Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

A LARGE-SCALE REAL-TIME NETWORK SIMULATION STUDY USING PRIME

Jason Liu
Yue Li
Ying He

School of Computing and Information Sciences
Florida International University
Miami, Florida 33199, USA

ABSTRACT

We examine the capabilities of conducting network experiments involving a large-scale peer-to-peer web-content distribution network. Our study uses a real-time network simulator, called PRIME, running on EmuLab, which is a shared cluster computing environment designed specifically for network emulation studies. Our study is one of the largest network experiments that involve a real implementation of a peer-to-peer content distribution system under HTTP traffic from a public-domain empirical workload trace and using a realistic large network model. Our experiments demonstrate the potentials of real-time simulation for studying complex behaviors of distributed applications under large-scale network conditions.

1 INTRODUCTION

1.1 Parallel Real-Time Simulation

PRIME (2009), which stands for Parallel Real-time Immersive network Modeling Environment, is a high-fidelity parallel network simulator capable of running large-scale network models. The implementation of PRIME inherits most of our previous efforts in the development of Scalable Simulation Framework (SSF), a process-oriented and conservatively synchronized parallel simulation engine designed for multi-protocol communication networks. SSF can run on most platforms, including shared-memory multiprocessors and clusters of distributed-memory machines. The SSF simulation engine is ultra fast (Liu et al. 1999) and has been demonstrated to be capable of handling large network models, including simulations of the Internet (Cowie et al. 1999, Nicol et al. 2003, Nicol et al. 2003), cellular systems (Delve and Smith 2001), wireless ad hoc networks (Liu et al. 2005, Newport et al. 2007), and wireless sensor networks (Liu et al. 2001).

In order to support large-scale simulation, PRIME relies on advanced parallel simulation techniques to harness the collective computing resources of parallel computers for an increased event-processing power. For example, to achieve good performance on distributed-memory machines, PRIME adopts a hierarchical synchronization scheme to address the discrepancy in the communication cost between distributed-memory and shared-memory platforms (Liu and Nicol 2001). Further, PRIME implements the composite synchronization algorithm (Nicol and Liu 2002), which combines the traditional synchronous and asynchronous conservative parallel simulation algorithms. When used alone, the traditional approaches have been shown to be inefficient to deal with particular aspects of the model topology. The composite algorithm is able to efficiently simulate diverse network scenarios, including those that exhibit large variability in link types (particularly with the existence of low-latency connections), and in node types (especially for those with a large degree of connectivity).

PRIME extends SSF with emulation capabilities, where unmodified implementations of real applications can interact with the network simulator that operates in real time. That is, the simulation time in PRIME can be made to advance synchronously with the wall-clock time. Traffic originated from the real applications is captured by PRIME's emulation facilities and forwarded to the simulator; the real network packets are treated as simulation events as they are "carried" on the virtual network and experience appropriate packet delays and packet losses according to the run-time state of the simulated network. We call such simulation-based emulation approach *real-time simulation* (Liu 2008). Since the virtual network can interact seamlessly with the real network appliances, PRIME appears to be indistinguishable from a physical network in terms of conducting real traffic.

978-1-4244-5771-7/09/$26.00 ©2009 IEEE

Large-scale real-time network simulation requires simulation be able to characterize the behavior of a network, potentially with millions of network entities and with realistic traffic load—all in real time. On the one hand, we apply parallel and distributed discrete-event simulation techniques to speed up simulation of large-scale networks. On the other hand, we use models at different levels of abstraction to reduce the computational demand while maintaining a desired degree of accuracy (Liu 2006, Liu and Li 2008, Liu and Li 2009). For example, our previous studies (Liu and Li 2009) have shown that the parallel hybrid traffic model implemented in PRIME, which combines a fluid-based analytical model based on ordinary differential equations with the traditional packet-oriented discrete-event simulation, can achieve a speedup of more than three orders of magnitude over packet-oriented simulation without significant loss of accuracy.

PRIME's multi-resolution modeling capability allows network emulation at unprecedented scales. Through extensive studies, we have been able to successfully emulate many applications, including routing algorithms, transport protocols, content distribution services, web services, multimedia streaming, and peer-to-peer networks (e.g., Li et al. 2008, Li et al. 2009, Erazo et al. 2009).

1.2 PRIME on Virtual Machines

To support real-time interaction with a large number of network applications, PRIME currently features a scalable and flexible emulation infrastructure based on a customized Virtual Private Network (VPN) framework (Liu et al. 2009). The VPN server is augmented to function as a gateway between the real-time network simulator and real applications running on machines that are potentially distributed geographically. To set up the emulation infrastructure, machines that run real applications (which we call "emulated hosts") need to first establish connection to the gateways as VPN clients. The VPN clients then create logical network interfaces (e.g., TUN/TAP devices), assign IP addresses, and modify kernel forwarding tables according to the emulation setup, so that real applications running on these emulated hosts can direct packets to the real-time simulator through the logical network interfaces. Most of this process is carried out automatically. Once the emulation infrastructure is set up, packets generated from the emulated hosts are forwarded to the real-time simulator and subsequently simulated by the real-time simulator as if they traversed on a real network. Packets arriving at simulated hosts designated for emulation are exported by the real-time simulator and sent to the corresponding emulation host via VPN.

A major advantage of the VPN emulation infrastructure is that it does not require special hardware arrangement. As such, PRIME allows flexible configuration of the emulation platforms supporting real-time simulation. VPN provides necessary security features, which are important for connecting machines that may be geographically distributed and belong to different administrative domains. The solution is also scalable since multiple VPN servers can be used to handle large traffic volume. VPN does, however, introduce noticeable overheads (Liu et al. 2009). In order to produce accurate results, the VPN emulation infrastructure requires a tight coupling (i.e., high-bandwidth low-latency connections) between the emulated hosts and the real-time simulator. Similar to other existing emulators, PRIME needs administrative privileges in order to establish the emulation infrastructure (such as setting up VPN on the client machines).

To allow large-scale real-time simulation, PRIME can run either on dedicated clusters or shared facilities, such as EmuLab (White et al. 2002) and PlanetLab (Peterson et al. 2002). Figure 1 shows an example of PRIME running in a cluster environment. To increase the number of emulated hosts, each compute node in the cluster may run several virtual machines. PRIME runs distributed simulation with multiple simulator instances, each running on a dedicated physical node or a virtual machine on a physical node (typically in a privileged domain to allow better efficiency). The simulator instances coordinate with other simulator instances using a parallel simulation synchronization algorithm. The emulated hosts are running within separate virtual machines, sometimes collocated with the simulator instances responsible for the part of the virtual network that contains the corresponding simulated hosts. These virtual machines are connected to the (real-time) simulator instances through the VPN emulation infrastructure.

For example, in Figure 1, packets generated by the emulated host VM_1 that corresponds to the simulated host H_1 on the virtual network are first intercepted by VPN and sent to the simulator instance SIM_0 running on VM_0. Once the packets are received, the simulator schedules simulation events to represent the packets being sent out from the corresponding simulated host H_1 on the virtual network. Suppose the packets are destined for H_3. The simulator simulates the packets being forwarded on the virtual network. When the packets are forwarded to a router simulated by the remote simulator instance, SIM_0 sends simulation events, in this case, to SIM_1. When the packets reach the simulated host R_1, SIM_1 exports the packets to the corresponding emulated host (in this case, VM_4). In this example, VM_4 runs a software router, which also conducts packet forwarding (e.g., see Li et al. 2008). As the packets are forwarded onward from VM_4, they are captured by the VPN emulation infrastructure and reinserted into the simulator to continue their simulated journey. Eventually, as the packets reach the simulated host H3, they are again exported to the corresponding emulated host VM_5. Note that this entire process is transparent to the applications running on VM_1 and VM_5; they are connected as if by a real network.

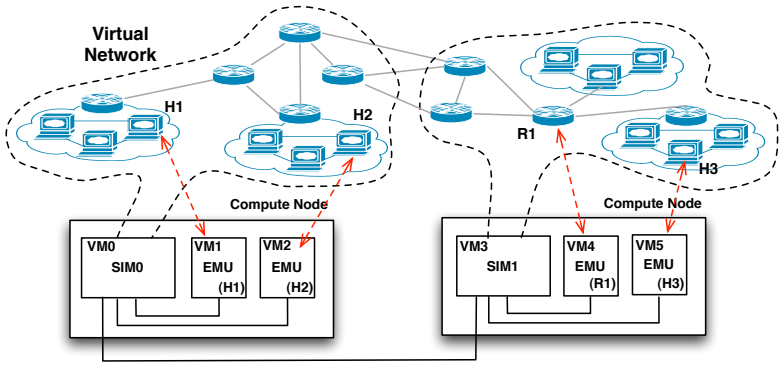

Figure 1: PRIME running on a VM cluster

1.3 Large-Scale Real-Time Simulation Studies

We conduct large-scale real-time simulations of a peer-to-peer content distribution network using PRIME on EmuLab machines (White et al. 2002). EmuLab is an experimentation facility consisting of many networked computers, integrated and coordinated to present an emulated network environment. In this study, we simply set up the EmuLab machines to run a customized OpenVZ kernel. OpenVZ (2009) is an open-source OS-level virtualization solution, which provides multiple instances (also called *virtual containers*) of the same (Linux) operating system on the same physical machine. OpenVZ has been demonstrated capable of providing superior performance and scalability compared to other virtualization approaches, such as Xen (Barham et al. 2003), VMWare Workstation (2009), and User-Mode Linux (Dike 2000).

Our experiment is designed to study the performance of web caching based on the content distribution network (CDN). We choose an open-source system, called CoralCDN (Freedman et al. 2004), for this study. We simulate a network of approximately 45,000 nodes. The network model is extended from a real tier-1 ISP topology resulting from a previous measurement study. To obtain realistic web traffic, we use a 24-hour segment of a publicly available HTTP trace collected at a web server; the trace consists of over 5 million HTTP requests from about 40,000 (anonymized) clients. We randomly map the clients onto the simulated network and direct the clients to replay the trace. We set up an Apache web server and use over 1,000 emulated hosts (i.e., OpenVZ containers) to run CoralCDN.

PRIME allows the *simulated* web clients to directly communicate with and download web objects from the CoralCDN nodes running directly on *emulated* hosts. The *emulated* CoralCDN nodes provide web caching functions using a peer-to-peer network over which data objects from the *emulated* web server are disseminated among the *simulated* clients. The implementation of TCP and HTTP in PRIME supports interoperability between simulated and emulated entities.

Our studies help us identify and demonstrate several important capabilities of PRIME:

(1) Immersion, realism and accuracy. Through real-time network simulation, PRIME allows real network protocols and applications to run unmodified on virtual machines connected through a virtual network. Since packet forwarding is carried out in simulation, one can incorporate detailed network models for an accurate representation of a large-scale network.

(2) Performance and scalability. PRIME targets existing commodity computing platforms with enough computation and communication power to accommodate common real-time large-scale network simulation experiments. The commodity computing platforms can be dedicated clusters or shared facilities, like EmuLab. PRIME is able to support emulation experiments with much larger networks than those supported by traditional emulation testbeds, in large part due to its capability of *simulating*, rather than really conducting the majority of the packet forwarding operations.

(3) Flexibility and controllability. The emulation capability supported by PRIME is model-driven. That is, at the core of PRIME is a simulated network, which is relatively easy to configure and largely independent from the underlying execution environment. Simulation is more effective than emulation at investigating design alternatives and conducting system-wide optimizations.

The rest of this paper is organized as follows. In section 2, we describe the process to derive the Internet backbone network model we use to conduct our large-scale real-time network simulation study. In section 3, we provide details of our hybrid approach to studying the web-content distribution network application in large-scale virtual network environments. The results of our experiments are discussed in section 4. We discuss related work in section 5, before we conclude this paper in section 6.

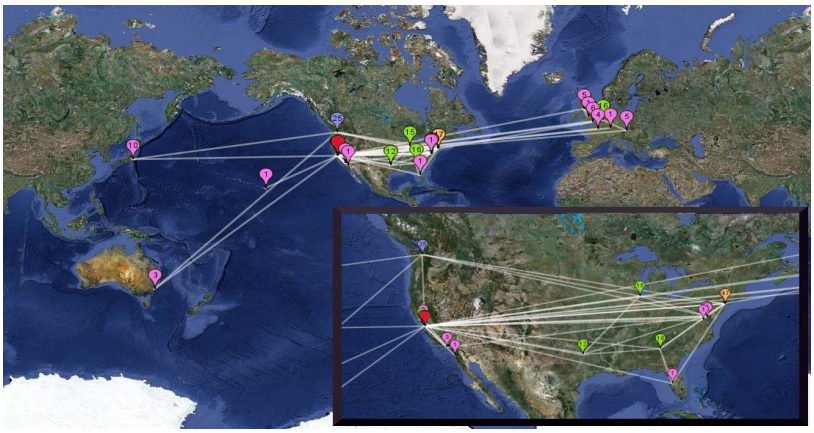

Figure 2: The backbone network

2 THE NETWORK MODEL

We extend the Rocketfuel dataset to build the network model for our study. Rocketfuel (Spring et al. 2004) contains the topology of 13 tier-1 ISPs, derived from information obtained from traceroute paths, BGP routing tables, and DNS. Previously, we created a best-effort Internet topology for large simulation studies using the Rocketfuel dataset (Liljenstam et al. 2003). Based on this study, we further process the Rocketfuel network topology to improve accuracy and reduce data noise.

To obtain a better assessment of the link delays, we first revisit the router locations in the Rocketfuel dataset. Most routers in Rocketfuel contain location information, and all the routers are associated with at least one IP address. For routers without location, we query the CAIDA NetGeo database (Moore et al. 2000), which provides a mapping from the router IP addresses to their locations (city, state, and country). For some routers, we still cannot resolve the locations. Since there are only a few of them, we simply remove these routers (altogether with the associated links) from the original dataset.

Once the locations are known, it is straightforward to find their GPS coordinates (latitudes and longitudes); the link delay can be calculated as the time it takes to traverse the distance between the two adjacent routers at the speed of light in fiber. Note that Mahajan et al. (2002) used a similar technique to infer the link delays for 6 of the 13 ISPs in Rocketfuel. Their study unfortunately does not cover all ISPs. To convert router locations to their GPS coordinates, we develop a script to send queries to the Astrodient Atlas Database (2009) and MIT Geographic Nameserver (2009). The reason we decide to simultaneously use two databases is due to the existing ambiguities in the Rocketfuel dataset. Some of the router locations are incorrect: they may contain misspelled names and sometimes are mistaken for some other locations. When discrepancies are observed in the separate responses from the two databases, we manually check the results using Google Maps (2009). In some cases, we also look up the GPS coordinates in Wikipedia (2009) and Place Names (2009).

The precise bandwidths are more difficult to obtain. In our previous study (Liljenstam et al. 2003), we use the backbone network maps made publicly available by the ISPs; these maps contain information about the connection types between the ISPs' major Points-of-Presence (PoPs), which can be directly translated into link bandwidths. Since the routers are labeled with their geographical information, we can map the routers to the known PoPs and then assign link bandwidths according to the connection types between the PoPs. However, the information about the connection types is somewhat spotty; also we cannot really deal with the existence of multiple trunk lines between the PoP sites.

In this paper, we decide to simply use the router's relative position from the backbone routers in the ISP's network to determine the link bandwidth. We classify different link types, including backbone links, access links, peering links, and assign different bandwidths accordingly. This method is intuitive but can only be treated as an approximation. We are investigating other methods to derive the link bandwidths, possibly learning from recent network measurement studies.

We choose to use one of the tier-1 ISP networks for our study (see Figure 2). This particular network contains 637 routers (out of which 235 are backbone routers), which are connected by 1,381 links. It covers several states of the United States and several countries in Europe, Asia and Oceania. We decide to manually partition the network according to the locations of the routers (for parallel simulation). We map routers in the same states of the United States to different processors. Since 78% of the routers are in the state of California, we break the routers between those in Northern and Southern California. For the 8% of the routers outside of the United States, we map routers in the same country to the same processor. The total number of partitions is 12.

Attached to the backbone network are medium-sized stub networks, which are called the *campus networks*. We attach equal number of campus networks to each network partition and select routers at the backbone network with external links as candidates for attaching the campus networks. We attach the campus networks randomly among the candidates with a probability proportional to the number of external links.

Campus network is part of a baseline network model originally designed for benchmarking the performance of various parallel simulators. Refer to DARPA NMS Challenge Topology (2009) for a more detailed description of this network. A campus network consists of 504 end hosts, which are organized into 12 local area networks (LANs) and connected by 18 routers. 4 extra end hosts are designated to form a server cluster. Each LAN is a 10 Mb/s network, consisting of a gateway router and 42 end-hosts. The entire campus network is divided into 4 OSPF areas. The links between the routers in the OSPF backbone area and those in the server cluster are configured with 1 Gb/s bandwidth and 10 ms delay. For links connecting routers in other OSPF areas (where all other end hosts are located), we set the bandwidth to be 100 Mb/s and the link delay to be 10 ms. The campus network is connected to the outside world through a border gateway router. We attach 84 such campus networks to the tier-1 ISP network. The entire network contains 42,672 end hosts and 3,157 routers.

3 EMULATION OF A LARGE-SCALE CONTENT DISTRIBUTION NETWORK

3.1 CoralCDN

The main idea behind a content distribution network is to replicate content at the edge of the Internet, closer to the clients. In doing so, CDN can alleviate both the workload at the server and the traffic load at the network core. By storing content closer to the clients, the content providers can deliver their content much faster and more efficiently. There exist several commercial content distribution networks today, including Akamai (2009) and Mirror Image (2009). For this study, we choose to use an open-source CDN system, called CoralCDN (Freedman et al. 2004).

CoralCDN is a peer-to-peer web-content distribution network that consists of three parts: 1) a network of cooperative web proxies for handling HTTP requests, 2) a network of domain name servers (DNS) to map clients to nearby web proxies, and 3) an underlying clustering mechanism and an indexing infrastructure to facilitate DNS mapping and content distribution. In our experiment we statically map the clients to nearby Coral nodes to send HTTP requests. Thus we ignore CoralCDN's DNS redirection function and only focus on web-content distribution.

The content distribution network is built upon a number of volunteer sites that run CoralCDN. CoralCDN consists of a Coral daemon and a Coral proxy. The Coral daemon cooperates with the Coral daemons on other sites to form an indexing infrastructure, using a technique called peer-to-peer distributed sloppy hash table (DSHT), for efficient content delivery services. The Coral proxy basically performs web caching. Upon receiving an HTTP request, the Coral proxy needs to find out whether the requested object has been stored locally. If so, it returns the object immediately to the client; otherwise, the Coral proxy tries to locate the object on the peer-to-peer network. If the object is found at a peer, the Coral proxy fetches the object from the peer over the overlay network before returning it to the client. If the object is not found on the network, it is retrieved directly from the web server.

We place one CoralCDN node within each of the 12 LANs of the 84 campus network (at one of the 42 end hosts in each LAN), thus making a total of 1,008 CoralCDN nodes overall. Each CoralCDN node is emulated in a separate OpenVZ container. The web clients are simulated; they send HTTP requests to the CoralCDN node within the same LAN and subsequently receive data objects from the Coral proxy. As mentioned previously, PRIME implements a full-fledged TCP model that allows simulated nodes to interact with real TCP counterparts. We attach a stub network to a backbone router in the tier-1 ISP network (located in Paris, France) to contain a web server, which corresponds to the Apache server emulated on a separate compute node.

3.2 EmuLab Machines

We conduct our experiment 25 EmuLab machines. Each of these machines (labeled as type *pc3000*) features dual Intel Xeon 3 GHz CPUs and 2 GB memory. We load the machines with a customized disk image running OpenVZ. We designate 12 machines to run the PRIME real-time simulator. We use another 12 machines to run the 1,008 CoralCDN instances, each within an OpenVZ container. That is, we have 84 containers for each machine. We use a separate machine to run the Apache web server. In order to limit the disk space used by the 1,008 OpenVZ containers, we use a union file system, called FunionFS (2009), for directories (such as `/usr` and `/home`) common to all operating system instances.

We also develop a set of scripts to automate the experiment, including setting up (and shutting down) the experiments and collecting experiment results from the virtual containers.

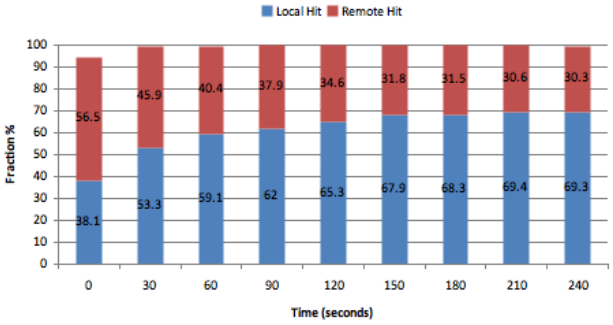

Figure 3: The CoralCDN hit rate

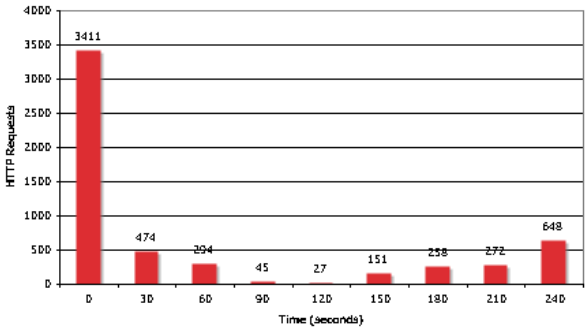

Figure 4: Apache server load

3.3 HTTP Trace

We select the HTTP trace at the 1998 World Cup web site, which is publicly available (Arlitt and Jin 1998). The trace is collected with all HTTP requests made to the 1998 World Cup Web site between April 30, 1998 and July 26, 1998, for a total of 88 days. The trace contains over 1.35 billion HTTP requests for objects of nearly 5 TB in total size. On average, the site received nearly 11,000 requests and sent about 41 MB of data per minute.

Note that in the trace almost all client requests (98.01%) were for either HTML (9.85%) or image files (88.16%). Only a very small fraction is due to dynamic content. This may not be true in more recent HTTP traces. In this respect, the trace may result in a somewhat *idealistic* caching performance. Also, due to privacy concerns, the trace has been anonymized with all clients' IP addresses removed and replaced with unique integer identifiers. In the experiment, we randomly map the clients to the end hosts. This random assignment may under-estimate the locality influence on caching, although previous studies (Wolman et al. 1999) have shown that such impact may be insignificant.

We select a 24-hour period of this trace (from June 5, 1998, 22:00:01 GMT to June 6, 1998, 22:00:00 GMT), consisting of 5,452,684 requests originated from 40,491 clients. We pre-process the trace to filter out the sequence of requests sent from each client and randomly map the 40,491 clients to the end hosts in our network model, for a complete daily pattern of the caching behavior. We implement a simple HTTP client in PRIME, which is able to send HTTP requests based on the sequence of requests recorded in the trace, and subsequently receive the data objects from the web server or proxy. On the Apache server, we generate the dummy objects beforehand according to their sizes also specified in the trace file.

4 EXPERIMENT RESULTS

During the experiment, we collect three important metrics to analyze the performance the peer-to-peer content distribution network: cache hit rate, web server load, and response time. We show the results for the first four hours in the following.

Cache Hit Rate. The cache hit rate can be obtained by analyzing the log files at the CoralCDN nodes. From the perspective of Internet service providers, higher cache hit rate means less data objects to be transferred over the global Internet. That is, more clients can be serviced under the same network configuration. Higher cache hit rate also alleviates potential congestion at the network core and improves the overall performance of the entire network.

The results are shown in Figure 3. The data were collected every 30-second interval. A high cache hit rate was achieved by the CoralCDN system. A cache hit can be distinguished by whether the requested object is found on the local CoralCDN node (we call it *local hit*), or at a remote CoralCDN node (we call it *remote hit*). The overall high hit rate is mainly due to the availability of sufficient caching space; each CoralCDN node has been configured to have a cache of 1 GB in size. That is, nearly all objects can be cached collectively by the CoralCDN system, except in the very beginning when the objects are loaded. The local hit increases and stabilizes over time as objects are moving closer to the clients.

Web Server Load. The web server load can be obtained by counting the number of HTTP requests that have made its way to the Apache web server. Under heavy traffic demand, content publishers would likely see a much lower workload at the web server as a result of caching. The traffic load at the Apache server is shown in Figure 4. The values were obtained by analyzing the Apache access log file. The figure shows the number of HTTP requests for every 30 minutes during the

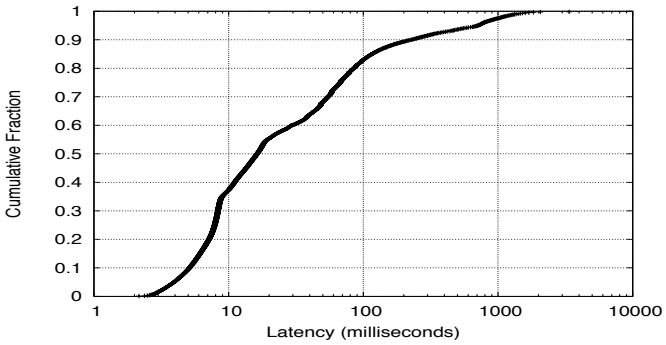

Figure 5: Response time

experiment. The server load peaks at 3,411 requests in the first 30 minutes, which are significantly higher than the rest of the simulation period, since most HTTP requests result as cache misses at the start of the experiment.

Response Time. The response time is defined as the time between a client sending an HTTP request and finally receiving the entire data object. It is a crucial metric of the overall efficiency of the content distribution network. The cumulative distribution of the response time among all clients is shown in Figure 5. Note that the x-axis is logarithmic. About 40% of the downloads took less than 10 milliseconds, while almost 98% of the downloads were completed within a second.

5 RELATED WORK

We can generally classify available experimental networking research testbeds into physical, emulation, and simulation testbeds. Physical testbeds, such as WAIL (Barford and Landweber 2003) and PlanetLab (Peterson et al. 2002), resemble the target network environment. In particular, PlanetLab offers the ability to conduct extensive live experiments directly on the Internet itself using overlay connections. In similar veins, the Global Environment for Networking Innovations (GENI) aims to provide a networking research experimentation platform consisting of programmable networking components supporting researchers conducting simultaneous experiments through virtualized computing and networking resources (GENI 2009).

Most of these physical testbeds offer emulation capabilities. By modulating network traffic in accordance with the latencies and bandwidths of the target network, emulation testbeds allow network experiments with flexible configurations. Their popularity is evident by the wide variety of network experiments they currently support. For example, VINI (Bavier et al. 2006) is an emulation environment built on PlanetLab, which consists of machines distributed across the Internet and shared by researchers simultaneously conducting experiments directly on the Internet. We have seen emulation testbeds built on a variety of computing infrastructures, ranging from dedicated high-performance computing clusters, including EmuLab and ModelNet (Vahdat et al. 2002), to distributed platforms, such as VINI, X-Bone (Touch 2000), and VIOLIN (Jiang and Xu 2004), to special programmable hardware, including the Open Network Laboratory at Washington University (DeHart et al. 2006) and the Wireless Emulator at CMU (Judd and Steenkiste 2004).

Emulation testbeds have become a mainstream method for experimental networking research, primarily due to their capability of achieving desirable realism and accuracy. One potential drawback of the emulation testbeds is that emulated network conditions are inherently dependent upon the available physical infrastructure (e.g., nodal processing speed, link bandwidths and latencies) and the traffic condition on the emulation infrastructure. These factors may dampen the flexibility and scalability of these emulation testbeds. Furthermore, one of the main objectives of PRIME is to increase the accessibility of network experimentation testbeds. PRIME's approach is to adopt commodity platforms, so that researchers would be able to run network experiments on existing computing facilities. This is in contrast to the commonly adopted approaches, such as PlanetLab, EmuLab, and GENI, which are based on resource sharing through proper resource management and delicate policy reinforcement.

Simulation testbeds, in contrast, allow examining large-scale network phenomena more expediently. However, simulation fares poorly in other aspects, particularly in its operational realism. Simulation model development is labor-intensive and error-prone; reproducing realistic network topology, representative network traffic, and diverse operational conditions in simulation is known to be a substantial undertaking. Despite these problems, simulation is effective at capturing high-level design issues, answering what-if questions, and providing preliminary analysis of complex system behaviors. Existing network

simulators include ns-2 (Breslau et al. 2000), SSFNet (Cowie et al. 1999), GTNetS (Riley 2003), ROSS.Net (Yaun et al. 2003), and QualNet (2009).

PRIME combines the advantages of both emulation and simulation. On the one hand, PRIME provides the operational realism as real systems and real implementations are included as part of emulation. On the other hand, PRIME inherits the obvious advantages of modeling and simulation, making it suitable for studying emerging complex network behaviors (potentially caused by divergent processes, multi-scale interactions, and self-organizing characteristics), and answering what-if questions, comparing design alternatives, and exploring parameter space. Similar to other real-time simulators, including NSE (Fall 1999), IP-TNE (Simmonds and Unger 2003), MaSSF (Liu et al. 2003), and Maya (Zhou et al. 2004), PRIME is based on augmenting existing network simulators with emulation capabilities. Different from these approaches, PRIME emphasizes large-scale real-time simulation enabled by parallel simulation and hybrid modeling techniques.

6 CONCLUSIONS

We conduct large-scale peer-to-peer web-content distribution network experiments using real-time simulation in the EmuLab cluster computing environment. This study demonstrates the capabilities of real-time immersive network simulation for studying large-scale complex distributed applications under semi-realistic traffic conditions.

Future work includes further streamlining the model development and deployment process, such as automated model configuration, resource allocation, and data collection, to facilitate extreme-scale network experiments.

ACKNOWLEDGMENTS

This research is supported in part by the National Science Foundation grants CNS-0836408 and HRD-0833093. We thank EmuLab at the University of Utah for allowing us to use the facility to conduct the experiments.

REFERENCES

Akamai 2009. http://www.akamai.com/. Last accessed: September 2009.

Arlitt, M., and T. Jin. 1998. 1998 World Cup web site access logs. Available at: http://www.acm.org/sigcomm/ITA/. Last accessed: September 2009.

Astrodient Atlas Database 2009. Available at: http://www.astro.com/atlas. Last accessed: September 2009.

Barford, P., and L. Landweber. 2003. Bench-style network research in an Internet instance laboratory. *ACM SIGCOMM Computer Communication Review* 33 (3): 21–26.

Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. 2003. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, 164–177.

Bavier, A., N. Feamster, M. Huang, L. Peterson, and J. Rexford. 2006. In VINI veritas: realistic and controlled network experimentation. *ACM SIGCOMM Computer Communication Review* 36 (4): 3–14.

Breslau, L., D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. 2000. Advances in network simulation. *IEEE Computer* 33 (5): 59–67.

Cowie, J., D. Nicol, and A. Ogielski. 1999. Modeling the global Internet. *Computing in Science and Engineering* 1 (1): 42–50.

Cowie, J. H., H. Liu, J. Liu, D. M. Nicol, and A. T. Ogielski. 1999. Towards realistic million-node Internet simulations. In *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*.

DARPA NMS Challenge Topology 2009. Available at: http://www.ssfnet.org/Exchange/gallery/baseline/index.html. Last accessed: September 2009.

DeHart, J., F. Kuhns, J. Parwatikar, J. Turner, C. Wiseman, and K. Wong. 2006. The open network laboratory. *ACM SIGCSE Bulletin* 38 (1): 107–111.

Delve, T. J., and N. J. Smith. 2001. Use of DaSSF in a scalable multiprocessor wireless simulation architecture. In *Proceedings of the 2001 Winter Simulation Conference (WSC'01)*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 1321–1329. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Dike, J. 2000. A user-mode port of the Linux kernel. In *Proceedings of the 4th Annual Linux Showcase & Conference*, 7. Atlanta, GA.

Erazo, M., Y. Li, and J. Liu. 2009. SVEET! a scalable virtualized evaluation environment for TCP. In *Proceedings of the 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom'09)*. Washington DC, USA.

Fall, K. 1999. Network emulation in the Vint/NS simulator. In *Proceedings of the 4th IEEE Symposium on Computers and Communications (ISCC"99)*, 244–250. Sharm El Sheik, Red Sea, Egypt.

Freedman, M. J., E. Freudenthal, and D. Mazieres. 2004. Democratizing content publication with Coral. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI'04)*, 239–252.

FunionFS 2009. http://funionfs.apiou.org/. Last accessed: September 2009.

GENI 2009. http://www.geni.net/. Last accessed: September 2009.

Google Maps 2009. http://maps.google.com/. Last accessed: September 2009.

Jiang, X., and D. Xu. 2004. VIOLIN: Virtual internetworking on overlay infrastructure. In *Proceedings of the 2nd International Symposium on Parallel and Distributed Processing and Applications (ISPA'04)*, 937–946. Hong Kong, China.

Judd, G., and P. Steenkiste. 2004. Repeatable and realistic wireless experimentation through physical emulation. *ACM SIGCOMM Computer Communication Review* 34 (1): 63–68.

Li, Y., M. Liljenstam, and J. Liu. 2009. Real-time security exercises on a realistic interdomain routing experiment platform. In *Proceedings of the 23rd Workshop on Principles of Advanced and Distributed Simulation (PADS'09)*. Lake Placid, NY, USA.

Li, Y., J. Liu, and R. Rangaswami. 2008. Toward scalable routing experiments with real-time network simulation. In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation (PADS'08)*, 23–30. Rome, Italy.

Liljenstam, M., J. Liu, and D. M. Nicol. 2003. Development of an Internet backbone topology for large-scale network simulations. In *Proceedings of the 2003 Winter Simulation Conference (WSC'03)*, ed. S. E. Chick, P. J. Sanchez, D. M. Ferrin, and D. J. Morrice, 694–702. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Liu, J. 2006. Packet-level integration of fluid TCP models in real-time network simulation. In *Proceedings of the 2006 Winter Simulation Conference (WSC'06)*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 2162–2169. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Liu, J. 2008. A primer for real-time simulation of large-scale networks. In *Proceedings of the 41st Annual Simulation Symposium (ANSS'08)*, 307–323.

Liu, J., and Y. Li. 2008. On the performance of a hybrid network traffic model. *Simulation Modelling Practice and Theory* 16 (6): 656–669.

Liu, J., and Y. Li. 2009. Parallel hybrid network traffic models. *Simulation: Transactions of the Society for Modeling and Simulation International* 85 (4): 271–286.

Liu, J., Y. Li, N. V. Vorst, S. Mann, and K. Hellman. 2009. A real-time network simulation infrastructure based on OpenVPN. *Journal of Systems and Software* 82 (3): 473–485.

Liu, J., D. Nicol, B. Premore, and A. Poplawski. 1999. Performance prediction of a parallel simulator. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS'99)*, 156–164.

Liu, J., and D. M. Nicol. 2001. Learning not to share. In *Proceedings of the 15th Workshop on Parallel and Distributed Simulation (PADS'01)*, 46–55.

Liu, J., L. F. Perrone, D. M. Nicol, M. Liljenstam, C. Elliott, and D. Pearson. 2001. Simulation modeling of large-scale ad-hoc sensor networks. In *In European Simulation Interoperability Workshop (Euro-SIW'01)*.

Liu, J., Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone. 2005. Empirical validation of wireless models in simulations of ad hoc routing protocols. *SIMULATION: Transactions of The Society for Modeling and Simulation International* 81 (4): 307–323.

Liu, X., H. Xia, and A. A. Chien. 2003. Network emulation tools for modeling grid behavior. In *Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'03)*.

Mahajan, R., N. Spring, D. Wetherall, and T. Anderson. 2002. Inferring link weights using end-to-end measurements. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (IMW'02)*, 231–236. Marseille, France.

Mirror Image 2009. http://www.mirror-image.com/. Last accessed: September 2009.

MIT Geographic Nameserver 2009. Available at: http://stuff.mit.edu/cgi/geo? Last accessed: September 2009.

Moore, D., R. Periakaruppan, J. Donohoe, and k claffy. 2000. Where in the world is netgeo.caida.org? Available at: http://www.caida.org/publications/papers/2000/inet_netgeo/inet_netgeo.html. Last accessed: September 2009.

Newport, C., D. Kotz, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. 2007. Experimental evaluation of wireless simulation assumptions. *SIMULATION: Transactions of the Society for Modeling and Simulation International* 83 (9): 643–661.

Nicol, D. M., M. Liljenstam, and J. Liu. 2003. Multiscale modeling and simulation of worm effects on the Internet routing infrastructure. In *Proceedings of the 13th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation (Performance TOOLS 2003)*.

Nicol, D. M., and J. Liu. 2002. Composite synchronization in parallel discrete-event simulation. *IEEE Transactions on Parallel and Distributed Systems* 13 (5): 433–446.

Nicol, D. M., J. Liu, M. Liljenstam, and G. Yan. 2003. Simulation of large-scale networks using SSF. In *Proceedings of the 2003 Winter Simulation Conference (WSC'03)*, ed. S. E. Chick, P. J. Sanchez, D. M. Ferrin, and D. J. Morrice, 650–657. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

OpenVZ 2009. http://openvz.org/. Last accessed: September 2009.

Peterson, L., T. Anderson, D. Culler, and T. Roscoe. 2002. A blueprint for introducing disruptive technology into the Internet. *HotNets-I*.

Place Names 2009. http://www.placenames.com/. Last accessed: September 2009.

PRIME 2009. http://www.primessf.net/. Last accessed: September 2009.

QualNet 2009. http://scalable-networks.com/. Last accessed: September 2009.

Riley, G. F. 2003. The Georgia Tech network simulator. In *Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools'03)*, 5–12.

Simmonds, R., and B. W. Unger. 2003. Towards scalable network emulation. *Computer Communications* 26 (3): 264–277.

Spring, N., R. Mahajan, D. Wetherall, and T. Anderson. 2004. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking* 12 (1): 2–16.

Touch, J. 2000. Dynamic Internet overlay deployment and management using the X-Bone. In *Proceedings of the 8th International Conference on Network Protocols (ICNP'00)*, 59–68. Osaka, Japan.

Vahdat, A., K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. 2002. Scalability and accuracy in a large scale network emulator. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, 271–284.

VMWare Workstation 2009. http://www.vmware.com/products/desktop/workstation.html. Last accessed: September 2009.

White, B., J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. 2002. An integrated experimental environment for distributed systems and networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, 255–270.

Wikipedia 2009. http://www.wikipedia.org/. Last accessed: September 2009.

Wolman, A., G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. 1999. On the scale and performance of cooperative web proxy caching. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, 16–31.

Yaun, G., D. Bauer, H. Bhutada, C. Carothers, M. Yuksel, and S. Kalyanaraman. 2003. Large-scale network simulation techniques: examples of TCP and OSPF models. *ACM SIGCOMM Computer Communication Review* 33 (3): 27–41.

Zhou, J., Z. Ji, M. Takai, and R. Bagrodia. 2004. MAYA: integrating hybrid network modeling to the physical world. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 14 (2): 149–169.

AUTHOR BIOGRAPHIES

JASON LIU is an Assistant Professor at the School of Computing and Information Sciences, Florida International University. His research focuses on parallel simulation and high-performance modeling of computer systems and communication networks. He received a B.A. degree from Beijing Polytechnic University in China in 1993, an M.S. degree from College of William and Mary in 2000, and a Ph.D. degree in from Dartmouth College in 2003. He served as a WSC proceedings editor in 2006. His email address is <liux@cis.fiu.edu>.

YUE LI is currently a postdoctoral student at the School of Computing and Information Sciences, Florida International University. Before that, he has been a post-doc at China's Tsinghua National Laboratory for Information Science and Technology from 2005 to 2007. He received his Ph.D. degree in Computer Science from Xi'an Jiao Tong University, China, in 2005. His research interests include network simulation and emulation, high performance simulation, and multimedia networking. His email address is <yueli@cis.fiu.edu>.

YING HE is a visiting student at the School of Computing and Information Sciences, Florida International University. She is currently pursuing a Ph.D. degree from the Beihang University, China. Her research interests include network measurement and modeling. Her email address is <yinghe@cis.fiu.edu>.