Proc. Summer Computer Simulation Conference, Montreal, Canada, July 2003, pp. 203-209

Integrating Distributed Wireless Simulation Into Genesis Framework*

Kiran Madnani and Boleslaw K. Szymanski
Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY

Abstract

The design and evaluation of ad-hoc wireless networks introduce several unique challenges not present in wired network environment. The most significant among them include the communication medium shared between potentially thousands of nodes, the mobility of the network nodes, and the unpredictable nature of the wireless channels with problems such as fading, obstacles and interference. That makes the simulation a valuable tool to design and optimize the wireless networks and to ensure their interoperability with wired networks.

In this paper, we present wireless version of Genesis, a system for on-line, scalable simulation of communication networks using distributed computing. The presented version is implemented over GloMoSim, a simulation library for wireless network simulation. We describe challenges of implementing such a wireless version of Genesis, its performance, as well as its interoperability with Genesis components simulating wired networks.

1 Introduction

The rapid advancement in portable computing platforms and wireless communication technology has resulted in increased interest in the design and development of instantly deployable wireless networks, often referred to as "ad-hoc networks". They are indispensable in environments in which a fixed communication infrastructure, wired or wireless, either never existed or has been destroyed. The applications of such networks extend to several different application areas. In the civilian applications, they can be used to interconnect users moving in a urban or rural area or a campus and engaging in collaborative activity such as distributed scientific experiments, emergency actions or search and rescue operations. In the law enforcement sector, applications such as crowd control and border patrols are the obvious targets. In the military environment, the modern communications in a battlefield theater require a very sophisticated instant infrastructure with far more complex requirements and constraints than deliverable through wired networks, especially in view of the unpredictability of engagement areas.

The design and evaluation of these ad-hoc networks introduce several unique challenges compared to wired networks. The most significant among them include the communication medium shared between potentially thousands of nodes, the mobility of the network nodes and the unpredictable nature of the wireless channels with problems such as fading, obstacles and interference. That makes the simulation a valuable tool in studying wireless networks. Simulations enable the network designers to optimize the design of new wireless networks and plan their interoperability with wired networks. Network managers can use simulations to improve wireless network performance.

To reap the full potential of the simulation in wireless network optimization, simulation must achieve real-time or near real-time performance. This is challenging because simulating large networks at the packet level requires large computational power to execute all events that packets undergo in the network. Packets crossing the boundaries of parallel partitions impose tight synchronization between parallel processors, thereby lowering parallel efficiency of the execution. In addition, in the case of ad-hoc networks, we also have nodes crossing the boundaries of the partitions during the simulation.

To address this challenge we designed the Genesis system and adopted it for simulating ad-hoc networks. The presented system is implemented over GloMoSim, a scalable simulation library for wireless network simulation developed at UCLA [10]. In the paper, we describe the design of the system, as well as its performance and interoperability with wired network simulators.

1.1 Genesis Overview

Although our approach has been described earlier [9, 6], we provide a brief summary here, to make

*This work was partially supported by the DARPA Contract F30602-00-2-0537 with the Air Force Research Laboratory (AFRL/IF) and by the URP of CISCO Systems Inc. The content of this paper does not necessarily reflect the position or policy of the U.S. Government or CISCO Systems—no official endorsement should be inferred or implied.

the paper self-contained. The system is able to use different simulators in a single coherent network simulation, hence we called it General Network Simulation Integration System, or *Genesis* in short.

In Genesis, each network domain consists of a subset of network sources, destinations, routers and links that connect them. It is simulated concurrently with other domains and repeatedly iterates over the same simulation time interval, exchanging information with other domains after each iteration. In the initial iteration, each domain assumes either zero traffic flowing into it (when the entire simulation or a particular flow starts in this time interval) or the traffic characterization from the previous time interval. External traffic into the domain for all other iteration steps is defined by the activities of the external traffic sources and flow delays and packet drop rates defined by the data received from the other domains in the previous iteration step.

Each domain simulator creates all flows whose sources are within this domain by itself, but needs to approximate flows with the sources that are external to its domain. This approximation is achieved as follows. In addition to the nodes that belong to the domain by the user designation, Genesis creates also *domain closure* that includes all the sources of flows that reach or pass through this domain. Since those are copies of nodes active in other domains, we call them *source proxy*. Each source proxy uses the flow definition from the simulation configuration file and the native traffic generator.

The flow delay and the packet drop rate experienced by the flows outside the domain are approximated by the random delay and probabilistic loss applied to each packet traversing in-link proxies. These values are generated according to the average packet delay and its variance as well as the observer packet loss frequency communicated to the simulator by its peers at the end of simulation of each time interval. Each simulator collects this data for all of its own out-link proxies when packets reach the destination proxy.

Each delay at the router is the sum of constant processing, transmission and propagation delays and a variable queuing delay. If the total delay over all external routers is relatively constant in the selected time interval, the actual delay can be approximated by randomly generated delay from the distribution with the same average value and variance as observed in the other domains and packet loss can be applied randomly with the probability defined by the observed frequency of the actual packet loss on the external path. These three values (average packet delay and its variance and the frequency of packet drop) are send to the source proxy to be used in generating the flow. Thanks to the aggregated effect of many flows on queue sizes, this delay changes more slowly than the traffic itself, making such model precise enough for our applications.

Our experience indicates that communication networks simulated by Genesis will converge thanks to monotonicity of the path delay and packet drop probabilities as a function of the traffic intensity (congestion).

The efficiency of our approach is greatly helped by the non-linearity of the sequential network simulation. It is easy to notice that the sequential simulation time grows faster than linearly with the size of the network. Some of our measurements [6] taken over the hierarchical networks indicate that the dominant term is of order $O(n^2)$ even for small networks. We conclude that it is possible to speed up the sequential network simulation more than linearly by splitting it into smaller networks and parallelizing the execution of the smaller networks. With modest number of iterations the total execution time can be decreased by the order of magnitude or more. Example of the superlinear speedup for 4 and 16 domain simulations of mixed TCP and UDP traffic is shown in Figure 1.

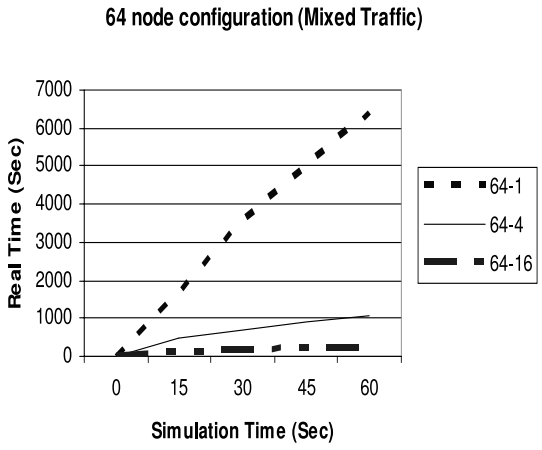

Figure 1: Simulation Times for 2,000 UDP and 1,000 TCP Flows in the 64-router Network Split into 4 and 16 Domains.

The extension of a wired network simulator needed for Genesis are quite standard and include the following components.

Domain Definition: which must be introduced by the user in the native simulation configuration file. It simply enumerates nodes belonging to each domain or labels each nodes with the domain identifier.

Source and Link Proxies: which are introduced by the Genesis based on the domain definition. Proxies belong to the domain closure. Source proxies represent sources that produce flows crossing or directed to the domain. Link proxies approximate packet delays and packet dropping on the path from the source to the domain boundary.

Data Collector: that is added to each flow leaving a domain. It collects the data about the packet delay and dropping from the flow source to the domain boundary.

Checkpointing and Freeze Event: that enable domain simulation synchronization. Freeze event causes all the simulators to stop at the same simulation time and it is added to the future event list by the Genesis system (this is the only new event introduced by Genesis). Checkpointing uses fast, diskless and application independent fork-based memory copy [2] to create a copy of each simulator at the beginning of each simulated interval. At the end of simulated interval, all domains either reactivate a copy (re-simulating the same interval but withe new data about external flow) or delete the copy and continue simulation into a new time interval.

1.2 Genesis Distributed Wireless Simulation Overview

The rapid advancement in portable computing platforms and wireless communication technology has led to significant interest in the design and development of mobile networks [3]. However, the management and evaluation of such networks presents unique challenges. Some of these include the shared broadcast medium between thousands of nodes, the mobility of these nodes and the unpredictable nature of the wireless channel with problems such as fading, obstacles and interference. The design overview of the Genesis interface to mobile network simulator, GloMoSim [8] is described in this section. The next few sections describe the design and implementation of this interface in detail.

As in the previous interfaces, we decompose the network into domains that contain network nodes, which in this case can be mobile. Thus, a domain is defined by the geographical area that it covers. The user's domain definition is a Genesis specific part of the GloMoSim configuration file. It contains also the radio-range of a node and using these parameters, Genesis computes the *closure* of the domain defined as the conjunction of the domain proper and its boundary regions of the radio-range width. The closure enables the system to account for nodes which lie outside the domain but still can interfere with the nodes inside it. Based on the domain closure, Genesis identifies the nodes active for each domain.

As in the Genesis interface to wired networks, domains are simulated concurrently with each other over the same time interval. The domains freeze [6] at user-specified intervals. At the time of freeze the inter-domain data exchange takes place. In GloMoSim, a node can schedule events (transmit and receive packets) while it is mobile. The current Genesis extension to GloMoSim accounts for the "mobility-trace" defined mobility in which the user specifies the speed, start and destination locations of the nodes in a configuration file. Knowing the above parameters, Genesis computes before the simulation the time and location at which the node crosses the domain boundaries (see Figure 2). Using this information, each domain simulator knows when and where the mobile node will be active in its domain.

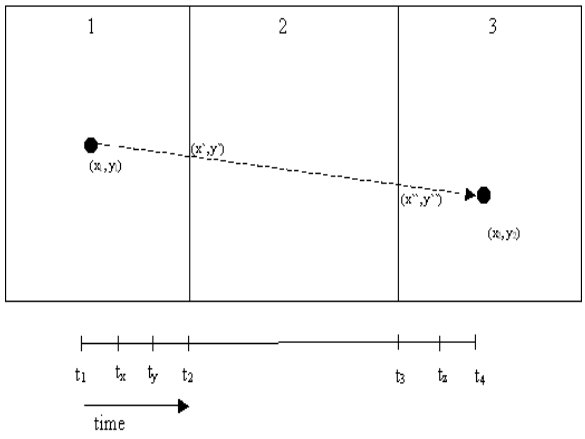

Figure 2: Example showing the mobility of a node

The introduction of domain closures creates regions in the network topology which overlap at least two domains. Thus, a node in such a region is active in both domains at the same time. The Genesis domain simulators which simulate activities of such a node must include the same events for the node. To achieve this, the inter-domain messages include information about communication (packets received and sent) by nodes ly-

ing in the domain-closure. Each domain receiving this information checks if the same communications were executed for its copy of the nodes in question. If not, the time interval is re-simulated with the modified list of events for the offending node.

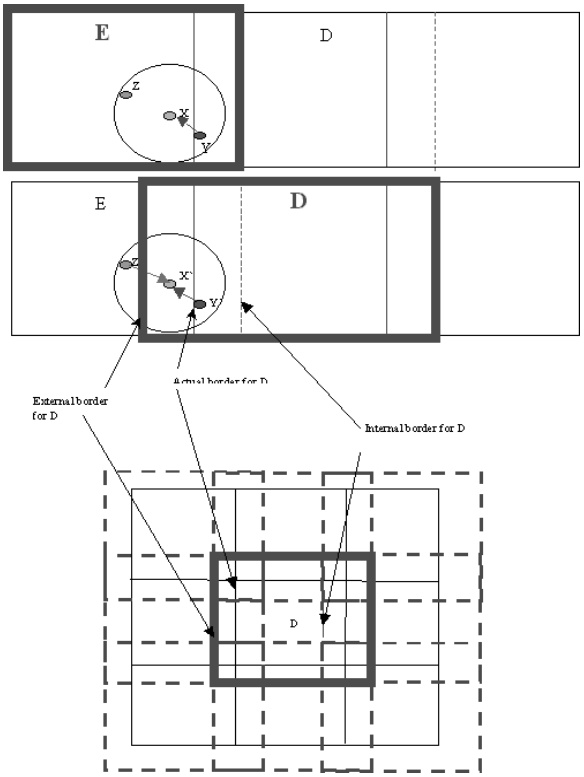

Figure 3: General Description of Domain D and its 8 neighbors

Each domain has at most eight domains as neighbors (see Figure 3). Thus, each domain needs to communicate information about the activity of domains lying in its closure to its corresponding neighbor only. We achieve this by establishing a peer-to-peer connection between domains. In other words, each domain receives data from at most eight of domains during the freeze event. On exchange of this information, each domain checks whether it needs to go-back and re-simulate the freeze interval (based on the information collected and its own information).

All the domains must simulate the same time interval, that is, all domains must simulate the next iteration at the same time. In order to achieve this synchronization, a simple farmer-worker architecture is overlayed over all domains. Once each worker (from now on, we use the terms worker and domain interchangeably unless otherwise specified) has made a decision to go-back/go-ahead, it informs the farmer about the same. The farmer then broadcasts a re-check signal to all the domains along with the domain id's of all the domains that need to go-back. This is to force all domains to re-check their logs if any of their neighbors have resimulated the freeze interval (this is to prevent cascading conflicts - a chain of conflicts in one domain which may lead to conflicts in another domain, see Figure 4). The farmer then waits for all domains to again resend their status for this iteration. Thus, only when all the domains are ready to go-ahead, the farmer sends a sync-signal to all the workers. In other words, only when all workers send the farmer a go-ahead signal, the farmer in-turn sends them all a signal to go-ahead. Each worker waits needs to receive this sync from the farmer before it can go-ahead. Thus in this way, synchronization is based on messages between the farmer and worker, and the former is used to identify the state of the simulation.

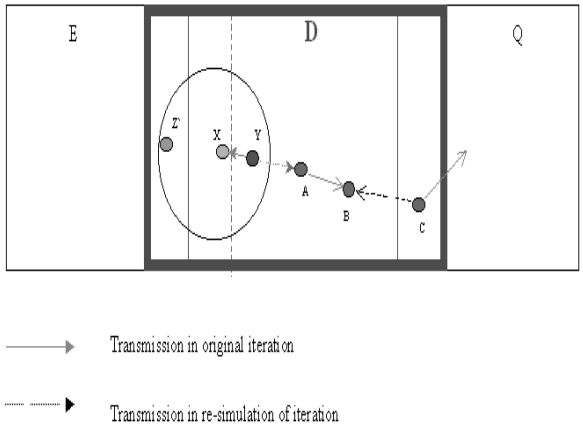

Figure 4: Chain of events in one domain causing a cascading conflict in the neighboring domain

The following algorithms briefly describes the Genesis operation for distributed wireless simulations.
Each domain iterates over the following steps till the end of simulation:

1. Send up to 8 messages to neighboring domains.

2. Receive logs from neighbors and compare the received and locally logged data.

3. If there are conflicts between logs, the domain which needs to take corrective action goes-back and re-simulates the freeze interval after informing the

farmer about the conflict. Otherwise, the domain sends a go-ahead signal to the farmer.

4. Wait for a signal from the farmer. If the farmer sends a recheck signal, then the domain checks to see if its neighbor has gone back. If not, the domain sends back a go-ahead signal to the farmer. If yes, it rechecks the logs of the neighbors with that of its own for any (cascading) conflict. If the domain then needs to go-back, it informs the farmer about it. Otherwise, the domain waits for a sync signal from the farmer to go-ahead and simulate the next simulation time interval.

The farmer executes the following steps till the end of simulation.

1. Wait for a go-ahead/go-back signal from all workers(domains). If any domain reports going-back, the farmer sends a recheck broadcast to all the domains along with the id's of the domains going-back; the farmer then waits for all domains to resend their status for this iteration. Otherwise, the farmer busy-waits until all workers can go-ahead.

2. When all workers can go-ahead, the farmer broadcasts a sync message to all workers to simulate the next time interval.

2 Interoperability with other wired network simulation engines

As part of the Genesis project, we have previously demonstrated interoperability between Java-based SSFNet and C++/TCL-based ns2 which required the definition of a generic network model and a flow-based message exchange format [1]. We adopt a similar approach (described below) in order to demonstrate interoperability between SSFNet and GloMoSim. Our main objective is to create a scenario where we have mixed-mode traffic between a wired network (modeled using SSFNet) and a wireless network (modeled using GloMoSim).

In order to interoperate between the above mentioned simulators, our network configuration includes wired domains simulated by SSFNet and wireless domains simulated by GloMoSim. The SSFNet part of the network will view the wireless GloMoSim domains as a single node network, which is the fake source and sink for all traffic originating and destined respectively to the latter. Similarly, for GloMoSim, the SSFNet domains are represented by a single (fake) source and sink node. At each freeze interval, the information about delay-drop rates (just as in Genesis interoperability interface for ns2 and SSFNet) is exchanged for inter-domain traffic. This information about flows is exchanged only for cut-flows (as in ns-SSFNet interoperability). Based on this information and local conditions in the domain, a decision whether to go back or not is made by each of the domains (at each freeze interval). If all (wired and wireless) domains are convergent and can go ahead, the exchanged delay-drop information is used by the the respective proxy sources to generate packets in the next freeze interval.

Number of Domains	1	4	16
Flows per domain	1120	280	70
Internal flows per domain	1120	224	56
External flows per domain	1120	56	14
Max/Min flow intensity	1/98	1/98	1/98
Average Times	13827	3287	943
Speedup	1	4.2	14.7

Table 1: Measurements results on IBM Netfinities (times are in seconds)

3 Experiments

In order to test the performance of the above devised interface, we configured a simple network topology. We designed a network configuration consisting of 900 nodes over a geographical area of 225,000 square-meters and perimeter of 6000 meters (that is, it is a square of 1500 meters by 1500 meters). We assumed that the mobile nodes moved with a velocity of 50m/hour. There are 15 mobile nodes in each domain (for every node that leaves the domain, there is a node that enters the domain from the corresponding opposite boundary). Figure 5 shows a sample scenario. Also for these experiments, we used CBR traffic. About 80 percent of the traffic was intra-domain, that is between nodes within a domain, and 20 percent of the traffic was inter-domain. This was done in order to reduce cost of synchronization between domains.

3.1 Distributed Simulation Experiments

We conducted tests for 3 different sizes of domains - 1 domain (vanilla GloMoSim), 4 domains and 16 domains. The simulation time was 200 seconds with a freeze interval of 20 seconds. The intensity of flows shown in the Table 1 represents the time interval in seconds between the transmission of 2 packets. All of these experiments were carried out on a cluster of IBM Netfinity processors. Figure 6, shows the speed-up for 4 and 16 domains respectively. The Genesis interface for distributed wireless simulations outperformed the sequential simulation to produce a speed-up of 4.2 and 14.7 for 4 and 16 domains respectively.

To measure the accuracy of the simulation runs, the flows of the sequential runs were compared with those

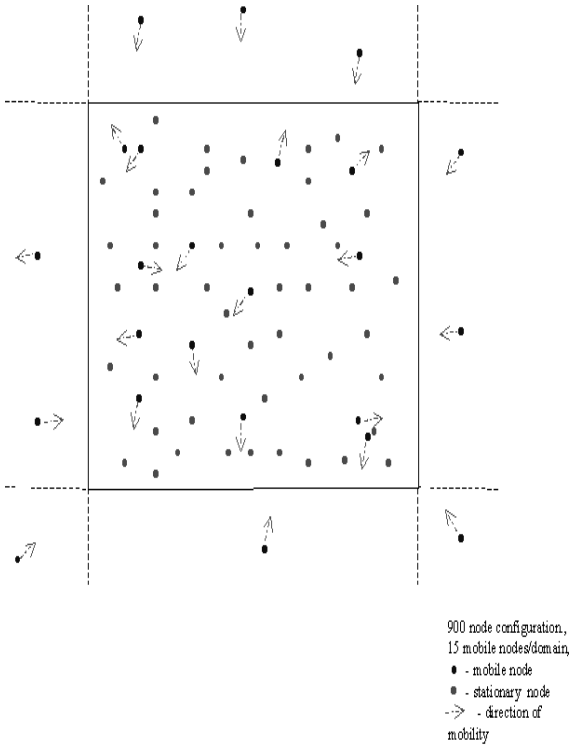

Figure 5: Sample network configuration showing a domain of the topology

of the parallel runs. We monitored the flow statistics of the distributed simulations with that of the sequential simulation. Since the number of nodes in the closure of any domain is always small (these nodes are the ones which cause the inconsistency in the simulation), the differences in these statistics was observed due to these nodes was minimal too. Comparison of the throughput, drop-rates and delays indicated that the values of the distributed simulation differed by about 4.3 percent when compared to the corresponding of the sequential simulation as shown in table 2.

3.2 Experiments on Interoperability with SSFNet

To demonstrate interoperability between wired and wireless domains, we used the network configuration shown in Figure 7 that consists of 4 domains, each containing 4 nodes. Three of these domains are wired, simulated by SSFNet; while one domain contains wireless nodes, simulated by GloMoSim.

In order to demonstrate interoperability, we had 2 TCP flows, one each from a node in the SSFNet domain to the GloMoSim domain and vice-versa (as shown

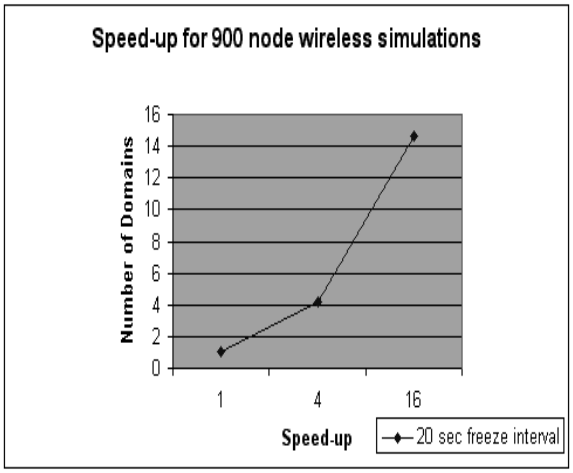

Figure 6: Speed-up versus Domain Size

in Figure 7). For each flow, we short-cut it to the proxy sink in the respective domain. During the freeze-interval, delay and drop information of these flows is communicated to the domain in which the destination domain lies. Each domain also makes a decision to go-ahead or back based on past and present parameters. Once all the domains can go-ahead, in the next iteration, the fake source in the destination domain makes use of this information to generate packets to the destination node.

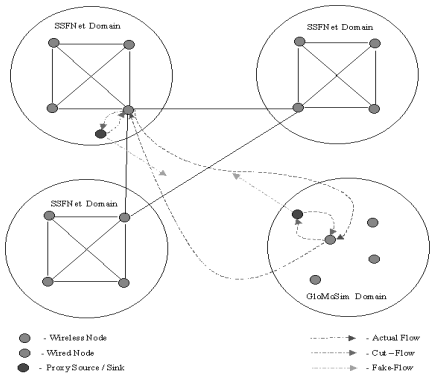

Figure 7: Network Used in Interoperability Experiment

For the described configuration, the average time the simulation took to execute for a stand-alone GloMoSim configuration was 10.52 seconds while the corresponding value for the SSFNet configuration was 8.43 seconds. The time of execution for the interoperable configura-

Number of Domains	1	4
No. of flows averaged	1120	1120
Throughput(bits/secs):		
Sample inter-domain flow	2128.4	2190.7
Sample intra-domain flow	2143.7	2180.3
Average	2133	2183.7
Delays(secs):		
Sample inter-domain flow	0.00282	0.00301
Sample intra-domain flow	0.00261	0.00263
Average	0.00273	0.00281
% of packets dropped		
Sample inter-domain flow	2.15	2.24
Sample intra-domain flow	2.08	2.10
Average	2.10	2.19

Table 2: Comparison of results of sequential and parallel simulations

tion was 12.38 seconds. Thus, the time of execution for the stand-alone configurations is comparable to that of the interoperable one. The latter result is greater because of the time taken for synchronization between the 2 different network simulators (this cost would be minimum when the configuration is scaled). We believe that equally good results would be achieved if the network is scaled in terms of the number of domains, number of flows and number of nodes per domain. The time of simulation execution is in direct relation to the latter three parameters.

4 Conclusion and Future Work

In this paper, we presented a novel approach to large scale wireless network simulation, that combines simulations of distributed domains and models it as a single system. The model is run until it converges to the fixed point solutions so each domain produces the required outputs based on the received inputs. Each model is fed by the data produced by the simulation (of the neighboring domains) and sends its output to other simulations (domains).

The worked described here extends the Genesis techniques to wireless simulations using peer-to-peer connections between neighboring domains. Using Genesis approach, we achieved a superlinear speed up of a wireless network simulation on distributed computer architecture.

Some of the directions to improve the current implementation may focus on extending supported protocols to include 802.11-based MAC and to expand mobility types beyond "mobility-trace". Another improvement would be to optimize the peer-to-peer protocol used for communication between neighboring domains to improve the speed-up of the simulation. Finally, the interoperability may be extended to include wired simulation domains running SSFNet simulator.

References

[1] B. Szymanski, Q. Gu and Y. Liu, "Time-Network Partitioning for Large-Scale Parallel Network Simulation under SSFNet," *Proc. Applied Telecommunication Symposium*, San Diego, CA, April 2002, SCS Press, pp. 90-95.

[2] C. Carothers and B. Szymanski B., "Checkpointing Multithreaded Programs," *Dr. Dobb's Journal*, vol. 15, no 8, August 2002, pp. 45-60.

[3] M. Gerla and J.T.-C.Tsai, "Multicluster, mobile, multimedia radio network," *ACM/Baltzer Journal of Wireless Networks*, vol. 1, (no 3) 1995, p.244-265.

[4] L. A. Law, and M. G. McComas, "Simulation Software for Communication Networks: the State of the Art," *IEEE Communication Magazine*, vol. 32, pp. 44-50, 1994.

[5] R. M. Fujimoto, "Parallel Discrete Event Simulation," *Communications of the ACM*, vol. 33, pp. 31-53, Oct. 1990.

[6] B. Szymanski, Y. Liu, A. Sastry, and K. Madnani, "Real-Time On-Line Network Simulation," *Proc. 5th IEEE International Workshop on Distributed Simulation and Real-Time Applications DS-RT 2001*, August 13-15, 2001, IEEE Computer Society Press, Los Alamitos, CA, 2001, pp. 22-29.

[7] B. Szymanski, A. Saifee, A. Sastry, Y. Liu and K. Madnani, "Genesis: A System for Large-scale Parallel Network Simulation," *Workshop on Parallel Network Simulation*, Washington D.C., May 2002, pp. 89-96.

[8] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks," *Proceedings of the 12th Workshop on Parallel and Distributed Simulations*, May 26-29, 1998, Banff, Alberta, Canada.

[9] J. F. Zhang, J. Jiang and B. K. Szymanski,"A Distributed Simulator for Large-Scale Networks with On-Line Collaborative Simulators," *Proc. European Multisimulation Conference*, vol. II, SCS Press, 1999, pp. 146-150.

[10] GloMoSim: http://pcl.cs.ucla.edu/projects/glomosim/