WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.2481

RESEARCH ARTICLE

RVIP: an indistinguishable approach to scalable network simulation at real time

Jiejun Kong[1]*, Tingzhen Li[2] and Dapeng Oliver Wu[1]

[1] University of Florida, Gainesville, FL 32611, U.S.A.
[2] Beijing Institute of Technology, Beijing, 100081, China

ABSTRACT

In the hybrid simulation research, we investigate a new approach to build software virtual networks (SVNs) that are indistinguishable from their equivalent real live networks (LNs). We define the concept of 'Network's Interactive Turing Test' based on the similar concept used in the artificial intelligence areas. Our goal is to actualize the interactive and indistinguishable *real–virtual interface pair* (RVIP) for large-scale computer network simulations. By RVIP's support, a single SVN is indistinguishable from its equivalent LN. In the entire hybrid system, multiple LNs and multiple SVNs are connected using many RVIPs in an arbitrary topology and at real time. To actualize RVIP, the following necessary conditions must be satisfied: (i) the performance of the underlying simulation platform must be faster than real time; (ii) all needed changes incurred by introducing any SVN into an LN scenario are put on the simulation's side. To interact with an SVN, RVIP requires that no change is made on any live node; (iii) an SVN does *not* exchange simulation events with LNs, that is,only standard IP protocol interactions between SVN and LN are allowed. (iv) Any LN can be dynamically plugged into the hybrid scenario at real time, just like being plugged into an equivalent purely LN. Compared with existing hybrid simulation efforts on NS-3, QualNet's EXata and OPNET's system-in-the-loop, in this paper, we use the actual RVIP implementation to show that RVIP is a better candidate to pass the Network's Interactive Turing Test owing to the following two advantages: (i) an interactive network tester can easily distinguish the existing hybrid networks from the LNs by using a live topology that *cannot* be simulated, for example, by including the entire live Internet. But RVIP is not vulnerable to such tests. RVIP can support hybrid scenarios with multiple SVNs and multiple LNs connected by an arbitrary network topology, and with the LNs on and off at anytime. (ii) Performance-wise, our studies show that RVIP provides more efficient support in terms of common metrics such as larger throughput limit and smaller extra latency; thus, the simulated SVNs are more indistinguishable from their live counterparts. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
hybrid network simulation; virtual protocol stack; no topological constraint; zero live node change

***Correspondence**
Jiejun Kong, Department of Electrical & Computer Engineering, University of Florida, Gainesville, FL 32611, U.S.A.
E-mail: jiejun.kong@gmail.com

1. INTRODUCTION

To evaluate the performance of a large-scale network, for example, an IP network that consists of thousands of nodes, one can take two major approaches, namely, physical test and computational simulation. Either approach has its advantages and disadvantages. For a physical test, the network nodes under test operate in a real testbed and their performance is evaluated under various testing conditions. This approach features the highest accuracy of the performance measures obtained in the experiments due to using real networks and real physical environments.

However, this approach is not scalable because a largescale real-world experiment requires a large number of hardware units and considerable resources, resulting in a prohibitive cost. In contrast, computational simulation is a more cost-effective alternative that supports flexible and controlled experimentation of arbitrary network scenarios. However, a simulator is always implemented based on some degree of abstraction and simplification of the network protocols and real applications, and thus may miss important characteristics of a real-world system [1]. Therefore, it is appealing to construct a hybrid system which can combine the advantages of both approaches and provide a balanced solution to accomplish the needed tests and performance evaluation [2–5].

In the hybrid system, there are at least one software virtual network (SVN) and at least one LN interacting with

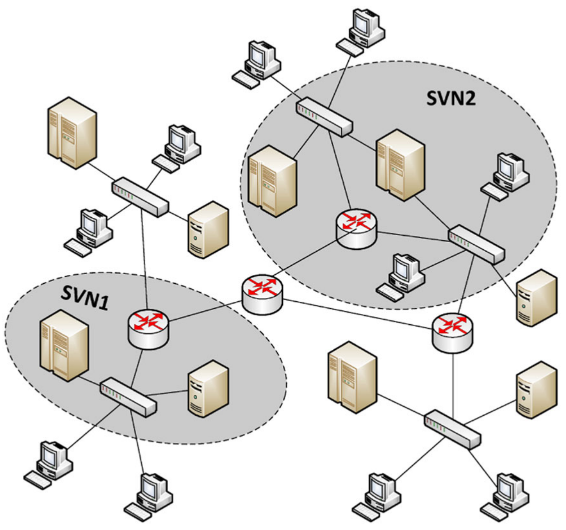


Figure 1. Replace any connected subgraphs of an LN by SVNs.

each other. The most widely adopted method to implement an SVN is using an off-the-shelf network simulator, such as NS-3, OPNET, and QualNet, and then an interface system is devised to connect the LNs with the SVN(s). While the live applications run on the live hosts at real time, the simulator is responsible for modeling the discrete events in the link layer, the physical layer, and the environmental scenario. Live packets generated by the live applications are injected into and extracted from the SVN(s) at real time.

1.1. Our contributions

As shown in Figure 1, in an IP-based LN of any size, we use a software-simulated SVN running on a simulation server to replace a set of live nodes in a *connected subgraph* of the entire network topology.

Definition 1. ***Network's Interactive Turing Test**: For a simulated network system X simulating LN system Y, we define that X passes Network's Interactive Turing Test in terms of a set of network protocol P_{XY} if any third-party network members cannot differentiate X from Y by running any protocol in the set P_{XY}. We say that X runs under Turing-indistinguishable mode if X passes Network's Interactive Turing Test.*

In the ideal form shown in Figure 2, the SVN is *Turing indistinguishable*[†] from its live counterpart. In other words, other live nodes can interact with the virtual nodes inside SVN using the protocols such as Address Resolution Protocol (ARP), routing, and Internet Control Message Protocol (ICMP) as if they are interacting with the SVN's live

[†]We borrowed the name from the AI field, where an AI entity passes the interactive *Turing Test* if it is indistinguishable from the real entity (human brain) in regard to black-boxed interactions.

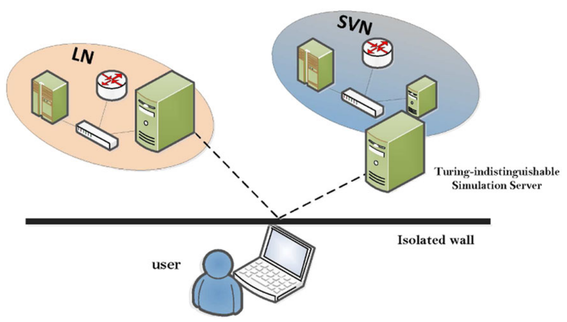


Figure 2. Network's Interactive Turing Test: SVN and its equivalent LN are indistinguishable by black-boxed protocol interactions.

equivalence. This way, the indistinguishability is satisfied between LNs and SVNs at the granularity of network protocols. Unlike the previous efforts, the ideal *Turing-indistinguishable mode* can seamlessly integrate LN's and SVN's protocol stacks together by replacing any LN with an SVN, and vice versa.

To realize the ideal Turing-indistinguishable mode, the following necessary conditions must be satisfied:

(1) **Time criticality:** The performance of the underlying simulation platform must be faster than real time. A large-scale discrete-event simulation platform with high fidelity and faster-than-real-time guarantees is needed to simulate a large number of network nodes in a near-realistic software virtual environment. Currrently in the commercial market, only QualNet and OPNET are approved by the DoD because they follow both civilian and military industrial standards at all protocol stack layers including the physical layer and signal propagation simulations. QualNet can simulate computer networks of more than 1000 nodes in an off-the-shelf multi-core or multi-processor server in real time. According to [6], QualNet is the largest and fastest simulation platform currently available, typically at least three times faster and with three times more network nodes than its competitors including OPNET. Therefore, we choose QualNet as the underlying simulator due to the real-time scalability concerns.
(2) **Intra-SVN system changes:** All needed software and hardware system changes incurred by introducing an SVN into a live network scenario are put on the simulation's side. It is imperative that no system change is made on any live node for the purpose of interacting with the SVN.
(3) **Protocol standard conforming SVN-LN crossover**: An SVN does not exchange simulation events with LNs. In other words, only standard IP protocol interactions between SVN and LN are allowed. Anything that is non-conforming to the standard protocols, including any discrete simulation event, must be stopped at the SVN-LN boundary.

(4) **Hot swappability:** Any LN can be dynamically plugged into the hybrid scenario at real time, just like being plugged into an equivalent purely live network.

All existing solutions, including OPNET SITL (System-In-The-Loop) module and QualNet's EXata system, fail to deliver an implementation of Turing-indistinguishable mode because they violate one or more of these four conditions. (1) In QualNet/EXata, the conditions of intra-SVN system changes and protocol standard conforming SVN-LN crossover are violated. Any live node can easily identify the SVN by intercepting the EXata proprietary messages to and from the SVN. (2) In OPNET, the time criticality condition is violated in an SVN with more than a thousand routers [6]. In addition, OPNET SITL module limits the SVN-LN link to be Ethernet link only. When the connection between the live nodes and the SVN is not Ethernet, for example, ATM/SONET or 802.15 Bluetooth 802.15.4 Zigbee, the live nodes have to switch to the Ethernet link to adapt to OPNET SITL's needs. This means that the condition of intra-SVN system changes is violated.

Therefore, we implement the RVIP interface, a highly-efficient and Turing-indistinguishable coupling of LNs and SVNs, to meet these conditions. RVIP realizes a one-toone correspondence between a virtual IP protocol stack (VIPS) and its counterpart live IP protocol stack (LIPS). It provides translation functions between live packets and virtual packets for multi-layer protocols such as ARP, IP, ICMP, IGMP, OLSR, OSPF, RIP, AODV, TCP and UDP, implements the interaction between live nodes and virtual nodes, and makes live nodes and virtual nodes indistinguishable in terms of network behavior. In addition, all RVIP functions are implemented at the simulation side, there is no protocol stack difference made on a live node for the node to connect with an SVN or an LN. For each SVN, this implies that, on the edge cut-set between the SVN's connected subgraph and the neighboring live nodes, the SVN simulation server host must be equipped with all data link types featured in the edge cut-set. For example, if there are five types of data links in the edge cut-set between an SVN and its live neighbors, 1Gbps Ethernet link, 10Gbps Ethernet link, 802.11 WiFi link, 802.15.4 Zigbee link and 802.16 WiMaX link, the SVN simulation server host must have all the five network interfaces, and connect each interface into its corresponding RVIP pair accordingly. Live nodes in our system include broad range of heterogeneous IP nodes, from light-weight virtual machines such as LXC [7] to any fixed box which the simulator has no modifying permission. This greatly enhances the scalability and flexibility of our hybrid system.

The paper is organized as follows: Section 2 describes related work. In Section 3, we design and implement the RVIP system. Then Section 4 uses actual implementations and measurements to illustrate the advantages of RVIP to pass Turing-indistinguishability tests. Finally, Section 5 concludes the paper.

2. BACKGROUND AND RELATED WORK

As proposed in [8], the hybrid simulation facility can operate in two modes, according to whether the live packet is transparent to SVN or not.

- *Opaque mode:* The simulator treats a live packet as an uninterpreted packet. Live packets sent by a live node flow through the SVN and arrive at other live nodes as if the SVN is a tunnel.
- *Protocol mode:* The simulator is able to interpret and/or generate LN traffic containing arbitrary field assignments. Live nodes can interact with virtual nodes at the granularity of network protocols.

In the opaque mode, when a live packet enters an SVN, the simulator prepends and/or appends additional information, which will be used to process and forward the live packet by the virtual nodes. In the SVN, the packet may be dropped, delayed, reordered, or duplicated. However, as no protocol processing is performed in the opaque mode, protocol-specific traffic operations are impossible in the SVN. When the packet arrives at the virtual destination node, which acts as the proxy of the live destination node, the simulator removes the additional information from the packet and sends it out to the live node. Thus, this mode can only be used in the 'LN–SVN–LN' scenarios.

A distinct feature of the protocol mode is that a node in SVN can interpret any protocol field of an injected live packet and accordingly actualize protocol-specific behaviors. In the protocol mode emulation, there are two types of protocol entities: LIPS, which runs on a live device in the physical world, and VIPS emulated by a network simulator.

- LIPS and VIPS must be consistent with each other in protocol operations. The protocol implementation of LIPS on live system and the simulated protocol modeling of VIPS in network simulator must be strictly in accordance with the protocol standard.
- LIPS and VIPS must share the same time reference. This implies that the network simulator should have real-time (including faster-than-real-time) operation ability. Although it is impossible to achieve perfect timing synchronization between two network clocks [9], the clocks of LIPS and VIPS can be synchronized into a very small difference, for example, at the scale of 100 ms [10].
- The packets between two end-to-end VIPS are always optimized by the simulation, making them different from live packets between two end-to-end LIPS. To realize the fusion of LNs and SVNs, an interface system is needed to convert the packets sent from one side to the other.

Based on the NS-2 simulator, Mahrenholz and Ivanov [2] improved the emulation facility by allowing live applications to be connected to specific nodes in the NS-2's

SVN. It supports two working modes, the single host extension and the distributed clients extension. The former allows live applications run on a single live host only. The latter introduces a distribution layer between the live nodes and the NS-2 SVN, allowing live applications to run on several physical hosts. Kristiansen and Plagemann [11] evaluated the efficiency of this distribution layer. NS-2's successor, NS-3, also supports the single host emulation capability. Live nodes, some of them can be virtualized machines, are connected through a TUN/TAP device of the Linux kernel to proxy nodes in the SVN. Alvarez *et al.* [12] developed a distributed client extension for NS-3 adopting the same approach used in NS-2. Just like the existing systems, RVIP supports both the single host mode where a (virtualized) host's TAP device is mapped to a virtual node, and the distributed mode where a set of live nodes is connected to the SVN via LAN. However, RVIP does *not* require any change in the live nodes (such as installing or configuring an extra software), greatly simplifies the operational setup, and aims at the Turing indistinguishability.

Based on the OMNeT++ simulator, Staub *et al.* [13] proposed an emulation framework 'VirtualMesh', which requires live nodes to create virtual wireless interfaces with parameters configured just like real interfaces. The virtual interface intercepts live traffic and forwards it to a simulation model. The live nodes and the simulator are connected by a UDP session, through which the live node can configure, manage, and communicate with the simulator. The same method is also adopted by Weingärtner *et al.* [14]. These efforts only support the opaque mode in hybrid simulation and require installing some new modules on live nodes (which may not grant the needed administrative permission).

There are previous efforts to improve the simulation scalability. Weingärtner *et al.* [15] tried to increase the simulation network's scale by using 'virtual time'. Basically, the clocks on the network nodes are slowed down for the network applications, so that the applications work slower than what should be in the real scenario. This allows more discrete events be processed by the simulation server in unit time. Unfortunately, this approach does not apply to real-time simulation systems with real human and live devices in the loop.

Liu [16] proposed the concept of 'network immersion' so that the virtual network is indistinguishable from a physical testbed in terms of network behavior, but it focuses on Virtual Private Network (VPN) rather than the protocol (e.g., AODV and ARP) interaction between LNs and SVNs. Liu *et al.* [17] constructed a mapping between a virtual interface in SVN and a TAP device in client machine, and the connection between them is realized by VPN, supporting geographically distributed applications to dynamically connect to the real-time network simulator. Compared with RVIP, this approach needs to install VPN clients on live nodes, and the thoughput of the VPN connection is significantly reduced. In order to produce accurate result, the VPN emulation infrastructure requires a tight coupling (i.e., high-band width low-latency connection) between the live hosts and the real-time simulator. Similar to other existing emulators, it needs administrative privileges to establish the emulation infrastructure (e.g., setting up VPN on the client machines).

In the commercial market, OPNET corporation [5] has developed a module called system-in-the-loop (SITL), which transfers packets between LN and SVN to achieve interaction between live node and virtual node. By SITL, developer can construct two types of connections, namely *LN–SVN–LN* and *SVN–LN–SVN*. QualNet provides the interface of IPNE [18] and its successor EXata. IPNE supports NAT-YES, NAT-NO, and TrueEmulation working modes. The first two modes are opaque, while TrueEmulation is the protocol mode. The aforementioned interfaces can accomplish the interaction between LN and SVN with limited real-time scalability and to a certain degree measured by the opaque mode or the protocol mode. The TrueEmulation mode of IPNE is difficult to configure, in particular for a large number of changes made on the live nodes; OPNET only supports a chain connection of LNs and SVNs. All the aforementioned interfaces cannot realize a simulation where LNs and SVNs are arbitrarily mixed. Additionally, to our best knowledge, neither OPNET/SITL nor QualNet/EXata has published performance analysis and evaluation of their LN-SVN interfaces.

3. REAL–VIRTUAL INTERFACE PAIR

3.1. Design and implementation

We design an interface of RVIP that realizes a one-to-one mapping between a LIPS (e.g., on a real IP network interface card) and a VIPS. As demonstrated in Figure 3, a live node A has a real interface eth0, and its shadow node A' has a virtual interface wlan0, and then eth0 and wlan0 form a RVIP, where eth0 is called *RVIP-Real*, wlan0 is called *RVIP-Virtual*, and they have the same IP address and hardware Medium Access Control (MAC) address. RVIP seeks to implement the same behavior on RVIP-Real and RVIP-Virtual. In other words, when RVIP-Real sends out a live packet, the RVIP-Virtual sends out an equivalent packet in the SVN. Likewise, when RVIP-Virtual receives a virtual packet from other virtual nodes in the SVN, the RVIP-Real receives an equivalent packet. Thus, RVIP-Real and RVIP-Virtual act like a single IP protocol stack with double faces, one in the actual world and the other in the virtual world.

RVIP has three basic modules: *packet capture*, *packet translation*, and *packet construction*. After the *packet capture* module captures a live packet sent by RVIP-Real in real time, the *packet translation* module will analyze and convert the live packet to its equivalent virtual form. Then the virtual packet is inserted into the SVN and processed by the simulator. Similarly, when a virtual packet arrives at RVIP-Virtual, RVIP-Virtual does not deliver it to upper-layer protocols; instead, it translates it into a live form and sends it to the RVIP-Real.

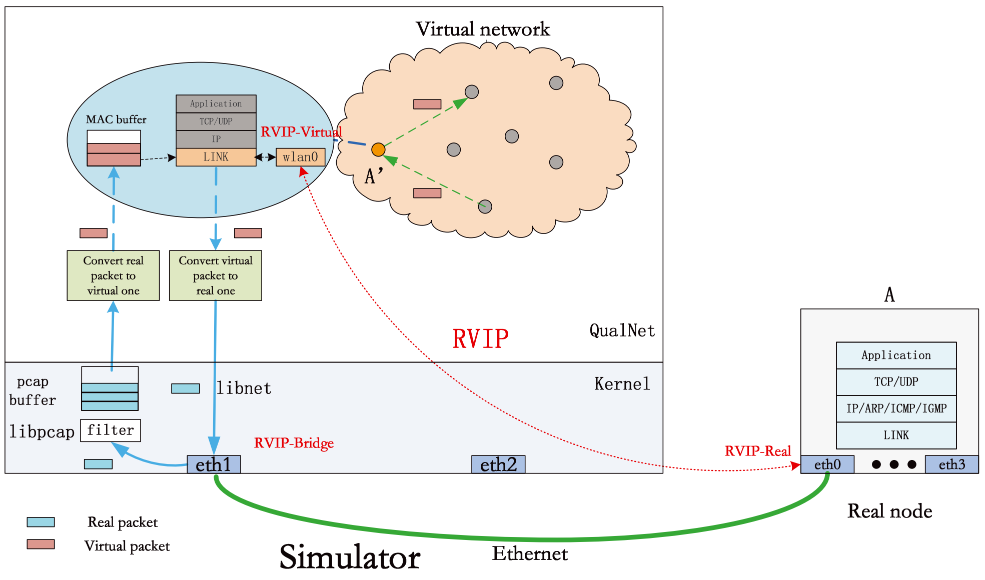


Figure 3. The structure of real–virtual interface pair.

As Figure 3 shows, there is a live node connecting to the simulation server via an Ethernet Network Interface Card (NIC) eth1 called *RVIP-Bridge*, which serves as a bridge between a live node and the SVN. Several RVIPs can share a same RVIP-Bridge. At the beginning of the simulation, the simulator side does the following tasks:

- disables the IP and upper-layer protocols on the virtual node and keeps its MAC and physical layers alive;
- registers an entry in the RVIP chain table for every RVIP;
- assigns a *libnet* handle [19] and a *pcap* handle [20] on RVIP-Bridge for every RVIP;
- sets the *pcap* filter for every RVIP, making it only receives packets sent by RVIP-Real.

After that, every RVIP has a *libnet* handle, a *pcap* handle, and a *pcap* buffer. *Libnet* handle is used by RVIP-Virtual to construct live packets, and *pcap* handle is used to read packets buffered in the *pcap* buffer.

The simulator runs in real time and keeps pace with RVIP-Bridge in the process of simulation. All packets sent by RVIP-Real get through the *pcap* filter and are buffered in the *pcap* buffer. A timer is set in the simulator, and its time-out interval is T (currently set to 1 ms). When the timer times out, it triggers the RVIP to poll the RVIP-Real and RVIP-Virtual interfaces successively. If the *pcap* buffer of an RVIP has packets, all of them are taken out by this RVIP's *pcap* handle. When a live packet is captured, RVIP will translate it to a virtual one and insert it into the MAC buffer of the RVIP-Virtual. Then the control is handed over to the simulator engine, which schedules and processes the virtual packet.

3.1.1. Discussions.

3.1.1.1. Per-protocol translation. In RVIP, the LIPS versus VIPS translation is carried out at the granularity of single protocol. In other words, the real versus virtual Turing-indistinguishability support is added on a per-protocol base. Currently, we have provided the translation functions for the following protocols: OSPF, RIP, OSLR, AODV, UDP, TCP, ICMP, IGMP, IP, and ARP. The function is implemented in a modular framework to ease the efforts to add new protocol translation unit. The *packet construction* module is implemented at each RVIP-Virtual's MAC layer. When RVIP-Virtual receives a virtual packet, it calls its *libnet* handle to construct a new live packet and sends it out.

3.1.1.2. Relationship between RVIP-Real and RVIP-Virtual. Essentially, RVIP-Real is the master who simply runs its equipped IP protocol stack without any change required in the live part of the network. RVIP-Virtual acts as the slave proxy of the RVIP-Real in SVN and only enables its MAC layer and physical layer functions. RVIP creates an integrated protocol stack, in which the IP and above layers are the live ones on live nodes, and the MAC and physical layer are the virtual ones emulated by the simulator. This way, there is no change required for the live nodes joining into the SVNs and interact with virtual nodes.

3.2. Constructing hybrid simulations in any topology

3.2.1. Mapping single virtual node to live node.

RVIP supports both distributed connections and single host connections when establishing the mapping between a live node and a virtual node. In the distributed mode,

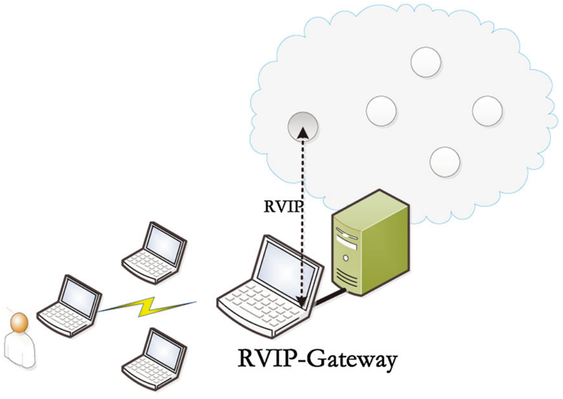


Figure 4. *RVIP-Gateway* combines LN with SVN.

any device supporting TCP/IP and having a LAN card can be connected to the SVN to act as RVIP-Real. This mode distributes the workload to multi-hosts. In the single host mode, live nodes realized by virtualization technology, such as VMware and LXC, run on the same host with SVN, and their TAP devices are connected to SVN as RVIP-Real via the Linux kernel bridge. In summary, applications and protocols above the link layer on the live nodes run as usual without any modification, while the lower layers below MAC are connected via the SVN. Packets generated by live applications can be injected into and routed through the SVN. Then the effect of network delays and protocol processing can be imposed on LN traffic.

3.2.2. Combining LN and SVN.

In Figure 4, a live node called *RVIP-Gateway* connects a multi-node LN and an SVN. The gateway can be an Internet gateway when LN is the Internet. The gateway has two network interfaces, one directly connecting with the LN and the other directly connecting with an SVN. On both interfaces, RVIP-Gateway runs the same routing protocol of the virtual nodes in SVN and forwards packets from one side to another, so that the topology information of one network can be introduced into the other. Now, we could *ping*, *traceroute* each node in LNs or SVNs. When inspecting a live node's routing table, there are routing entries to those virtual nodes after the live node has interacted with virtual nodes on the network protocol level. As a result, the SVNs improve the flexibility and scalability of the target network modeling, and the LNs enable the actual users and the real devices to participate in the simulation at real time.

3.2.3. Connecting multi-SVNs.

In Figure 5, a live node called *RVIP-Router* has multiple network interfaces, each of them connecting to an SVN and forming an RVIP. On every network interface of the RVIP-Router, the same routing protocols used by the corresponding SVN run on the interface and interact with the virtual nodes. The RVIP-Router exchanges topological information with every directly connected SVN and distributes them into other SVNs. This way, virtual nodes in

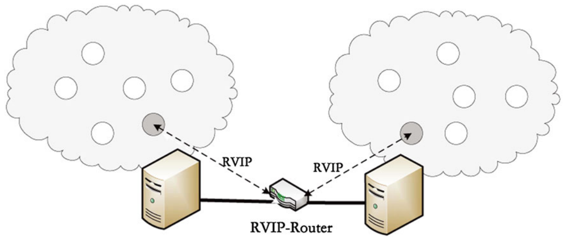


Figure 5. *RVIP-Router* connects multi-SVNs.

one SVN can communicate with the ones in other SVNs. As a result, we can use multiple simulation instances running at real time and at different simulation servers to enlarge the scale of network simulation. Each simulation server only takes charge of one SVN in the simulation, and the RVIP-Routers connects all of them.

Based on the aforementioned basic patterns, we can construct more complex hybrid simulation scenarios where LNs and SVNs arbitrarily mixed without topological constraints. This realizes a far more scalable network simulation capability than the underlying simulation engine.

3.2.4. Large-scale simulation.

A large-scale network may have tens of thousands of nodes, making it difficult to simulate in real time. When network scale increases, a single simulator is unable to accomplish the real-time simulation task. An existing solution is using parallel computing to address the scalability concern and real-time concern. For example, PDNS [6] relies on computing clusters and supercomputers to simulate hundreds of thousands stations, with expensive hardware cost and decreased cost efficiency.

RVIP presents another solution with low cost. In RVIP, multiple simulation instances can run independently with each other in real time. Each simulation instance only takes charge of a single SVN in the entire simulation. The connection between different SVNs is realized by RVIP-Routers, which are live devices playing the role of hybrid router between different network parts. RVIP-Routers and SVNs can run the same set of routing protocols and other network protocols in the IP protocol stack, so that all SVNs and live nodes are seamlessly integrated as a monolith.

RVIP is in general a distributed solution. Like RVIP, existing solutions like *PDNS* and *PRIME* [21] also adopt a distributed framework. However, unlike RVIP, they use a set of parallel synchronization algorithms, instead of IP networks, to synchronize different simulation instances. For example, when packets are forwarded to a router simulated by the remote simulator, the local simulator must send a simulation event to that simulator. What they have realized are synchronization and interaction between different simulation instances at the granularity of discrete simulation event.

In contrast, RVIP is a pure networking solution that does *not* rely on the simulation events fired among different simulation instances. Unlike all previous work, RVIP seeks to eliminate the difference between an SVN and an LN in regard to a third-party live device/network. In a hybrid network scenario with multiple LNs and multiple SVNs, RVIP allows any LN be replaced by an equivalent SVN, and vice versa. In a nutshell, if the entire network topology is treated as a set of connected network tiles, each tile can be instantiated as an SVN or an equivalent LN without extra constraints in the SVN/LN tiling pattern.

In addition, we can dynamically connect any (IP-based) LN into an existing live hybrid scenario and consequently attract more live devices, novel radios, and actual users via the LN. This feature of *dynamic LN plug-in* is a new real-time simulation capability that improves scalability and flexibility of our solution. The Internet itself can be connected into an RVIP scenario as a dynamic LN plug-in.

3.2.4.1. Wireless network discussion. In wired networks, RVIP can easily address the scalability and flexibility concerns. Given the real-time capability of the underlying simulation engine (QualNet or equivalence), we can replace any LN (any connected subgraph of a reasonable size in the entire network topology) with an equivalent SVN running on the simulation engine at real time, or vice versa.

However, for wireless networks, owing to node mobility and wireless signal propagation, two neighboring SVNs may have to exchange *environmental* discrete simulation events (in particular, node mobility and wireless signal propagation events) to deliver correct simulation results. Currently, RVIP can only be used in wireless networks with Frequency Division Multiple Access (FDMA) or Time Division Multiple Access (TDMA) radios. All wireless nodes using the same frequency or TDMA schedule form an SVN. We are working on the following tasks to enable RVIP for other types of wireless networks in the real world.

- *Definition of geographical theater*: The entire wireless network area is divided into hexagon theaters, similar to cellular network's cells. Each theater is a wireless SVN (WSVN), which runs on a single simulation engine. The diameter of the theater is greater than the wireless interference radius.
- *Cross-WSVN mobile node*: A WSVN monitors the mobility pattern of its internal mobile nodes. If a mobile node steps into a neighboring WSVN, the previous WSVN and the next WSVN must exchange discrete events prior to the crossover.
- *Cross-WSVN signal propagation*: A WSVN monitors each wireless transmission event and notifies all neighboring WSVNs about the event.

However, because QualNet's physical layer and signal propagation layer are *not* open to the public, currently, we are unable to realize the WSVN features in our QUP (QualNet University Program, currently Scalable Education Program)-based implementation.

4. INTERACTIVE TURING TEST CASE STUDIES

This section presents case studies to demonstrate the advantages of RVIP over the existing solutions, in terms of capability of passing the Network's Interactive Turing Tests.

4.1. Tests on topological changes

We first carry out an experiment under the change of network topology. Multi-SVNs and multi-LNs, including the real world Internet, are mixed in the scenario. As Figure 6 shows, there are three SVNs, and each has three virtual nodes connected by wired links with 20-ms propagation delay. Live nodes, D and E, both have two NICs embedded into SVNs as RVIP-Reals. Either D and E acts as an RVIP-Router that connects two SVNs. We then instantiate two VIPSs, F and G, in server B and C using LXC virtual machine. F and G also have two NICs: one is a TAP device functioning as an RVIP-Real towards an SVN and the other as an interface to the real world Internet, namely interface X on F and interface Y on G. X and Y are configured with static routes in high priority so that they are selected as the prior routing interfaces if the Internet connection is on. All nodes (simulated nodes in SVNs as well as D, E, F, and G) also run RIPv2 routing protocol during the experiment. The virtual nodes run QualNet's RIPv2 protocol, while the live nodes run RIPv2 protocol from the Quagga [22] package. We measure the round trip time (RTT) between D and G using *ping* and record the route between B and G using *traceroute* at the same time. Before the experiment, the interfaces to the real world Internet, X and Y, are connected via an Ethernet cable. During the experiment, we disconnect X and Y for a while and then reconnect them. In the procedure, we find that, before the disconnection, the route path between D and G is D-SVN2-F-G, and after the disconnection, it becomes D-SVN1-E-SVN3-G. The RTT measurement between D and G shows a corresponding change in the experiment

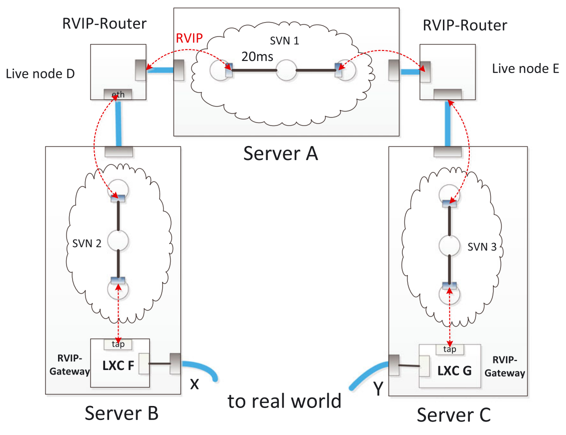


Figure 6. Scenario of the case study.

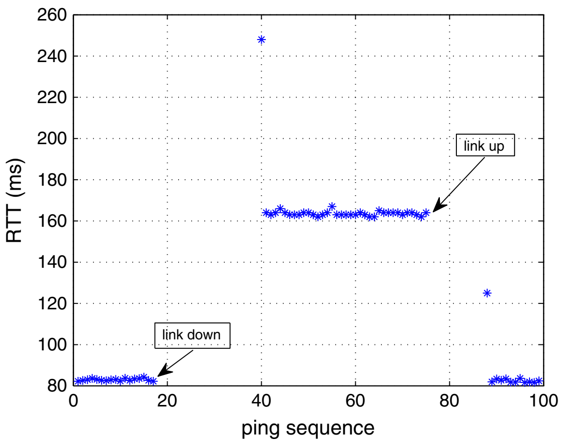


Figure 7. RTT measurement in the experiment.

(Figure 7). We can draw the conclusion that RVIP connects SVNs and LNs following the depicted hybrid topology, and IP networking is accomplished because of fair contributions from routing protocols running on both live nodes and virtual nodes.

4.2. Performance tests

4.2.1. Accuracy test.

The major overhead of RVIP comes from packet capturing, interpretation, and construction when packets are transferred across the SVN–LN boundary. The overhead incurs inaccuracy in a simulated network as compared with its purely live equivalence. To ensure accuracy in hybrid simulation, RVIP should minimize the overhead. We conduct experiments to evaluate the accuracy of RVIP, mainly focusing on overhead in regard to network bandwidth and latency.

As shown in Figure 8, we construct a virtual wireless network in QualNet. In the SVN, there are four wireless nodes following a chain topology. Each node is configured with 802.11b MAC protocol (without Request To Send/Clear To Send (RTS/CTS)), 2-Mbps transmission rate and approximately 70-m transmission radius. The distance between two neighbors is 50 m. To focus on data traffic measurement, static routes are used in the network to remove routing traffic. Node 1 sends CBR packets to node 4 with the following Constant Bit Rate (CBR) parameters: CBR payload is 512 bytes, and packet sending rate is set to 1/s, 5/s, 10/s, 50/s, 100/s, 500/s, and 1000/s, progressively.

We conduct two comparative tests. Test1 runs pure-software simulation without enabling RVIP, and Test2 runs a hybrid simulation where node 3 was mapped to a live laptop. As Figure 8 shows, the RVIP-Virtual V interface and RVIP-Real interface R is connected by the RVIP-Bridge eth0. R/eth0 is a 1000-Mbps Ethernet NIC. RVIP's overhead is highlighted by comparing the performance measured in the two tests.

Let us inspect the packet processing workflow of the hybrid simulation. A virtual packet sent by node 2 arrives

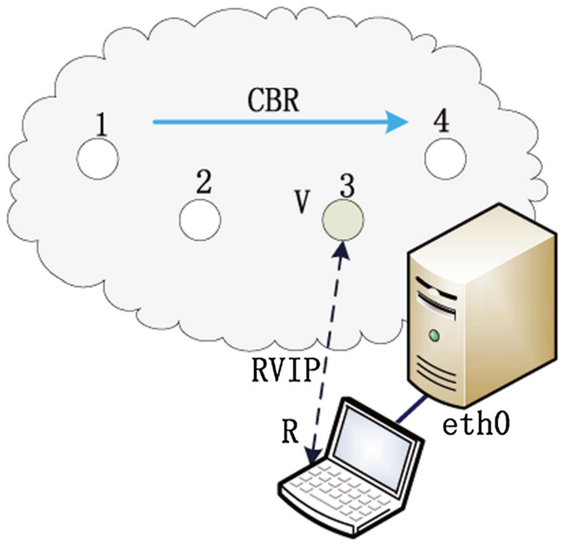


Figure 8. The accuracy test scenario.

Table I. Delay at different packet processing phases.

Phase	Task	Average time consumption
$t1-t0$	Construct a live packet according to the virtual packet	6 μs
$t2-t1$	A live packet is transmitted by RVIP-Bridge and propagated	max ≈ 1514 bytes *8/1000 Mbps = 12 μs
$t3-t2$	Processed by the live node	Varies per operation
$t4-t3$	Same as $t2-t1$	Same as $t2-t1$
$t5-t4$	Waiting in the *pcap* buffer	*pcap* clears the buffer every ≤1 ms
$t6-t5$	Construct a virtual packet according to the live packet	2 μs

at V at $t0$; then the RVIP interprets the virtual packet and constructs a corresponding live packet, which takes $t1-t0$. Then the live packet is sent out by the RVIP-Bridge at $t1$ and arrives at R at $t2$. $t2-t1$ is the transmission and propagation time of the live packet. From $t2$ to $t3$, the laptop processes the live packet, such as doing route table lookup. Then the processed packet is sent out by R at $t3$ and arrives at RVIP-Bridge at $t4$. $t4-t3$ is nearly identical to $t2-t1$. After being received by RVIP-Bridge, the live packet is buffered in the *pcap* buffer until $t5$, when RVIP queries the buffer to fetch live packets. Then RVIP constructs a corresponding virtual packet and inserts it into V's MAC buffer, which takes $t6-t5$. We have conducted experiments and recorded all aforementioned timing points. The results are shown in Table I, from which we know that all the phases result in additional delay compared with the purely simulated scenario. Among all phases, $t5-t4$, the waiting time

Table II. Differential measurement to estimate RVIP overhead.

Packet rate	Performance	Pure Sim	Hybrid Sim	Difference	Deviation (%)
1/s	Latency (ms)	11.30	11.77	0.47	4.2
	Throughput (bps)	34.4K	34.4	0K	0
5/s	Latency (ms)	11.11	11.53	0.42	3.8
	Throughput (bps)	165.6k	165.6	0K	0
10/s	Latency (ms)	11.07	11.51	0.44	4.0
	Throughput (bps)	329.6K	329.6K	0K	0
50/s	Latency (ms)	11.04	11.53	0.49	4.4
	Throughput (bps)	1640K	1640K	0K	0
100/s	Latency (ms)	1480	1430	−50	−3.4
	Throughput (bps)	3072K	3072K	0K	0
500/s	Latency (ms)	2860	2800	−60	−2.1
	Throughput (bps)	3072K	3064K	−8K	−0.2
1000/s	Latency (ms)	2880	2790	−90	−3.1
	Throughput (bps)	2632K	2640K	−8K	0.3

in the *pcap* buffer, is the most significant one. This profiling result tells us to optimize the packet capture mechanism in our future work.

Other performance results are shown in Table II. We learn that the deviation of network throughput caused by RVIP is insignificant (within 5% deviation). When the traffic load is low (packet sending rate $\leq$ 50/s), in the pure simulation scenario, there is no packets loss, and the average end-to-end delay is about 11 ms; in hybrid simulation scenario, the extra delay caused by RVIP is about 0.5 ms. As the traffic load increases, packet loss appears, and the average end-to-end delay rapidly increases. The deviation caused by RVIP stays as insignificant compared with the measured results.

4.2.2. Stress test.

4.2.2.1. Distributed mode. The distributed mode allows live nodes to be realized on hosts other than the simulation server, thus bringing appealing features such as dynamic LN plug-in and multi-SVN connections. We conduct the following stress experiments under the distributed mode. In the scenario, live data streams originate from a live node, go through an SVN, and arrive at another live node. By comparing the metrics of the data stream at both live terminals, we can assess the overhead of RVIP compared with an equivalent LN. This method is also adopted in [11].

As Figure 9 shows, we construct a QualNet-based SVN where two nodes are connected by an 802.3 link with 10-Gbps bandwidth and 1-μs propagation delay. Each virtual node is mapped to a live laptop by RVIP. One laptop serves as the traffic source, and the other as the traffic destination. The simulation server has an Intel Core 2 Duo 2.9-GHz processor and 2-GB RAM; each laptop has an Intel Core i5 2.4-GHz processor and 3-GB RAM. All hosts run Ubuntu Linux with a 2.6.35 kernel, and all NICs perform at 1 Gbps.

We use a simple SVN so that the simulation server's CPU cycles are mostly consumed by processing events related to packet capturing and transferring from/to the

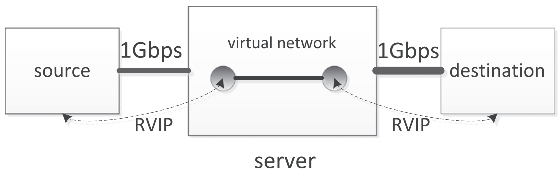


Figure 9. The stress test scenario.

LN. The packet operations invoke system calls to copy full packets between the operating system's kernel and user spaces and are thus much more expensive than processing the relatively simple simulation events.

In the test, the source sends CBR traffic to destination using the Traffic Generator (TG) tool [23], with packet size (P) set to 64, 512, and 1024 bytes, progressively. During the experiment time, we also progressively change the packet sending rate (W) from 1K packet-per-second (pps) to 60K pps with 5K pps increment per step. We monitor the sending and reception rates and also measure the round trip time (RTT) via *ping*.

The relationship between sending rate and reception rate is demonstrated in Figure 10. There is a clear point of saturation for all P, and the saturation point is about 40K pps when all additional traffic injected is dropped by the server. Considering that the live and virtual links are still far beyond their capability limit, the saturation point can be attributed to the exhaustion of the server's computational CPU power. We observe that the server's CPU is too busy according to the system monitor; thus, the simulation server fails to fetch packets from its *pcap* buffer (PB) timely and results in the buffer overflow, and packet drops. In RVIP's implementation, there are two buffers, PB and RVIP-Virtual's MAC buffer (MB). Packets sent by RVIP-Real are buffered in the PB, and the simulator will clear N packets and insert them to MB every T (1 ms per our design). N is determined by the free space of MB and P. We further conduct experiments to study the relationship between N, T, and P when W is 50K pps. Results

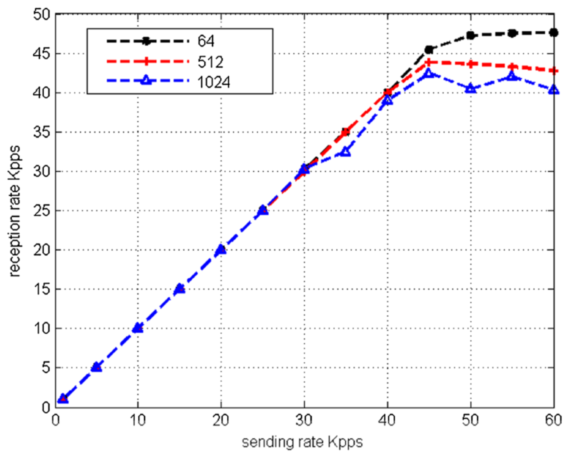


Figure 10. Relationship between sending rate and reception rate.

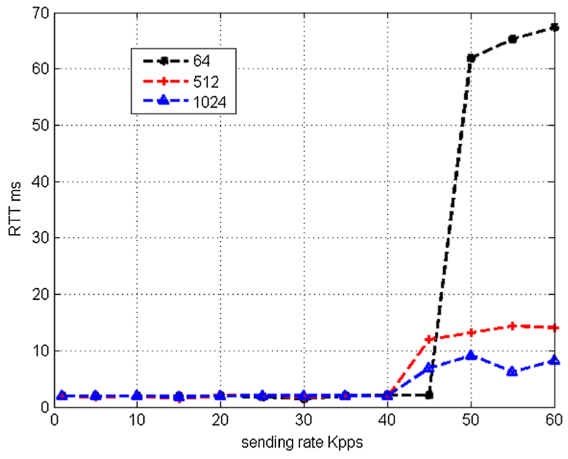


Figure 11. Relationship between sending rate and RTT.

Table III. Relationship among P, N, and T.

P (byte)	N (packet)	T (ms)	N/T (Kpps)
64	1453	32.0	45.4
512	277	6.7	41.3
1024	144	3.6	40.0

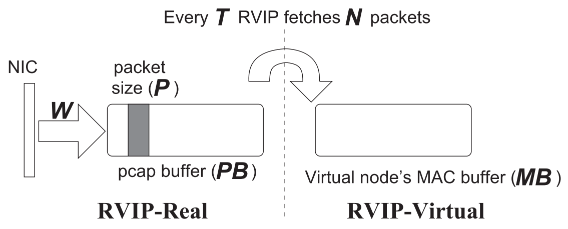

Every T RVIP fetches N packets
NIC
W
packet size (P)
pcap buffer (PB)
Virtual node's MAC buffer (MB)
RVIP-Real
RVIP-Virtual

Figure 12. Relationship among P, N, and T.

Table IV. Measurement on a server with more computational resources.

P (byte)	N (packet)	T (ms)	N/T (Kpps)
64	7260	19.0	382.1
512	1381	4.2	328.8
1024	720	2.3	313.0

are illustrated in Table III. If P is smaller, the simulator can fetch more N every time, and the bigger the N is, the more time it will take the simulator to handle them, resulting in a larger T. N/T is the fetching rate of the simulator. Figure 12 explains the major reason of the saturation, that is, when the live sending rate W is larger than N/T, which is mostly limited by the simulation platform's computational power and the configuration of SVN, there comes the PB overflow, and packet drops. The recorded data shown in Table III match the saturation points shown in Figure 10.

Table IV shows a similar result on a simulation server with more computational resources, which runs on a Thinkpad W510 with Intel Core-i7 CPU, 16 GHz and 3-GB RAM.

Figure 11 shows the measurement of RTT with an increasing W. As our virtual link is configured with a negligible delay, the RTT is mostly attributed to the timing overhead incurred by RVIP. From the figure, we notice that RTT is a constant before reaching the simulation server's saturation point, and then RTT increases significantly as W exceeds the saturation point. The most significant RTT increment happens at the saturation point when P is 64 bytes. The underlying reason is that, as there are an increasing number of packets (N) to be inserted into the SVN, the average PB queuing time of the *ping* packets increases and then causes the observed jump in RTT measurement. For a smaller packet size, the queue size is larger, and the expected waiting time is longer (Figure 12).

4.2.2.2. Single host mode. RVIP can also be used in single host mode scenarios for a comparison with NS-3's TapBridge model. We did the following tests to compare the performance between the two systems on the same equipment (Table V).

- Test1 (purely LN) is carried out with two laptops (used in Section 4.2.2) directly connected via 1-Gbps Ethernet link.
- Test2 (TapBridge with NS-3) is carried out on an NS-3-based SVN where two nodes are connected by a CSMA[‡] channel with 1-Gbps bandwidth and 1-µs propagation delay. Each virtual node corresponds to an LXC virtual machine via NS-3's TapBridge UseBridge mode. NS-3 and both LXCs are running on the single simulation server (used in Section 4.2.2).

[‡]NS-3's CSMA model is designed to approximate the 802.3 Ethernet model. For more details, please refer to Chapter 9 of NS-3 Model Library.

Table V. Results of the four tests.

	RTT (ms) before download	Throughput (Mbps)	RTT (ms)
Test1	0.36	344	0.4
Test2	0.4	28	460
Test3	1.2	160	4.8
Test4	1.4	51	15.3

- Test3 (RVIP with QualNet) is carried out on a QualNet-based SVN where two nodes are connected by an 802.3 link with 1-Gbps bandwidth and 1-μs propagation delay. Each virtual node corresponds to an LXC via RVIP. QualNet and both LXCs are running on the single simulation server (used in Section 4.2.2).
- Test4 (SITL with OPNET) is carried out on an OPNET-based SVN where two nodes are connected by an 802.3 link with 1-Gbps bandwidth and 1-μs propagation delay. Both nodes connect to an OPNET SITL virtual gateway corresponding to an LXC. OPNET and both LXCs are running on the single simulation server.

In each test, we use *scp* on one host (live laptop or LXC) to download a massive file from the other one and measure the RTT by *ping*. Before the download, the average RTT of Test1 and Test2 are nearly the same, while Test3's RTT is about 1.2 ms, which is a constant as analyzed before. During the download, the throughput in Test1 can reach 344 Mbps with 0.4-ms RTT. This heavy load exceeds both NS-3 and RVIP's saturation points. RVIP (Test3) can achieve 160-Mbps throughput with 4.8-ms RTT, while NS-3 (Test2) can only achieve 28-Mbps throughput with much larger RTT (about 10^2 times more). From our saturation measurements, the OPNET/SITL duo's performance is about three times slower than that of the QualNet/RVIP duo, similar to that of various OPNET versus QualNet performance comparisons reported in [6].

4.2.3. Impact on packet flow's jitter.

This experiment is devoted to evaluate RVIP's impact on the packet flow's jitter. Owing to the limited computational resource available on the simulation host, there is a deviation between the simulated scenario and its live equivalence. We employ the following approach to measure the deviation.

(1) In the test, the sender sends total N packets. To a particular packet whose sequence number is i ($1 \le i \le N$), its transmission time at the sender is t_{si}, and arrival time at the receiver is t_{ri}. All this information is recorded by the traffic generator TG.

(2) The time interval between consecutive packets is a constant in the flow. Thus, there is a linear relation between i and t_{si} or t_{ri}:

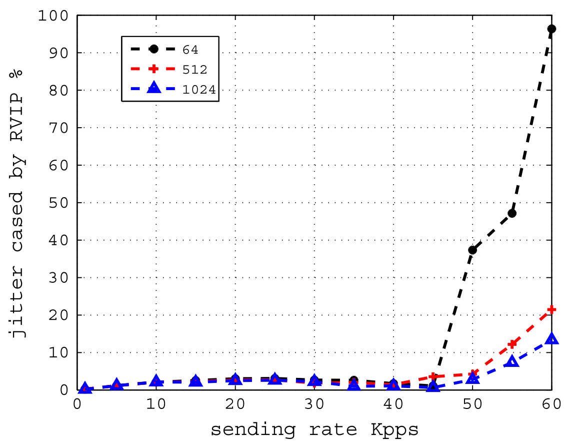


Figure 13. Relationship between W and $\frac{\tau}{T}$.

$$t_{si} = \alpha_s i + \beta_s \tag{1}$$

$$t_{ri} = \alpha_r i + \beta_r \tag{2}$$

Here, α_s and β_s are coefficients at the sender's regression line; while α_r and β_r are the ones for the receiver. All coefficients are calculated by the least square method.

(3) We define the jitter as the average deviation from the linear regression line, denoted as

$$\tau_s = \frac{\sum_{i=1}^{N} |t_s - (\alpha_s i + \beta_s)|}{N} \tag{3}$$

$$\tau_r = \frac{\sum_{i=1}^{N} |t_r - (\alpha_r i + \beta_r)|}{N} \tag{4}$$

(4) The jitter deviation caused by RVIP can be computed by $\tau = |\tau_s - \tau_r|$. τ is a fraction of T, the ideal packet interval determined by the packet sending rate (W).

We use this method to measure the relation between W and $\frac{\tau}{T}$ when packet size (P) is set to 64, 512, and 1024 bytes, progressively. The results are shown in Figure 13. Similar to Figure 11, before W reaches the saturation point, the jitter deviation caused by RVIP is negligible, and then it increases significantly after W exceeds the saturation point. In addition, the smaller P is, the sharper is the increase.

4.2.4. Scalability impact of underlying simulator.

In our paradigm, the underlying simulation engine's capability determines the size of a single SVN. RVIP is a scalable replacement of EXata, which is built on top of QualNet, an *a priori* parallel simulation engine.

Our test runs QualNet 5.2 on a Thinkpad W510 host with Intel Core-i7 CPU, 1.6 GHz and 3-GB RAM. We construct a wireless network where 500 nodes are randomly placed in a 1500 m×1500 m surface with TWO-RAY propagation

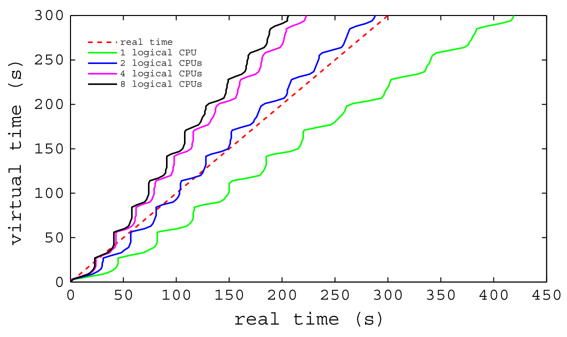


Figure 14. Relation between real time and virtual time in Test1.

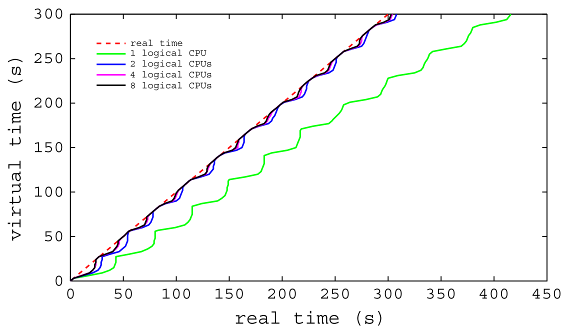


Figure 15. Relation between real time and virtual time in Test2.

model. Each node has an 802.11 radio with 2-Mbps rate and runs the proactive RIP routing protocol. We conduct two comparative tests. The simulator runs at the top speed of its virtual time clock in Test1, and at the real time clock in Test2. We progressively run the tests using one, two, four, and eight logical CPUs in the physical Core-i7 CPU. Figure 14 shows the relation between real time clock and virtual time clock using different number of logical CPUs in Test1. It shows that parallel computing increases the simulation speed to be faster than real time as the number of logical CPUs grows, but the acceleration rate is diminishing. Figure 15 illustrates that the simulator runs nearly at real time, except that the non-parallel test with single logical CPU is too slow to match the real time clock. Besides, from a closer look shown in Figure 16, the more logical CPUs the parallel simulator uses, the more closer the virtual time line approaches the real-time line.

4.2.5. Test summary.

In this section, we have shown that RVIP surpasses QualNet-based EXata in topological tests, as well as N3-based and OPNET-based solutions in performance tests. In order to achieve Turing indistinguishability by passing the Network's Interactive Turing Tests, RVIP is a better candidate than these existing solutions.

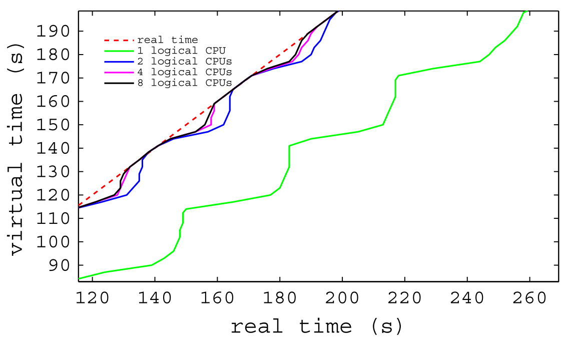


Figure 16. Partial enlarged detail in Test2.

5. CONCLUSION

Network emulation has been proposed to overcome the disadvantages of pure-software simulation and physical test for large-scale network research. The emulation facility can be subdivided into the *opaque mode* and the *protocol mode* according to whether the live packet is transparent to SVN or not. In comparison, the protocol mode has more advantages because it can realize protocol-specific interactions between LNs and SVNs. In this paper, we further enhance the protocol mode towards its ideal case, namely the Turing-indistinguishable mode. This new ideal mode aims to implement SVNs that can pass the Network's Interactive Turing Tests, so that any SVN and its counterpart LN would be indistinguishable via protocol interactions from the angle of any third-party live node/network. This requires that live nodes with standard IP protocol stack must strictly follow standard IP protocols in interacting with the SVNs, and there are no simulator-specific changes required on any live node.

We design the RVIP interface to realize the new ideal mode [24]. In a hybrid simulation scenario, multiple LNs and multiple SVNs are arbitrarily mixed with no modification required on any live nodes. RVIP realizes the mapping between a live protocol stack and a virtual protocol stack (VIPS) by presenting a progressive series of connection patterns, that is, *mapping single virtual node to live node*, *combining LN with SVN*, and *connecting multi-SVNs*. Then we can construct more complex hybrid scenarios by combining these patterns. We constructed real-time hybrid scenarios with multiple LNs and multiple SVNs, with each SVN instantiated on a single simulation server host running at real time. When simulating wired networks, there is no discrete event exchanged between two SVNs. When simulating wireless networks, there is no IP protocol discrete event exchanged between any two SVNs, and only environmental discrete events (node mobility and wireless signal propagation events) are exchanged between two neighboring SVNs.

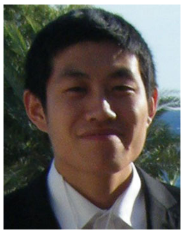
We present various case studies to show RVIP's advantages to implement the new Turing-indistinguishable mode over other existing solutions. The topology tests show that QualNet-based EXata cannot simulate many scenarios, for example, those with the real world Internet hot swappably plugged into the scenario at real time. The performance experiments are carried out to evaluate the overhead incurred by RVIP on latency and throughput compared with an equivalent LN. The result shows that RVIP's performance depends on the SVN configuration and the computational resources of the underlying simulation platform. Before the computational power is exhausted at a saturation point, RVIP's overhead on throughput is insignificant, and the extra delay is only about 0.5 ms. We draw the conclusion that RVIP bridges the gap between LN and SVN with little impact on the simulation accuracy and achieves better Turing indistinguishability than other existing solutions.

REFERENCES

1. Vahdat A, Yocum K, Walsh K, Mahadevan P, Kostic D, Chase J, Becker D. Scalability and accuracy in a large-scale network emulator. *ACM SIGOPS Operating Systems Review* 2002; **36**(SI): 271–284.
2. Mahrenholz D, Ivanov S. Real-time network emulation with NS-2. In *The 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT)*, Toulouse, France, 2004; 29–36.
3. Zhou J, Ji Z, Varshney M, Xu Z, Yang Y, Marina M, Bagrodia R. WHYNET: a hybrid testbed for large-scale, heterogeneous and adaptive wireless networks. In *Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, Los Angeles, California, USA, ACM, 2006; 111–112.
4. Zhou J, Ji Z, Bagrodia R. TWINE: a hybrid emulation testbed for wireless networks and applications. In *25th IEEE International Conference on Computer Communications*, Barcelona, Spain, 2006; 1–13.
5. Inc O. System-in-the-Loop (SITL) module. http:// www.opnet.com/solutions/network_rd/system_in_the_loop.htmll [accessed on May 2012].
6. Fujimoto R, Perumalla K, Park A, Wu H, Ammar M, Riley G. Large-scale network simulation: how big? how fast?. In *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, Orlando, Florida, USA, 2003; 116–123.
7. LXC: Linux Containers. http://lxc.sourceforge.net/ [accessed on March 2014].
8. Fall K. Network emulation in the vint/ns simulator. In *IEEE International Symposium on Computers and Communications*, Sharm El Sheikh, Red Sea, Egypt, 1999; 244–250.
9. Zhang L. VirtualClock: a new traffic control algorithm for packet switching networks. In *ACM Symposium on Communications Architectures and Protocols*, Philadelphia, Pennsylvania, USA, 1990; 19–29.
10. Scalable Network Technologies, Inc. Qualnet. http://www.qualnet.com/ [accessed on March 2014].
11. Kristiansen S, Plagemann T. Accuracy and scalability of NS-2's distributed emulation extension. *Simulation* 2011; **87**(1-2): 45–65.
12. Alvarez A, Orea R, Cabrero S, Pañeda XG, Garcia R, Melendi D. Limitations of network emulation with single-machine and distributed NS-3. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, Torremolinos, Malaga, Spain, ICST, 2010; 67.
13. Staub T, Gantenbein R, Braun T. VirtualMesh: an emulation framework for wireless mesh networks in OMNeT++. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Denver, Colorado, USA, 2009; 64.
14. Weingärtner E, vom Lehn H, Wehrle K. Device-driver enabled wireless network emulation. In *the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Paphos, Cyprus Island, 2011; 171–178.
15. Weingärtner E, Schmidt F, Heer T, Wehrle K. Synchronized network emulation: matching prototypes with complex simulations. *ACM SIGMETRICS Performance Evaluation Review* 2008; **36**(2): 58–63.
16. Liu J. Immersive real-time large-scale network simulation: a research summary. *IEEE IPDPS* 2008: 1–5.
17. Liu J, Li Y, Vorst N, *et al.* A real-time network simulation infrastructure based on openvpn. *Journal of Systems and Software* 2009; **82**(3): 473–485.
18. Scalable Network Technologies, Inc. QualNet 4.5 Network Emulation Interface Model Library. http://www.qualnet.com/.
19. Schiffman M, 2005. The libnet packet construction library.
20. McCanne S, Leres C, Jacobson V, 1989. Libpcap.
21. Liu J, Li Y, He Y. A large-scale real-time network simulation study using prime. In *Winter Simulation Conference*, Austin, Texas, USA, IEEE, 2009; 797–806.
22. Quagga Routing Software Suite. http://www.nongnu.org/quagga/.
23. Traffic Generator. http://www.postel.org/tg/.
24. Li T, Kong J, Li P, Gong P. RVIP: bridging live networks and software virtual networks for large scale network simulation at real time. In *the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*, Paphos, Cyprus Island, 2012; 171–178.

AUTHORS' BIOGRAPHIES

Jiejun Kong (jkong@turingnetwork.servehttp.com) was a senior researcher in Scalable Network Technologies, Inc. and a postdoctoral researcher in the Network Research Lab of Computer Science Department at University of California, Los Angeles. He is interested in developing efficient, scalable, and secure network protocols for wireless networks. His research topics include secure and anonymous routing, authentication, access control, distributed data harvesting, and network security modeling in mobile wireless networks, in particular, those with challenging network constraints and high security demands, such as mobile ad hoc networks and underwater sensor networks. He has contributed to the design, implementation, and testing of network protocols within the NSF iMASH, ONR MINUTEMAN/STTR, NSF WHYNET, and all QualNet/EXata commercial software development projects. He is now founding Turing Network Test L.L.C. to continue his computer network research and development.



Tingzhen Li (tingzhen.li@gmail.com) was a graduate student in Beijing Institute of Technology. He is interested in network simulation and emulation research. He has published several network related papers in IEEE/ACM conferences and journals, also served in IEEE SCORED 2012 Technical Program Committee.

Dapeng Oliver Wu (S'98-M'04-SM'06-F'13) received BE in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 1990; ME in Electrical Engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1997; and PhD in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2003. He is a Professor at the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, and information and network security. He received the University of Florida Research Foundation Professorship Award in 2009, AFOSR Young Investigator Program (YIP) Award in 2009, ONR YIP Award in 2008, NSF CAREER award in 2007, the IEEE Circuits and Systems for Video Technology (CSVT) Transactions Best Paper Award for Year 2001, and the Best Paper Awards in IEEE GLOBECOM 2011 and International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine) 2006. Currently, he serves as an Associate Editor for IEEE Transactions on Circuits and Systems for Video Technology, Journal of Visual Communication and Image Representation, and International Journal of Ad Hoc and Ubiquitous Computing. He is the founder of IEEE Transactions on Network Science and Engineering. He was the founding Editor-in-Chief of Journal of Advances in Multimedia between 2006 and 2008, and an Associate Editor for IEEE Transactions on Wireless Communications and IEEE Transactions on Vehicular Technology between 2004 and 2007. He is also a guest editor for the IEEE Journal on Selected Areas in Communications, Special Issue on Cross-layer Optimized Wireless Multimedia Communications. He has served as Technical Program Committee (TPC) Chair for IEEE INFOCOM 2012 and TPC chair for IEEE International Conference on Communications (ICC 2008), Signal Processing for Communications Symposium, and as a member of executive committee and/or technical program committee of over 80 conferences. He has served as Chair for the Award Committee and Chair of Mobile and wireless multimedia Interest Group (MobIG), Technical Committee on Multimedia Communications, IEEE Communications Society. He was a member of Multimedia Signal Processing Technical Committee, IEEE Signal Processing Society from 1 January 2009 to 31 December 2012. He is an IEEE Fellow.