

Toward Scalable Routing Experiments with Real-Time Network Simulation *

Yue Li, Jason Liu, and Raju Rangaswami
School of Computing and Information Sciences
Florida International University
Miami, Florida 33199
Emails: {yueli, liux, rajur}@cis.fiu.edu

Abstract

The ability to conduct accurate and realistic experiments is critical in furthering the research and development of network routing protocols. Existing framework for routing experiments is found to be lacking in one or more of the three required features: realism, scalability, and flexibility. We develop a new software infrastructure that combines the scalability and flexibility benefits of real-time network simulation with the realism of open-source routing protocol implementations. The infrastructure seamlessly integrates the open-source XORP router software with a previously developed real-time network simulation engine. Our design of the infrastructure uses a novel forwarding plane offloading approach that decouples routing from forwarding and confines the more resource consuming forwarding operations inside the simulation engine to reduce I/O overhead. Experiments demonstrate superior performance of the experimental infrastructure without impairing accuracy.

1 Introduction

Researchers are continuously searching for techniques that can better evaluate the dynamic behavior of routing protocols in large-scale complex network scenarios. Such techniques can improve our understanding of complex problems, such as the stability issues of BGP and OSPF [3, 29–31], BGP convergence [24], BGP security and misconfiguration [8, 12, 22, 23]. Existing techniques for understanding and evaluating network routing protocols can be categorized as *analytical*, *simulation-based*, and *emulation-based*. Here we exclude direct experimentation on physical networks, because it is almost infeasible to conduct custom large-scale routing experiments directly on routers deployed deep in the Internet.

Analytical techniques, which can often bring valuable insight to the design of large-scale complex systems, are unlikely to address the complexities and dynamics inherent with the actual protocol execution. Simulation is used commonly by researchers to perform large-scale routing experiments [4, 10, 20]. While simulation can capture the network dynamics with a great amount of detail, a simulation model can be quite different from a real routing protocol implementation. Consequently, they do not provide the necessary *realism* of protocol execution behavior. Further, simulation implementation of routing protocols typically comes with only a modest selection of protocols and protocol versions, thus limiting its utility. Without an active involvement from the protocol research and development community, simulation implementation of routing protocols tends to have only a short lifespan.

Emulation-based techniques allow the execution of network routing protocols “as is” within an emulated network environment. Emulation allows the necessary *realism* in the protocol execution. With the emergence of common open-source router software, such as XORP [34], Zebra [35], and Quagga [27], researchers can now prototype and evaluate routing protocols on actual systems as part of the emulation testbed. Emulation-based techniques, however, must address the issues of *scalability* and *flexibility*. The scalability of an emulation testbed is constrained by its available resources and physical setup. For example, experimental network scenarios must observe the limits on bandwidth and latency of the underlying network infrastructure. Emulation-based solutions have difficulties in creating arbitrary network topologies and controlling background network traffic conditions.

Our position is that the design of a successful framework for routing experiments must comprehensively address the three challenges of *realism*, *scalability*, and *flexibility*. In this paper, we describe an infrastructure that we built to support scalable routing experiments based on a real-time network simulation system. A key advantage of the real-time network simulation approach is that, since it provides a real-

*This research is supported in part by National Science Foundation grants CNS-0546712 and HRD-0317692.

time communication interface to real client machines, real applications can be executed unmodified within this framework [17]. Consequently, if engineered correctly, such a simulation framework can provide a realistic network infrastructure indistinguishable from a real network from the standpoint of the real applications. When this framework is coupled with real clients running open-source router software, it can provide a *realistic, scalable, and flexible* environment for conducting routing experiments.

Our infrastructure uses a previously developed real-time network simulator, called PRIME [15]. We address the three challenges comprehensively as we describe below:

- **Realism.** The infrastructure that we built uses XORP, a widely-used open-source routing software that provides a natural environment for experimenting with existing as well as new protocols [34]. XORP implements a number of routing protocols, including BGP, OSPF, RIP, PIM-SM, IGMP and MLD. There is a one-to-one relationship between the simulated routers within the real-time simulation mechanism (that simulates a virtual network) and the XORP instances. The routing messages exchanged between XORP instances are considered as foreground traffic and are forwarded to the real-time simulator as if they were carried on the virtual network. Since the routing function is performed by the actual routing software (XORP), the necessary realism in protocol behavior is preserved by the infrastructure.
- **Scalability.** Traffic on the virtual network must be forwarded based on routing decisions made by XORP instances. If each packet must be exported from the simulator and sent to an XORP router instance, and then immediately received and imported back into the simulator as soon as forwarding is achieved at the router, the I/O overhead would be substantial and might limit the scale of the routing experiments. We propose a forwarding plane offloading approach, which allows the XORP router to designate the packet forwarding function to the simulator and communicates with the simulator its routing decisions through an out-of-band channel. In doing so we can confine the background traffic (unrelated to routing) within simulation, thereby eliminating the I/O overhead associated with the bulk of the network traffic. Furthermore, since packet transmissions can be parallelized somewhat easily, we are able to conduct routing experiments with a network size far beyond what can be supported by emulation testbeds.
- **Flexibility.** It is relatively easy to apply real-time simulation to explore a wide spectrum of network scenarios and address what-if questions (e.g., by injecting

network failures) when we design routing experiments. Real-time simulation is based on simulation. Once a simulation model is developed, reasonably verified and validated, it takes little effort to conduct simulation experiments to cover a large parameter space. We can also incorporate analytical models in the real-time network simulation. For example, we can use low-resolution models to describe aggregate Internet traffic (e.g., [16]), which can significantly increase the size of the network we can include for routing experiments. Further, it is relatively easy to manipulate and control dynamic network behavior with simulation. For instance, it is quite simple to bring down a link in the virtual network and observe the reaction mechanism of the routing protocols.

2 Background

Our infrastructure for scalable routing experiments builds on top of a real-time parallel simulation engine called PRIME, which stands for a Parallel Real-time Immersive Modeling Environment [15]. PRIME enables real-time large-scale network simulations and supports seamless interactions with real distributed applications. PRIME is built on top of a parallel discrete-event simulation kernel that implements the Scalable Simulation Framework (SSF) [9]. The parallel simulation kernel deals with synchronization and communication issues between logical processes that execute simulation events in parallel, while PRIME itself enables the notion of real time by incorporating packets from external application as real-time events. In order to meet the deadline requirement of the real-time events, PRIME employs a greedy priority-based scheduling algorithm—real-time events are assigned with a higher priority to ensure their timely processing.

PRIME can interact with a large number of clients, which may be traffic sources, sinks, or other network entities like routers. It is important to be able to incorporate a large number of such clients and yet remain transparent to the applications running on the clients. There are several ways to incorporate real applications as part of the simulation environment, which include using packet capturing techniques (such as libpcap, IP table, and IP tunnel), preloading dynamic libraries, and modifying the binary executables, and so forth. PRIME instead uses an open systems approach based on a Virtual Private Network (VPN) solution [18]. We customize VPN to function as a gateway that bridges traffic between the clients and the simulated network. Clients establish connection to the simulation gateway. Traffic generated by the clients and destined for the virtual network is directed by the modified VPN through the gateway to the real-time network simulator.

We illustrate this approach using an example. Suppose

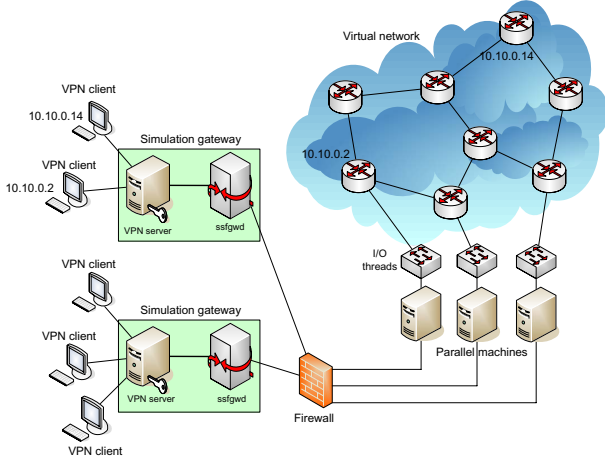


Figure 1. The emulation infrastructure based on VPN.

two clients with IP addresses configured as 10.0.0.14 and 10.0.1.2, respectively, need to communicate with each other (see Figure 1). Packets sent from 10.0.0.14 to 10.0.1.2 are first sent to a modified VPN server at the simulation gateway, which in turn forwards the packets to the real-time simulator via a dedicated TCP connection. At the simulator, the packets are injected into the simulation event list; the simulator simulates the packets being forwarded on the virtual network as if they were created by the virtual node with the same IP address 10.0.0.14. Upon reaching the virtual node 10.0.1.2, the packets will be exported from simulation engine, traveling in the reverse direction via the simulation gateway back to the client with the IP address 10.0.1.2.

A distinct advantage of this approach is that the emulation infrastructure does not require special hardware to set up. It is also secure, a merit inherited directly from the underlying VPN implementation, and scalable, since clients need not be co-located within the simulation infrastructure. Further, multiple simulation gateways can be used simultaneously to accommodate larger traffic capacity between the clients and the real-time network simulator. In order to produce accurate results, however, the emulation infrastructure needs a tight coupling between the emulated entities (i.e., the clients) and the real-time simulator. In particular, the segment between the clients and the real-time network simulator needs to include low-latency links. To maintain potentially high traffic throughput demand, this segment must also have sufficient bandwidth for tunneling traffic through the emulation infrastructure.

In the next section, we develop techniques that allow the above real-time simulation infrastructure to scale up when used specifically for network routing experiments.

3 An Infrastructure for Scalable Routing Experiments

The availability of open-source router platforms, such as XORP, Zebra and Quagga, has simplified the task of researchers, who can now prototype and evaluate routing protocols with relative ease. To support experiments on a large-scale network consisting of many routers with multiple traffic sources and sinks, we propose to integrate the open-source router platforms with a real-time network simulation infrastructure.

3.1 Forwarding Plane Offloading

As described in Section 2, PRIME provides real-time simulation capabilities and an emulation infrastructure that can seamlessly integrate multiple clients, may they be routers or otherwise. Since the routers must be emulated outside PRIME on client machines where they can run the real routing software directly, every packet traveling along its path from the source to the destination must be exported to each intermediate router for forwarding decisions, and subsequently imported back into PRIME. Thus, the forwarding operation for each packet at each hop incurs a non-trivial I/O overhead. Consequently, the overall overhead will significantly impact the performance of the infrastructure, especially in the case of large-scale routing experiments. To avoid this problem, we propose a forwarding plane offloading approach, which we move the packet forwarding functions from the router software to the simulation engine. Since packet forwarding operations are carried out entirely with PRIME, we can eliminate the I/O overhead associated with sending packets back and forth between the router software and the real-time simulator.

This approach, however conceptually straightforward, requires two key design issues be addressed: First, the forwarding table for a specific *virtual* router used by the simulator for internal forwarding decisions must be synchronized with the actual forwarding table, which is updated by the routing protocols running within the corresponding external router instance. The synchronization is critical to ensure correct forwarding operations within the simulation engine. Second, the simulation engine must be instantly informed of any network interface reconfigurations performed at the external router instance. For instance, a network administrator may create virtual network interfaces or reset the MTU of an existing network interface; in such cases, the corresponding virtual router inside simulation must be kept consistent with the changes.

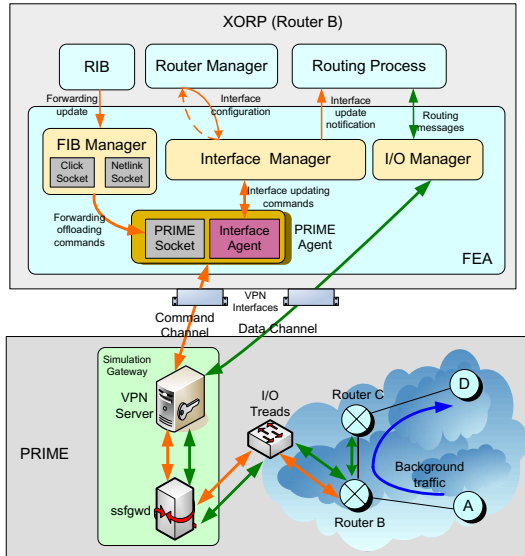


Figure 2. Offloading the forwarding plane to the PRIME simulation environment.

3.2 The XORP Architecture

In our current implementation, we combine the open-source XORP router software with PRIME to provide a scalable routing experiment platform. Two reasons prompted our choice of XORP. First, XORP has the most up-to-date implementations of routing protocols, including those commonly used protocols, such as BGP, OSPF, and RIP. XORP operates very much like real router software, and is a well-established experimental platform for routing protocol design and evaluation by the routing protocol research community. Second, XORP implements a Forwarding Engine Abstract (FEA) layer, which splits the routing and forwarding plane as two separate layers within its implementation, thus considerably simplifying our task of offloading the latter to PRIME.

The primary role of FEA is to shield XORP processes from concerns of forwarding plane operations, so that arbitrary forwarding mechanisms can be employed via plug-ins. FEA has three main components: Forwarding Information Base (FIB) manager, interface manager, and I/O manager. These components are depicted in Figure 2, along with the corresponding XORP processes that interact with FEA. The FIB manager receives forwarding table updates from the routing information base (RIB) process, which arbitrates route updates among routing protocols (e.g., BGP or OSPF). The FIB manager propagates the necessary changes (such as inserting or deleting routing entries) to the underlying forwarding engine. Currently, XORP can choose either the traditional Unix kernel (via Netlink) or the Click mod-

ular software router [14] for forwarding. Access to the forwarding plane is performed by the corresponding plug-ins.

The router manager process can also issue interface configuration requests to the FEA interface manager, according to a pre-specified router configuration file or real-time input from the system administrator via a command-line shell. FEA interprets and executes these requests on behalf of the underlying forwarding engine. Individual routing processes can register with FEA to be notified of changes in the interface configuration. The I/O manager also provides an interface for all the routing protocols to communicate the routing information (such as OSPF’s Link State Update messages) with their peers on other router instances.

3.3 XORP Forwarding Plane Offloading Implementation

Figure 2 describes our architecture for forwarding plane offloading with XORP. We create an additional XORP forwarding plane plug-in within FEA, which we call the PRIME Agent. It consists of two components: the PRIME Socket and the Interface Agent. Both components maintain a common *command channel* with the PRIME simulator for transferring forwarding information updates (via the PRIME Socket) and interface configuration requests (via the Interface Agent) to the corresponding virtual router in PRIME. This effectively results in the forwarding functionality being offloaded from XORP to PRIME; traffic on the simulated network can thus be routed within simulation.

The interaction between the PRIME Agent and the simulation mechanism is managed as follows. The PRIME Socket provides primitives to manage the forwarding table in PRIME, including those that add or remove a single forwarding entry as well as the entire forwarding table. The PRIME Socket accepts a command from the FIB manager, packages the command into a control packet, and sends it through the command channel. We designate an unused address as the destination address of the control packets, so that they can be distinguished from regular data packets when reaching the simulator through the emulation infrastructure. At the PRIME-end of the command channel, the control packets are translated into simulation events and presented to the emulation protocol session at the corresponding virtual router. The emulation protocol session interprets the control commands and executes them by modifying the forwarding table on the virtual router as directed by XORP. The procedure of interface management is essentially the same. The interface manager can modify the network interface configuration at the corresponding virtual router through primitives provided by the Interface Agent of the FEA plug-in.

Since forwarding table updates and interface management requests are propagated immediately to the simulation

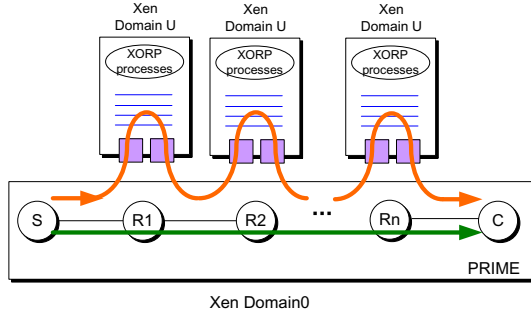


Figure 3. Model topology used for microbenchmark experiments.

engine, the forwarding decisions inside simulation are synchronized with the XORP instances. The routing messages created by the XORP routing protocol instances are treated as foreground traffic from the perspective of PRIME and they are forwarded through the simulation engine with the aid of the emulation infrastructure as described in Section 2. This mechanism is enabled by independent *data channels* on top of VPN automatically.

4 Experiments

We conducted two sets of experiments to evaluate our infrastructure for both performance and accuracy. The initial set of experiments (Section 4.1) are controlled microbenchmarks that allow fine-grained evaluation of the proposed infrastructure, both with and without forwarding plane offloading to the simulator. Next, we use an intra-domain routing experiment of a real network topology (Section 4.2) to demonstrate the capability of our infrastructure.

4.1 Microbenchmark

The purpose of the microbenchmarks is to illustrate that PRIME can support scalable routing experiments with the forwarding plane offloading technique. Since PRIME needs to interact with the XORP instances and simulate the virtual network in real time, the timing accuracy is paramount in obtaining trustworthy experimental results. This means that the infrastructure (both PRIME and the external XORP router instances) should keep up with the wall-clock time, receive real-time events promptly and output them without missing their real-time deadlines.

We use an extensible network model for microbenchmarking, as shown in Figure 3. Node *S* and *C* act as the server and client, respectively; the server initiates constant-bit-rate UDP traffic to the client. The intermediate nodes, R_1, R_2, \dots, R_n , are routers. Each virtual router has one

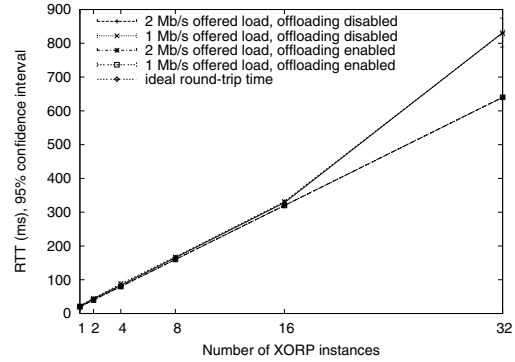


Figure 4. Impact on round-trip delays.

corresponding XORP instance running outside PRIME. We configured each XORP instance to run a OSPFv2 protocol.

In order to scale up the microbenchmark in an easy way we use Xen virtual machines [2] to host PRIME and XORP instances. Xen is considered as a high-performance virtual machine monitor (VMM). In Xen terminology, each domain is a virtual machine running on the host machine. Domain 0 is the first domain created when the host is powered on and has privileges to access physical machine resources. It also provides an interface to manage other domains (called Domain U's) run unprivileged.

We set up the Xen environment on a DELL OPTIPLEX 745 workstation with Intel Core 2 Duo 2.4 GHz processors and 4 GB memory. Both the host OS and guest OS run Debian Linux 4.0 distributions with kernel version 2.6.16.33. PRIME and the simulation gateway run in Xen Domain 0 and each Domain U contains one XORP instance. We wrote a set of scripts to support rapid deployment of an arbitrary network experiment scenario.

To evaluate performance of the simulation infrastructure and its impact on timing fidelity, we use round-trip delay and throughput as the primary metrics for evaluation. We evaluated two settings of the simulation infrastructure implementation: *offloading-disabled* (in which the forwarding plane offloading optimization is not used) versus *offloading-enabled* (in which the optimization is employed).

Figure 4 depicts the round-trip delay measured by ping between the client and the server as we increase the number of intermediate router hosts running XORP. The propagation delay of individual links were set to be 10 ms each. Consequently, the ideal round trip time would be $2 \times 10 \times n$, where n is the number of intermediate routers. We vary the UDP background traffic between the server and the client to be either 1 Mb/s or 2 Mb/s. If we disable forwarding plane offloading, the observed round-trip times deviates from the ideal case, especially when the number of intermediate XORP instances is large. The two curves representing the cases with offloading disabled overlap. The observed

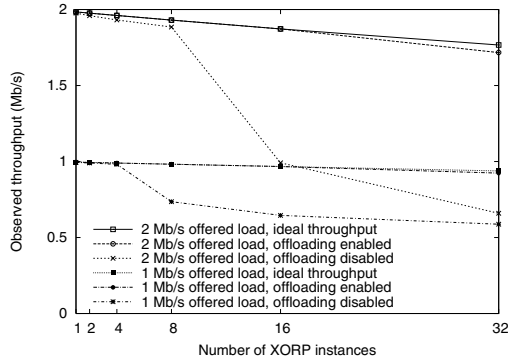


Figure 5. Impact on throughput.

values deviate by as much as 30% with 32 XORP instances. With 2 Mb/s offered load, the variation of the round-trip time is slightly larger. Note that the link bandwidths were set to 100 Mb/s, which is substantially larger than the offered load, and consequently should not noticeably affect the round-trip times in the ideal case. The observed values when offloading is enabled match almost exactly with the ideal values, with less than 0.2% error, regardless of the offered load.

Figure 5 depicts the effect of the simulation infrastructure overhead on traffic throughput between the client and server nodes, again as we vary the number of intermediate XORP router instances. The throughput is calculated by the total size of data transferred from the server to the client divided by the time since the download request is made by the client. The throughput decreases slightly (even in the ideal case) with more router instances and therefore longer path between the server and the client. With offloading disabled, the observed throughput is lower than that of the ideal case, becoming more apparent with more intermediate XORP instances. With offloading enabled, the observed throughput deviates only slightly (less than 3% with 32 XORP instances). As expected, since the data download traffic is routed within simulation, rather than being forwarded through the emulation infrastructure, we achieved much better accuracy.

4.2 Case Study

In this section we describe an intra-domain routing experiment as a case study to demonstrate that a realistic routing experiment can be achieved with the integration of PRIME and XORP. The experiment was originally used by VINI [5] to demonstrate its capability to conduct elaborate routing tests on the PlanetLab. The experiment consists of a realistic Abilene network model with eleven routers, as shown in Figure 6. We configured the model using the real topology, with link delays and bandwidths obtained from



Figure 6. The Abilene network.

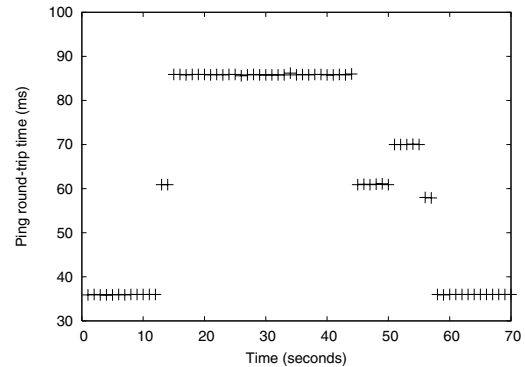


Figure 7. OSPF route convergence indicated by ping RTT.

the observatory data available at Abilene Observatory Web Site [1]. In particular, our model reflects a snapshot of the Abilene backbone on Jan 14 2008 at 07:45:11 UTC. We also configured each XORP instance to run OSPFv2 protocol. We used the same Xen virtual machine environment as described in Section 4.1 for the experiment setup.

The purpose of the experiment is to observe the convergence of OSPF and its effect on data traffic. We injected a link failure, followed by a recovery between the routers at Chicago and Indianapolis (via the XORP command shell). We measured the effect on the round-trip time and data throughput between Denver and Chicago. Figure 7 shows the round-trip time measured by ping. When the link went down at 14 seconds, OSPF instances recomputed the routes resulting a diversion of the traffic from the upper links to the lower ones. The round-trip delays were increased from around 36 ms to around 60 ms. The link was later repaired at 43 seconds into the experiment. After the routes converged, OSPF re-established the shortest path between the two cities.

We also used `iperf` to initiate a TCP transfer from Denver to Chicago in a separate run. We analyzed the

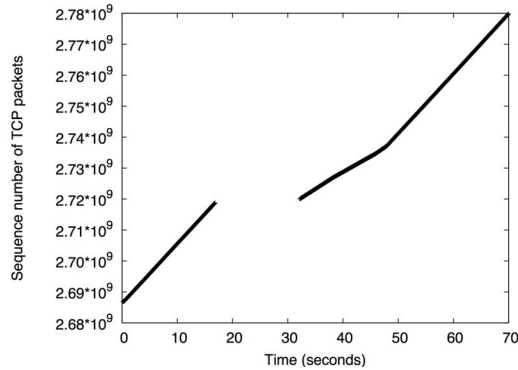


Figure 8. TCP transfer during link failure and recovery.

`tcpdump` results. Figure 8 shows the number of bytes received at the receiver end (i.e., the router at Chicago). The TCP transfer was interrupted when link failure occurred at 18 seconds and then resumed when an alternative route was found (at 32 seconds). At 50 seconds, the link was repaired, the shortest path was re-established, and a better throughput was achieved (demonstrated by a steeper slope in the figure).

5 Related Work

Most current real-time network simulators are based on existing network simulators added with the capabilities of interacting with real applications. Examples include NSE [11], IP-TNE [6], MaSSF [21], and Maya [36]. NSE is an emulation extension of the popular *ns-2* simulator [7] augmented with the support for connecting with real applications and scheduling real-time events. IP-TNE is the first simulator we know that adopts parallel simulation techniques for large-scale network emulations. MaSSF is based on the DaSSF simulator [19] with support for the grid computing environment. Maya is an emulation extension of the commercial QualNet simulator [28] for wireless mobile networks. ROSENET [13] is an emulation framework that separates network simulation that provides high-fidelity network resource estimates from low-fidelity emulation traffic forwarding. To some extent, this is similar in concept to our approach of separating routing and forwarding functions.

Other alternatives for conducting experimental network research are emulation and physical testbeds. EmuLab [33] is a notable emulation tool that consists of a large number of networked computers integrated and coordinated to present a virtual network environment. EmuLab currently supports hundreds of projects with thousands of users running over 18,000 experiments per year. PlanetLab [26] is a collection

of machines distributed across the Internet and used by researchers to develop new network services. These machines run a customized operating system that allows the computing and networking resources to be shared among multiple experimentations that may take place simultaneously. A critical component of this type of physical testbeds is the live traffic.

Our work bears resemblance with VINI [5]. VINI is a virtual network infrastructure that enables realistic and controlled network experiment over PlanetLab. By using virtualization technology, researchers are able to create a flexible virtual network topology. The implementation of VINI runs XORP instances within UML virtual machines and Click for packet forwarding. We drew inspiration from the VINI design. Our scheme is different, however, in that we focus on enabling real network simulation to achieve realism, scalability, and flexibility. Our approach allows a large number of XORP instances beyond what can be supported by PlanetLab.

6 Conclusions and Future Work

A sound infrastructure allowing scalable routing experiments is a key enabler for the design and development of network routing protocols. We propose a novel infrastructure that combines a real-time network simulation engine with an emulated implementation of open-source XORP router software to provide a realistic, scalable, and flexible testbed for routing experiments. Offloading the forwarding plane from the XORP instances to the simulator allows scalability, by confining bulk traffic to be forwarded more efficiently within the simulator without exposing it to the external routers. An experimental evaluation of this infrastructure, with microbenchmark tests and a case study, demonstrates the effectiveness of our approach in minimizing overhead and consequently improving accuracy and scalability. We are currently testing PRIME to interact with hundreds of XORP instances running on a single physical machine and we will report the results in another paper.

Our current infrastructure for scalable routing experiments can be improved along two directions. First, the infrastructure currently requires one operating system image per XORP instance (running either on a physical or virtual machine). This situation can be improved with lightweight network virtualization solutions, such as VRF [32] and OpenVZ [25], which provide the abstractions necessary to run multiple XORP instances within a single OS instance to achieve additional scalability. Second, the synchronization updates between the XORP protocol instances and their corresponding virtual router instances within PRIME are only one-directional in our current prototype implementation; updates are only propagated from the emulated XORP routers to the simulation engine. We will allow information

exchange in both directions so that we will be able to deal with more complex requirements in protocol operation.

References

- [1] Abilene Observatory Summary Data View. <http://abilene.internet2.edu/observatory/data-views.html>.
- [2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, 2003.
- [3] A. Basu and J. G. Riecke. Stability issues in OSPF routing. In *SIGCOMM 2001*.
- [4] D. Bauer, M. Yuksel, C. Carothers, and S. Kalyanaraman. A case study in understanding OSPF and BGP interactions using efficient experiment design. In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)*, 2006.
- [5] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: Realistic and controlled network experimentation. In *SIGCOMM 2006*.
- [6] R. Bradford, R. Simmonds, and B. Unger. A parallel discrete event IP network emulator. In *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'00)*, pages 315–322, August 2000.
- [7] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.
- [8] T. W. Chim and K. L. Yeung. Time-efficient algorithms for BGP route configuration. In *Proceedings of the IEEE International Conference on Communications (ICC'04)*, 2004.
- [9] J. Cowie, D. Nicol, and A. Ogielski. Modeling the global Internet. *Computing in Science and Engineering*, 1(1):42–50, January 1999.
- [10] X. A. Dimitropoulos and G. F. Riley. Large-scale simulation models of BGP. In *Proceedings of IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'04)*, 2004.
- [11] K. Fall. Network emulation in the vint/ns simulator. In *Proceedings of the 4th IEEE Symposium on Computers and Communications (ISCC'99)*, pages 244–250, July 1999.
- [12] G. Goodell, W. Aiello, T. G. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Proceedings of NDSS (NDSS'03)*, 2003.
- [13] Y. Gu and R. Fujimoto. Applying parallel and distributed simulation to remote network emulation. In *Proceedings of the 2007 Winter Simulation Conference*, 2007.
- [14] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(8):263–297, August 2000.
- [15] J. Liu. The PRIME research. <http://www.cis.fiu.edu/prime/>.
- [16] J. Liu. Packet-level integration of fluid TCP models in real-time network simulation. In *Proceedings of the 2006 Winter Simulation Conference (WSC'06)*, pages 2162–2169, December 2006.
- [17] J. Liu. A primer for real-time simulation of large-scale networks. In *Proceedings of the 41st Annual Simulation Symposium (ANSS'08)*, April 2008. To appear.
- [18] J. Liu, S. Mann, N. V. Vorst, and K. Hellman. An open and scalable emulation infrastructure for large-scale real-time network simulations. In *INFOCOM 2007*.
- [19] J. Liu and D. M. Nicol. Dartmouth Scalable Simulation Framework (DaSSF). <http://www.cis.fiu.edu/~liux/research/projects/dassf/index.html>.
- [20] X. Liu and A. A. Chien. Realistic large-scale online network simulation. In *Proceedings of the SuperComputing Conference (SC'04)*, 2004.
- [21] X. Liu, H. Xia, and A. A. Chien. Network emulation tools for modeling grid behavior. In *Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'03)*, May 2003.
- [22] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *SIGCOMM 2002*.
- [23] O. Nordstrom and C. Dovrolis. Beware of BGP attacks. *ACM Computer Communications Review*, 34(2):1–8, April 2004.
- [24] D. Obradovic. Real-time model and convergence time of BGP. In *INFOCOM 2002*.
- [25] OpenVZ. <http://openvz.org/>.
- [26] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. <http://www.planet-lab.org/>.
- [27] Quagga Routing Suite. <http://www.quagga.net/>.
- [28] Scalable Network Technologies. <http://scalablenetworks.com/>.
- [29] A. Shaikh, R. Dube, and A. Varma. Avoiding instability during graceful shutdown of OSPF. In *INFOCOM 2002*.
- [30] A. Shaikh and A. Greenberg. Experience in black-box OSPF measurement. In *Proceedings of the SIGCOMM Internet Measurement Workshop*, 2001.
- [31] A. Shaikh, L. Kalampoukas, R. Dube, and A. Varma. Routing stability in congested networks: Experimentation and analysis. In *SIGCOMM 2000*.
- [32] Linux Virtual Routing and Forwarding (VRF). <http://sourceforge.net/projects/linux-vrf/>.
- [33] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, pages 255–270, December 2002.
- [34] XORP. <http://www.xorp.org/>.
- [35] GNU Zebra. <http://www.zebra.org/>.
- [36] J. Zhou, Z. Ji, M. Takai, and R. Bagrodia. Maya: integrating hybrid network modeling to the physical world. *ACM Transactions on Modeling and Computer Simulation*, 14(2):149–169, April 2004.