Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

APPLYING PARALLEL AND DISTRIBUTED SIMULATION TO REMOTE NETWORK EMULATION

Yan Gu

College of Computing
266 Ferst Drive, Georgia Institute of Technology
Atlanta, G.A. 30332-0765, U.S.A.

Richard Fujimoto

College of Computing
266 Ferst Drive, Georgia Institute of Technology
Atlanta, G.A. 30332-0765, U.S.A.

ABSTRACT

Many of today's military services and applications run on geographically distributed sites and need to be tested and evaluated under realistic scenarios with many unpredictable factors. A remote network emulation framework called ROSENET is proposed that can meet this requirement by using a remote parallel simulation server to model the wide area network and a local network emulator to provide timely QoS predictions for real world applications. This paper discusses problems faced in applying parallel and distributed simulation technique for the remote network emulation. The experimental results show that timeliness and remote accessibility are main concerns in applying parallel simulation to remote network emulation.

1 INTRODUCTION

As modern military operations are becoming increasingly reliant on network communication and connectivity, the importance of network centric warfare continues to grow in today's modern warfare. A typical military communication scenario involves heterogeneously interconnected networks in a possibly hostile setting that supports a large number of users and multimedia traffic that are severe, critical, and real-time. The network design, configuration, and deployment problems in such a domain are extremely challenging since military networks are usually large in scale, the network scenarios are often complex, the information needs to be exchanged and processed timely, and the military applications are geographically distributed over a wide area network.

Although network centric warfare has provided information superiority for modern war which translates into warfighting advantage over adversaries, information technology may fail to work as expected. In one Navy SEALs mission in 1983, a soldier in Grenada had to call air support from the base using commercial landlines because of failures in military communication. More recently, in the Iraq War in April 2003 (Talbot 2004), ground forces suffered from out of bandwidth range and software lockup problems that rendered the computer system useless. Soldiers had to stop their vehicle to receive data on enemy positions which made them easy target for enemy fire. Therefore, before deployment in an actual network environment, it is extremely important to test and evaluate these military services and applications under a wide variety of network scenarios to determine possible unexpected system behaviors.

ROSENET (Gu and Fujimoto 2004) is a remote network emulation system intended to test and evaluate distributed services and applications, including modern military applications, by integrating remote parallel simulation servers with local network emulators. It is designed to provide scale, accuracy, timeliness, flexibility, and remote accessibility. Sample military applications where ROSENET may be applied include:

- Information Assurance in Global Information Grid.
  The Global Information Grid aims at integrating all of DoD's information systems, services, and applications into a seamless, secure, and reliable information environment to achieve information superiority over adversaries and form the basis of network centric warfare doctrine. Tools are needed to ensure reliable delivery of critical messages, e.g., Call-for-fire messages, or medical evacuation orders.
- QoS in wireless networks in urban environments.
  Mobile applications operating in urban environments are becoming increasingly more important in military operations. People and vehicle movements, weather, terrain (e.g., high-rise buildings), are important and often unpredictable factors that have strong impacts on Quality of Service (QoS) for wireless communications. ROSENET's ability to integrate different simulators into one framework and make them accessible to remote users are useful features for this type of applications, e.g., to provide real time analysis and control of networks.
- Realistic communication in military training.
  The need for collective and joint training is increasing as a result of the transformation to network centric

warfare (Mevassvik et al. 2006). A testing framework such as ROSENET capable of providing realistic communication scenarios over a secure wide area network at geographically distributed training sites will provide greater realism for joint tactical training.

The performance of the remote network emulation model on a single node simulation is evaluated in (Gu and Fujimoto 2007). This paper focuses on addressing the challenges faced in applying parallel and distributed simulation techniques in the ROSENET remote network emulation, specifically for military applications. The remainder of this paper is organized as follows. Section 2 provides background information and discusses related work. Section 3 describes the design of a parallel and distributed version of the ROSENET architecture. Preliminary experimental results are presented in Section 4. Section 5 gives conclusions.

2 BACKGROUND AND RELATED WORK

Three techniques are commonly used in testing network protocols and applications. Live network testing (Bavier et al. 2004) experiments with new services directly in the real network such as the Internet. However, the underline network status is not controllable, the experiments are not repeatable, and the testing environment is not secure for military applications. Network simulation can model different types of networks independent of the underlying platforms, but the difficulty in modeling real world applications has limited the use of network simulation as a tool to test new applications and services. Network emulation (Ingham and Parrington 1994; Allman et al. 1997; Rizzo 1997; Yeom and Reddy 2001; Vahdat et al. 2002; Carson and Santay 2003; Zheng and Ni 2004) integrates real world physical applications with a virtual target network. Existing simulation tools can be used for emulation (Black and Harris 1999) if they can interact with real world applications. An emulator in this sense is also referred to as "real time simulator". The real-time constraints, also referred to as ***timeliness***, require that network emulators process events and deliver the results to applications within certain deadlines.

High fidelity simulations at large scale are always desirable for network researchers and general users, but the limited availability of physical resources may force users to trade accuracy over time. A single node emulator may not be able to simulate the network with much detail. Trace-based methods (Noble and Satyanarayanan 1997) try to make up for this inaccuracy by using real world data. However this method is only limited to specific network scenarios, and the trace data cannot be adaptable to specific application traffic. Clustered network emulators can improve the scale of network topology, but clustered machines may not be readily available and typically are not easily configured.

Parallel Discrete Event Simulation (PDES)(Fujimoto 1990) (Ferenci et al. 2002) partitions a target network and simulates it concurrently on multiple machines, thus offering the potential to simulate large scale network topologies at a detailed packet level. A number of tools have been developed utilizing parallel computing facilities to improve the scalability and performance of network simulation. Example systems include PDNS (Riley et al. 1999), GTNetS (Riley 2003) (Fujimoto et al. 2003), DaSSF (Liu and Nicol 2001) (Cowie et al. 1999), GloMoSim (Bagrodia et al. 1998) and its commercial successor QualNet , and Genesis (Liu et al. 2006). These efforts have studied large scale network simulation on a variety of platforms ranging from workstations to cluster computers to supercomputer and have demonstrated the ability to simulate a million web traffic flows with over a million network nodes in near real time.

One of the main challenges in applying parallel and distributed simulation for network emulation is to meet the timeliness requirement. Existing parallel simulation tools are designed to improve scale and accuracy, and achieving good performance is desired but not required to meet certain deadlines. In order to meet the timeliness requirements, different approaches are used when developing simulation-based emulators based on parallel simulation techniques. IP-TNE (Simmonds and Unger 2003) is a network emulator using the parallel discrete event simulation IP-TN that runs on shared memory multiprocessors. IP-TNE synchronizes only the edge of the simulated network that interacts with real traffic with wall clock time. Maya (Zhou et al. 2004) is a hybrid software emulator which uses a fluid model for TCP and ties the fluid model with the physical network interface and network statistics over a time interval calculated periodically. RINSE (Liljenstam et al. 2006) uses a priority-based schedule to prioritize emulation events in scheduling and sending the emulation packet ahead of its scheduled time from one router to another in the simulation, based on the assumption that without the physical connection latency the event would have entered the queue much earlier. Communication Effects Server (CES) (Bagrodia et al. 2006) is a wireless network simulator which adapts the concept of transactional real-time which is defined as one where the wall-clock time to execute an average transaction is less than its simulation time. In (Xu et al. 2001) real time lookahead is exploited from the interactions between wireless applications and the simulated wireless network in the parallel simulator GloMoSim since the maximum throughput of a traffic flow is limited by the link capacity. This way simulators can advance its simulation time ahead of real time to improve performance.

All of the above approaches attempt to apply parallel simulation for network emulation in order to meet the real time requirement. But they all fail to meet another requirement: ***accessibility***. Parallel computing facilities re-

quired in these network emulators may not be locally available, and co-locating application code with a remote high performance computing facility may be cumbersome and inconvenient. If a user attempts to use high performance computing facilities remotely, the real time constraints for emulation may not be met since latency between the application and remote simulation servers may exceed predicted delay. Although many research efforts have been designed to realize an emulation framework that is remotely accessible, they require participants to upload models to the test bed remotely and observe the system performance without interactions.

In addition to the previously mentioned systems targeting at applying parallel simulation in network emulation, PRIME (Liu et al. 2007) is a system most similar to ROSENET. PRIME aims to implement an open and scalable network emulation infrastructure to allow a large number of real time applications to dynamically interface with network simulators running on supercomputers. It uses a Virtual Private Network (VPN) to bridge traffic between physical entities and network simulators. Real applications run as VPN clients which automatically forward network packets to the VPN servers. VPN is used to circumvent the firewall of the supercomputing center and also serves as a network interface on the client machine.

Sharing a common goal of providing an open network emulation infrastructures to test real time applications using network simulators running on supercomputers, ROSENET differs from PRIME with its support for remote access capability. ROSENET allows users to access the high fidelity simulation remotely. The experimental results in PRIME show that with a simple dumbbell topology, losses and delays experienced by the packets as they travel through the simulation gateway and the client machines can have a profound impact on the emulation accuracy. Good results are achieved when the simulation and applications are in a local area network and results become very bad when they go through a wide area network, which is susceptible to loss and delay between them. Also the simulation gateway's bandwidth and latency could affect the quality of service on the applications. These two problems have been predicted in ROSENET design in (Gu and Fujimoto 2004). The results in PRIME further confirms that network emulation with remote accessibility is required. In this sense, ROSENET and PRIMSE are not direct competitors. ROSENET can complement PRIME by integrating PRIME into its client/server framework and using it as the high fidelity simulation server to provide remote users access.

ROSENET and PRIME also address the same problem in large scale network emulation using different approaches. In order to achieve results within emulation's real time constraints, PRIME integrates fluid model with packets to improve parallel simulation performance, while ROSENET uses network models describing network status within one time interval, instead of sending single packets as in PRIME, to trade time for accuracy. Using fluid model to improve simulation performance has been studied previously in various simulation tools and is not the focus of the ROSENET emulation system, but more of interests for a particular network simulation tool to be integrated into the ROSENET framework (such as how GTNetS uses fluid model to improve simulation performance). Its effects on network emulation performance is worth studying as future work.

3 SYSTEM ARCHITECTURE

This section describes the high level design for the ROSENET framework. The implementation of a parallel and distributed version of ROSENET using the GTNetS simulator is then described, and the time management issue is discussed as well.

3.1 Overall Architecture

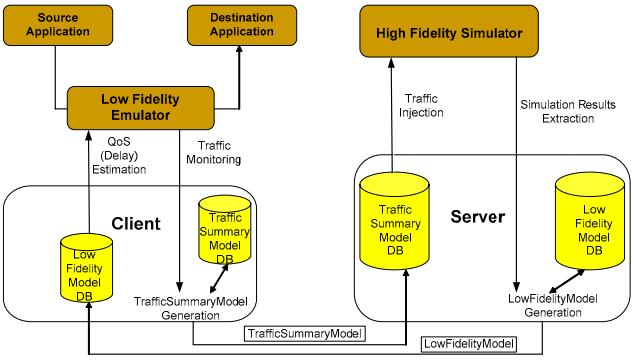

Figure 1: Overall architecture

Figure 1 shows the overall architecture of ROSENET. The client and server are the principle components of ROSENET. The low fidelity emulator provides rapid QoS estimation as well as application traffic monitoring by interacting with the local distributed applications. The remote server controls the high fidelity simulation by injecting traffic and extracting measurements such as end-to-end delay from the simulation. The simulation time is partitioned into time intervals with the assumption that the traffic characteristics change little within each time interval. Clients and servers exchange their status through periodic updates of network models at the end of each interval. The update frequency can be dynamically adjusted according to the required accuracy and capacity of the simulation as well as the available bandwidth between client and server. More detailed information about how the system works can be found in (Gu and Fujimoto 2007).

This design allows ROSENET to achieve timeliness through the low fidelity network emulation, scale and accuracy through the high fidelity network simulation, flexibility in integrating different simulator/emulators through

High Level Architecture (HLA) (Kuhl et al. 1999) and standard interface definitions to hide internal details of simulation and emulation. Finally periodic network model exchanges between simulators and emulators trade simulation accuracy for time and achieve accessibility for remote network emulation users by avoiding sending packets to remote high fidelity simulation facility to simulate, which will cause large bandwidth consumption and a wide area round-trip delay for each emulated packet.

3.2 Support for Parallel Discrete Event Simulation

In extending ROSENET's architecture depicted in Figure 1 to support for parallel discrete simulation, an intuitive approach will be to allow the emulator to directly interact with each simulator in the distributed simulation. However, this approach will cause several problems.

- Time management
  The emulator executes in real time while the distributed simulators execute in simulation time. In a parallel discrete event simulation, simulation time is maintained locally on each simulator and specific algorithms are used to manage time advance on each simulator. Thus it is rather difficult for the emulator to interact with distributed simulators with different simulation time clocks.
- Communication latency
  In ROSENET the simulators and the emulator are distributed over a wide area network. If the emulator joins the federation of the simulators, the synchronization among these federates may not be very efficient since they have to wait for the slowest federate before they can advance.
- Security
  Distributed simulators running on high performance computing facilities are usually protected by firewalls and only provide limited access to users. Allowing the remote emulator to frequently access each node in the parallel and distributed machines behind a firewall over a wide area network may be difficult to control and manage if security of these machines is not to be compromised.
- Locality
  Since the target network is partitioned and modeled by different simulators, the source and destination applications for the network emulation may be mapped as virtual nodes modeled by different simulators. The information about the locality of these mapped nodes in the distributed machines should not have to be maintained by the emulator. A thin client is desirable to reduce the computing resources needed at client machines.

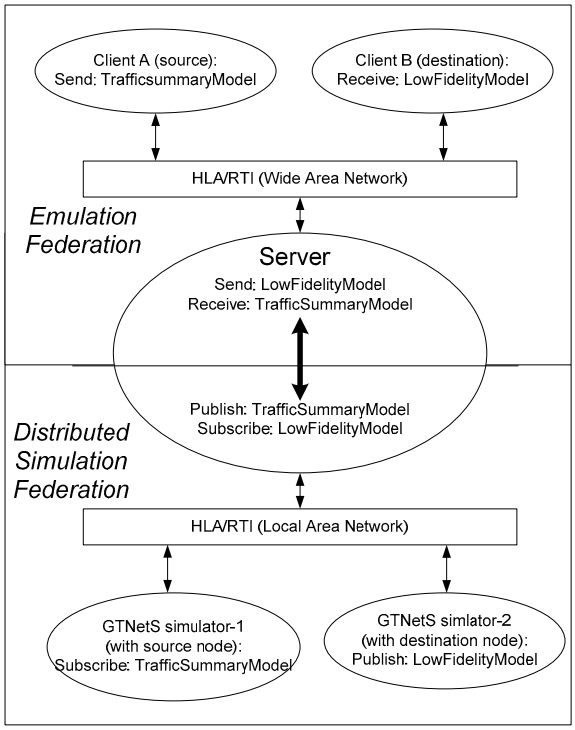

Figure 2: Parallel and distributed version

An extension of the ROSENET approach with support for parallel and distributed simulation has been developed. The parallel and distributed version of ROSENET is composed of two federations: the emulation federation and the distributed federation as shown in Figure 2. The server is involved in both federations and acts as a proxy to provide the following services:

- Network model exchange
  Two types of network models are exchanged in the distributed federation: TrafficSummaryModel and LowFidelityModel. The distributed simulation event scheduler on each simulation node receives TrafficSummaryModel generated by the remote emulation client and forwarded by the ROSENET Server, updates the corresponding virtual node's traffic patterns in its simulation, and executes the simulation till the end time of all the current TrafficSummaryModels in this federate. Compared with the previous version of ROSENET system with a sequential event simulation on a single node, this parallel and distributed version allows the source and destination applications to be modeled on different physical nodes which execute at their own pace. This requires that the traffic models be applied at the nodes at the same simulation time, which means they execute the simulation in the same time interval in ROSENET.
- Time management
  In ROSENET, the emulator and the applications execute at real time (or wall clock time) while the simulator executes at simulation time. Since the execution of

the emulation cannot be reversed as in optimistic interactive simulations, the high fidelity simulation has to follow the low fidelity emulation to avoid a causality error that occurs when traffic times-tamped behind the current simulation time is injected into the simulation. This is achieved by piggybacking time information on network models exchanged between simulator and emulator. When forwarding network models between the simulation federation and emulation federation, the proxy server also translates the timestamp on network models between real time and simulation time.

3.3 Time Management Protocols

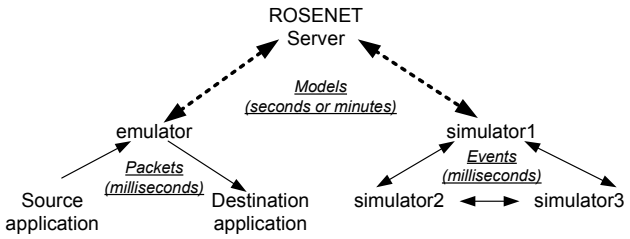

Figure 3: Time management in ROSENET

In a distributed version of ROSENET, the time management is two fold: time management among the distributed simulators and time management between the emulator and simulators. In a distributed simulation, the simulator advances its simulation time through time management service in the HLA RTI. They process events and make time advance requests to the RTI which grants time requests to make sure no causality error occurs within the distributed simulation. In this sense, time advance within simulators is fine-grain as the time advance is event-drive which is usually in the order of milliseconds. On the other hand, the time management between emulator and simulators is more coarse-grain, meaning that time advance for emulator and simulators is by time intervals which is usually in the order of seconds or minutes. The two-level time management in ROSENET is illustrated in Figure 3.

4 EXPERIMENTS

The objective of the experimental studies is to evaluate how the ROSENET system meets the timeliness requirement with support for parallel and distributed simulation. The baseline performance of the ROSENET measuring the emulation accuracy of end-to-end delay and loss has been evaluated and validated in (Gu and Fujimoto 2007) through a sequential simulation version of ROSENET.

4.1 Measurement Metrics

Network simulators are usually designed to achieve scale and accuracy. Although efficient execution of simulation models is required, no strict real-time requirements are imposed on the execution of the simulation. However in network emulation, it is required that simulation process events and deliver results to applications within certain deadlines. When measuring network emulation performance, one of the main concerns is whether the emulation results can be delivered within the real-time deadlines. This is often referred to as timeliness requirement. Timeliness can be measured using timeliness ratio which is the percentage of packets that meet the real time deadlines, or their estimated end-to-end delay. In the ROSENET system, the network emulation client can use network models to quickly generate packet QoS predictions and is unlikely to miss the timeliness requirement given that only some simple mathematical calculations with network models are needed.

Although using network models can achieve timeliness, the network models uses a network model from history data to predict QoS in the future. Hence we need another metric to measure the network emulation performance in the ROSENET system context. We call it obsoleteness or lateness, meaning how obsolete/late the network model is when generating predictions for a packet in emulation. The correctness of the emulation is directly related to lateness and an upper bound on the lateness in ROSENET emulation is particularly important to the users.

4.2 Experiment Settings

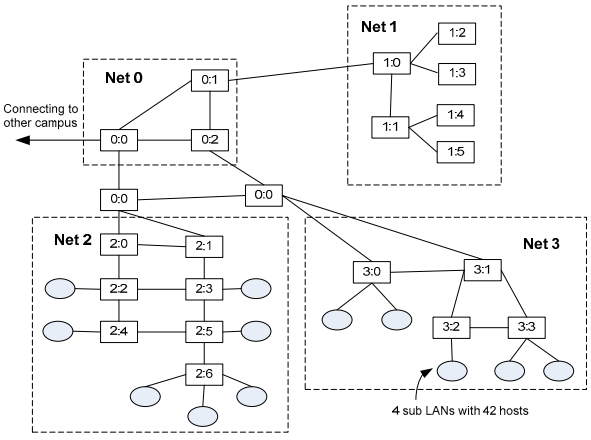

Figure 4: Basic DARPA NMS campus network

The ROSENET system is implemented using distributed GTNetS simulator and NIST Net emulator. The simulated network is a modified version from DARPA's NMS program's baseline network model (2003). The network topology is a ring of NMS campus networks. As shown in

Figure 4, each NMS campus network consists of four subnets, several routers, and a number of servers/clients generating background traffic. The hardware configuration to run the simulation experiment is a 40-node HP Integrity (2 x 900 MHz Itanium 2 IA-64) running Red Hat Enterprise Linux 4.

In this experiment, two machines in a cluster are used to execute the NMS campus network simulation, each of which models one or more of the NMS network. The source/destination nodes tested through the emulator are separately mapped to a leaf node in one NMS network on one processor and a leaf node in another NMS network on another processor. Each NMS network models 550 nodes. The background traffic is modeled as OnOff applications generating flows to neighboring NMS networks. The two mapped nodes send constant bit rate UDP packets and their end-to-end delay and loss is modeled by the distributed simulation.

4.3 Experimental Results

Two groups of experiments are performed measuring the lateness value under different scenarios. The lateness value is computed as the difference of the time when the TrafficSummaryModel's is generated and the time when the corresponding LowFidelityModel is received from the remote simulation server. The model update interval describes how often the TrafficSummaryModel and LowFidelityModel are updated, which is also the length of time of the traffic described by the models. Therefore a TrafficSummaryModel updated in one time interval needs to be generated using traffic data collected for one time interval in real time, and a LowFidelityModel for one time interval needs to be simulated in the simulator for one time interval in simulation time. The lateness value actually describes the time for the emulation client to collect the data from applications in one time interval to generate a TrafficSummaryModel, send it over a LAN or WAN to the simulator to simulate for one time interval in simulation time, and generate the LowFidelityModel to send it back to the emulation client to generate QoS predictions. Since this process is repeated and is executed in parallel by the distributed simulators and emulator, the actual lateness value is less than the sum of all the time consumed in this process due to time overlap.

4.3.1 Baseline Lateness

We first evaluate the basic lateness of the ROSENET system using a model update interval of 1 second. The emulator and simulator are placed within the Georgia Tech campus and there is an end-to-end delay within 2 milliseconds between them. Figure 5 shows the lateness value under normal execution conditions and the average lateness per model update is 1.264 seconds. Since the data collection takes at least 1 second to generate a model for 1 second traffic which is much larger than the cost in the other stages of the model update process, we perform another group of experiments using previously collected data for model generation to avoid the data collection cost, and the results are shown in Figure 6. The average lateness in this case is 0.521 seconds, which is only 40% of the lateness when data collection is involved.

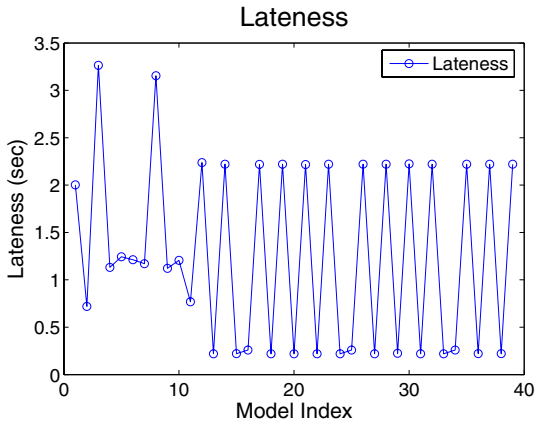

Figure 5: Lateness in normal execution

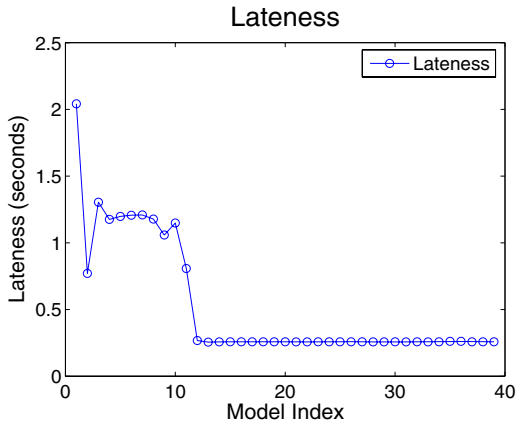

Figure 6: Lateness with pre-stored data

From both figures we can see that when the execution starts, the lateness value is varied dramatically due to the simulation/emulation warm-up period. After that the lateness value varies between 2.25 and 0.2 seconds for normal execution and remains constant at 0.2 second for execution with pre-stored data. This illustrates that the variance and extra overhead is caused by the data collection and model generation. For example, when no data is available, the models can not be generated and the system has to wait for enough data to accumulate. During this waiting and model generation period, more data is collected so that the next one or two models can be generated with much less lateness as shown in Figure 5. If all the data has been previous collected, the lateness is composed of transmission delay,

simulation cost, and model generation cost, which is of constant values. The fact that the smaller lateness in Figure 5 is around the same (around 0.2 second) as the lateness value in Figure 6 also confirms that the data collection is the main cost in lateness. This experiment shows that the data collection time, which is of one update interval time usually in the order of seconds or minutes is the bottleneck in the lateness value in ROSENET.

4.3.2 Lateness for Remote Access

In this experiment, an end-to-end delay is introduced to model the remote access to the simulation cluster by a network emulation client/user over a wide area network. According to (Floyd 2002), most (85%) of the round-trip delay in the Internet varies from 15 to 500 milliseconds. Therefore we introduce end-to-end delay in this range and observe the lateness value changes over the end-to-end delay. The update interval is 1 second and it is expected that this is the lower bound of update intervals in a remote access scenario.

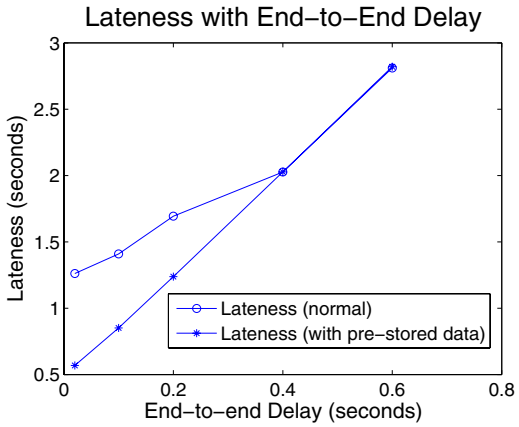

Figure 7: Lateness with varying end-to-end delay between emulation client and simulation server

Figure 7 shows the lateness value changes with the end-to-end delay between the simulation cluster and the emulation client. In comparison, we also collect the lateness value with pre-store data for model generation as in previous experiments. As the figure shows, the lateness increases with the end-to-end delay in both scenarios and the increase is linear with pre-store data. Starting from 0.4 seconds, lateness value is the same in both cases whether data collection is included or not. This is because when the end-to-end delay is 0.4 seconds, the round-trip delay (0.4 second *2) plus the overhead (0.2 second) observed from Figure 6 is around 1 second. The 1 second data collection time is overlapped by the model transmission time and the overhead cost in other components of the system. If the model transmission is larger than 0.4 second, the model transmission delay over the wide area network becomes the bottleneck in the ROSENET system. Since the update interval is likely to be much larger than 1 second and the end-to-end delay over Internet is usually smaller than 0.4 second, this is the worst case scenario performance for lateness with varying end-to-end delay.

5 CONCLUSION

This paper explains the challenges in applying parallel simulation techniques in a remote network emulation approach to meet today's new services and applications requirements, especially for military applications under the network centric warfare doctrine. This large scale network emulation framework capable of integrating a remote high fidelity simulation facility with a local network emulation addresses requirements not met in existing emulation tools. The experimental results show that timeliness and remote accessibility are main concerns in applying parallel simulation to remote network emulation, which are also desired features in testing today's military applications and services. ROSENET approach is very promising in meeting these requirements.

ACKNOWLEDGEMENT

This research was supported under DARPA contract N66001-00-1-8934 and NSF grants CNS-0540160 and ATM-0326431. The authors also thank Dr. George Riley and his students for their help with GTNetS.

REFERENCES

Allman, M., A. Caldwell and S. Ostermann. 1997. One: The Ohio Network Emulator. Technical Report TR-19972, Computer Science Department, Ohio University Athens, Ohio.

Bagrodia, R., R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park and H. Song 1998. Parsec: A Parallel Simulation Environment for Complex Systems. *IEEE Computer* 31(10): 77-85.

Bagrodia, R., K. Tang, S. Goldman and D. Kumar 2006. An accurate, scalable communication effects server for the FCS system of systems simulation environment. In *Proceedings of the 38th conference on Winter simulation*.

Bavier, A., M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink and M. Wawrzoniak 2004. Operating System Support for Planetary-Scale Network Services. In *Proceedings of 1st Symposium on Networked Systems Design and Implementation*.

Black, J. W. and D. L. Harris 1999. RTI Recommended Practices. In *Proceedings of the Fall Simulation Interoperability Workshop*, paper 078.

Carson, M. and D. Santay 2003. NIST Net: a Linux-based network emulation tool. *SIGCOMM Comput. Commun. Rev.* 33(3): 111-126.

Cowie, J. H., D. M. Nicol and A. T. Ogielski 1999. Modeling the Global Internet. *Computing in Science and Engineering* 1(1): 42-50.

Ferenci, S., R. Fujimoto, M. H. Ammar and K. Perumalla 2002. Updateable Simulation of Communication Networks. In *Proceedings of the sixteenth workshop on Parallel and distributed simulation*, IEEE Computer Society

Floyd, S. 2002. Building Models for Aggregate Traffic on Congested Links. Available via <http://www.icir.org/models/linkmodel.html> [accessed July 18, 2007].

Fujimoto, R. M. 1990. Parallel Discrete Event Simulation. *Communications of the ACM* 33(10): 30-53.

Fujimoto, R. M., K. Perumalla, A. Park, H. Wu, M. H. Ammar and G. F. Riley 2003. Large-Scale Network Simulation: How Big? How Fast? In *Proceedings of the11th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'03)*, IEEE Computer Society.

Gu, Y. and R. Fujimoto 2004. A Flexible Architecture for Remote Server-Based Emulation. In *Proceedings of the International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, IEEE Computer Society.

Gu, Y. and R. Fujimoto 2007. Performance Evaluation of the ROSENET Network Emulation System. In *Proceedings of The 11-th IEEE International Symposium on Distributed Simulation and Real Time Applications*, IEEE Computer Society.

Ingham, D. B. and G. D. Parrington 1994. Delayline: A Wide-Area Network Emulation Tool. *Computing Systems* 7(3): 313-332.

Kuhl, F., R. Weatherly and J. Dahmann 1999. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture for Simulation*. Prentice Hall.

Liljenstam, M., J. Liu, D. M. Nicol, Y. Yuan, G. Yan and C. Grier 2006. RINSE: The Real-Time Immersive Network Simulation Environment for Network Security Exercises (Extended Version). *Simulation* 82(1): 43-59.

Liu, J., S. Mann, N. Van Vorst and K. Hellman 2007. An Open and Scalable Emulation Infrastructure for Large-Scale Real-Time Network Simulations. In *Proceedings of the 26th IEEE International Conference on Computer Communications*, 2476-2480.

Liu, J. and D. M. Nicol. 2001. DaSSF 3.0 User's Manual. Available via <http://www.cs.dartmouth.edu/research/DaSSF/Papers/dassf-manual.ps> [accessed July 18, 2007].

Liu, Y., B. K. Szymanski and A. Saifee 2006. Genesis: a scalable distributed system for large-scale parallel network simulation. *Comput. Networks* 50(12): 2028-2053.

Mevassvik, O. M., K. Bråthen and R. M. Gustavsen 2006. JADE – An Experiment in Distributed Simulation Based Joint Tactical Training. In *Transforming Training and Experimentation through Modelling and Simulation*.

Nicol, D. M. 2003. DARPA Network Modeling and Simulation (NMS) baseline network topology. Available via <http://www.ssfnet.org/Exchange/gallery/index.html> [accessed July 18, 2007].

Noble, B. D. and M. Satyanarayanan 1997. Trace-based Mobile Network Emulation. In *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, ACM Press, New York, NY, USA.

QualNet. QualNet User Manual Available via <http://www.qualnet.com> [accessed July 18, 2007].

Riley, G., R. M. Fujimoto and M. Ammar 1999. A Generic Framework for Parallelization of Network Simulations. In *Proceedings of the Seventh International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 128-135.

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*, ACM Press.

Rizzo, L. 1997. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review* 27(1): 31-41.

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

Talbot, D. 2004. How Technology Failed in Iraq. *Technology Review*.

Vahdat, A., K. Yocum, K. Walsh, P. Mahadevan, D. Kosti, J. Chase and D. Becker 2002. Scalability and accuracy in a large-scale network emulator. *SIGOPS Oper. Syst. Rev.* 36(SI): 271-284.

Xu, K., M. Takai, J. Martin and R. Bagrodia 2001. Looking ahead of real time in Hybrid component networks. In *Proceedings of the fifteenth workshop on Parallel and distributed simulation*, IEEE Computer Society.

Yeom, I. and A. L. N. Reddy 2001. ENDE: An End-to-end Network Delay Emulator Tool for Multimedia Protocol Development. *Multimedia Tools Appl.* 14(3): 269-296.

Zheng, P. and L. M. Ni 2004. EMPOWER: A Cluster Architecture Supporting Network Emulation. *IEEE Trans. Parallel Distrib. Syst.* 15(7): 617-629.

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.* 14(2):149-169.

AUTHOR BIOGRAPHIES

YAN GU is a Ph.D. candidate of the College of Computing at Georgia Institute of Technology. Her research interests are in parallel and distributed systems and simulation, and her Ph.D. research is focused on developing a remote network emulation system that meets the requirements of scale, accuracy, and timeliness. In 2005 She did a summer intern at IBM Almaden Research Center. Before joining Georgia Tech, she earned a master degree at Johns Hopkins University. Her email address is <gygy@cc.gatech.edu>.

RICHARD FUJIMOTO is a professor and Chair of the Computational Science and Engineering Division of the College of Computing at the Georgia Institute of Technology. He received the Ph.D. and M.S. degrees from the University of California (Berkeley) in 1980 and 1983. Among his current activities he is the technical lead concerning time management issues for the DoD High Level Architecture (HLA) effort, and has also been chair of the steering committee for the Workshop on Parallel and Distributed Simulation, (PADS) since 1990. His email address is <fujimoto@cc.gatech.edu>.