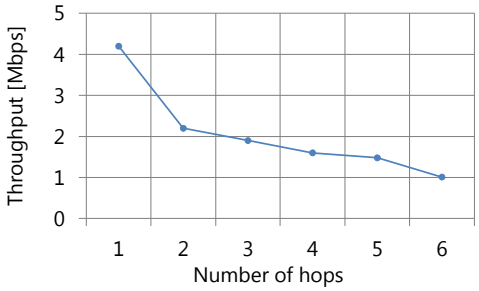

Fig. 28. Throughput traced from Hiertz et al. [2010]

Fig. 29. Throughput in HiFEE

can reproduce the evaluation environment of Hiertz et al. Therefore, HiFEE is usable for reproducing the evaluation environments of other papers. HiFEE is also able to evaluate native protocols.

9. CONCLUSION

In this paper, a wireless LAN evaluation environment that combines a network simulator and virtual machines was proposed. A virtual wireless LAN device was also implemented. The virtual machines accurately emulated the physical environment by running the programs of physical machines. In particular, a virtual wireless LAN device for a virtual machine has made it possible to emulate wireless LAN communication using the default OS drivers. A simulator and virtual machines also have addressed the cost of implementing nodes, ease of field preparation, and reproducibility of radio propagation by modeling the evaluation environment. The evaluation of the throughput shows that the proposed method modeled the physical environment more accurately than a simulator. The limitations of the available number of virtual machines and the practical use of HiFEE were also presented.

It is important to recognize the limitations of the current implementation of the prototype. First, the time in Scenargie and the clock of QEMU are not directly synchronized with each other. QEMU runs on actual time, and the time in Scenargie synchronizes to the actual time with 100 ms accuracy. Therefore, the difference in transmission and reception timing could cause communication delay between Sce-

nargie and the QEMU. In particular, when the specifications of the host machines are not sufficient to emulate a wireless LAN, the throughput among the virtual machines would be decreased by context switches. To achieve time synchronization, Weingärtner et al. [2011], Werthmann et al. [2014], and Erazo et al. [2015] have proposed approaches that synchronize the clocks of emulated nodes to simulated time. Second, the MAC control and lower layers are abstracted by the simulation model. For example, CSMA/CA is abstracted by Scenargie. The AR9160 virtual wireless LAN device was created based on the behavior of open source drivers included in backports-3.18.1-1. However, the drivers do not handle CSMA/CA, and it was difficult to obtain enough information to emulate CSMA/CA. Third, the functions of AR9160 have not been completely implemented. For example, the virtual wireless LAN device does not support IEEE 802.11n/ac and encrypted communication.

As a future work, the authors plan to implement time synchronization between Scenargie and QEMU. Time synchronization will overcome the limitation of the available number of virtual machines caused by the sharing of physical CPU cores. After time synchronization is achieved, the authors will implement additional functions of virtual wireless LAN devices.

REFERENCES

Jeff Ahrenholz, Claudiu Danilov, Thomas R. Henderson, and Jae H. Kim. 2008. CORE: A real-time network emulator. In *Proc. MILCOM '08*. IEEE, 1–7.

Adnan Aijaz, Hamid Aghvami, and Mojdeh Amani. 2013. A survey on mobile data offloading: technical and business perspectives. *IEEE Wireless Communications* 20, 2 (2013), 104–112.

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the Art of Virtualization. *SIGOPS Operating Systems Review* 37, 5 (2003), 164–177.

Rimon Barr, Zygmunt J Haas, and Robbert van Renesse. 2005. JiST: An efficient approach to simulation using virtual machines. *Software: Practice and Experience* 35, 6 (2005), 539–576.

Jon Dugan, John Estabrook, Jim Ferbuson, Andrew Gallatin, Mark Gates, Kevin Gibbs, Stephen Hemminger, Nathan Jones, Feng Qin, Gerrit Renker, Ajay Tirumala, and Alex Warshavsky. 2015. Iperf. Retrieved December 1, 2015, from https://iperf.fr/iperf-download.php.

Miguel A. Erazo, Rong Rong, and Jason Liu. 2015. Symbiotic Network Simulation and Emulation. *ACM Transactions on Modeling and Computer Simulation* 26, 1 (2015), 2:1–2:25.

Thomas R. Henderson, Mathieu Lacage, and George F. Riley. 2008. Network simulations with the ns-3 simulator. In *Proc. SIGCOMM '08*. ACM, 527.

Guido R. Hiertz, Dee Denteneer, Sebastian Max, Rakesh Taori, Javier Cardona, Lars Berlemann, and Bernhard Walke. 2010. IEEE 802.11s: The WLAN Mesh Standard. *IEEE Wireless Communications* 17, 1 (2010), 104–111.

Tronje Krop, Michael Bredel, Matthias Hollick, and Ralf Steinmetz. 2007. JiST/MobNet: Combined Simulation, Emulation, and Real-world Testbed for Ad Hoc Networks. In *Proc. WinTECH '07*. ACM, 27–34.

Philip Levis, Nelson Lee, Matt Welsh, and David Culler. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. SenSys '03*. ACM, 126–137.

Linux kernel backports. 2015. Linux kernel backports. Retrieved December 1, 2015, from http://drvbp1.linux-foundation.org/~mcgrof/rel-html/backports/.

Hiroshi Mano and Shunsuke Saruwatari. 2014. Implementation and Evaluation of a Wireless System Emulator. *IPSJ Journal* 55, 5 (2014), 1541–1554.

Toshiyuki Miyachi, Takeshi Nakagawa, Kenichi Chinen, Shinsuke Miwa, and Yoichi Shinoda. 2011. StarBED and SpringOS Architectures and Their Performance. In *Proc. TRIDENTCOM '11*. ICST, 43–58.

Junya Nakata, Satoshi Uda, Razvan Beuran, Kenji Masui, Toshiyuki Miyachi, Yasuo Tan, Kenichi Chinen, and Yoichi Shinoda. 2007. StarBED2: Testbed for Networked Sensing Systems. In *Proc. INSS '07*. IEEE, 142–145.

National Institute of Information and Communications Technology. 2015. StarBED^3. Retrieved December 1, 2015, from http://starbed.nict.go.jp/aboutus/index.html.

Networks and Communication Systems Branch. 2015. EMANE User Manual 0.8.1. Retrieved December 1, 2015, from http://downloads.pf.itd.nrl.navy.mil/docs/emane/emane.pdf.

Brian D. Noble, Mahadev Satyanarayanan, Giao T. Nguyen, and Randy H. Katz. 1997. Trace-based Mobile Network Emulation. In *Proc. SIGCOMM '97*. ACM, 51–61.

ns-2. 2015. The Network Simulator - ns-2. Retrieved December 1, 2015, from http://www.isi.edu/nsnam/ns/.

open80211s. 2015. open80211s. Retrieved December 1, 2015, from http://www.o11s.org/.

QEMU. 2015. QEMU. Retrieved December 1, 2015, from http://wiki.qemu.org/Main_Page.

Dipankar Raychaudhuri, Ivan Seskar, Max Ott, Sachin Ganu, Kishore Ramachandran, Haris Kremo, Robert Siracusa, Hang Liu, and Manpreet Singh. 2005. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *Proc. WCNC '05*, Vol. 3. IEEE, 1664–1669.

Riverbed Technology. 2015. OPNET. Retrieved December 1, 2015, from http://www.riverbed.com/products/performance-management-control/opnet.html.

Luigi Rizzo. 1997. Dummynet: A Simple Approach to the Evaluation of Network Protocols. *SIGCOMM Computer Communication Review* 27, 1 (1997), 31–41.

SCALABLE Network Technologies. 2015a. EXata. Retrieved December 1, 2015, from http://web.scalable-networks.com/content/exata/.

SCALABLE Network Technologies. 2015b. QualNet. Retrieved December 1, 2015, from http://web.scalable-networks.com/content/qualnet.

Space-Time Engineering, LLC. 2015a. Scenargie. Retrieved December 1, 2015, from https://www.spacetime-eng.com/en/products.

Space-Time Engineering, LLC. 2015b. Scenargie Dot Eleven Module. Retrieved December 1, 2015, from https://www.spacetime-eng.com/jp/products?page=Products_Scenargie_Comm_Node_jp.

Thomas Staub, Reto Gantenbein, and Torsten Braun. 2009. VirtualMesh: An Emulation Framework for Wireless Mesh Networks in OMNeT++. In *Proc. SIMUTools '09*. ICST, 64:1–64:8.

András Varga and Rudolf Hornig. 2008. An Overview of the OMNeT++ Simulation Environment. In *Proc. SIMUTools '08*. ICST, 60:1–60:10.

Elias Weingärtner, Hendrik vom Lehn, and Klaus Wehrle. 2011. Device Driver-enabled Wireless Network Emulation. In *Proc. SIMUTools '11*. ICST, 188–197.

Thomas Werthmann, Matthias Kaschub, Mirja Kühlewind, Sebastian Scholz, and David Wagner. 2014. VM-SimInt: A Network Simulation Tool Supporting Integration of Arbitrary Kernels and Applications. In *Proc. SIMUTools '14*. ICST, 56–65.

Wireshark. 2015. Tshark. Retrieved December 1, 2015, from https://www.wireshark.org/docs/man-pages/tshark.html.

Lei Xia, Sanjay Kumar, Xue Yang, Praveen Gopalakrishnan, York Liu, Sebastian Schoenberg, and Xingang Guo. 2011. Virtual WiFi: Bring Virtualization from Wired to Wireless. *SIGPLAN Notices* 46, 7 (2011), 181–192.

Xiang Zeng, Rajive Bagrodia, and Mario Gerla. 1998. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proc. PADS '98*. IEEE, 154–161.

Yongguang Zhang and Wei Li. 2002. An Integrated Environment for Testing Mobile Ad-hoc Networks. In *Proc. MobiHoc '02*. ACM, 104–111.

Received February 2016; revised March 2016; accepted June 2016