GloMoSim: A Scalable Network Simulation Environment

Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Ken Tang, Rajive Bagrodia, Mario Gerla
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

Large-scale hybrid networks that include wireless, wired, and satellite based communications are becoming common in both military and commercial situations. This paper describes a scalable simulation environment called GloMoSim (for Global Mobile Information System Simulator) that effectively utilizes parallel execution to reduce the simulation time of detailed high-fidelity models of large communication networks. The paper also presents a set of case studies that evaluate the performance of large wireless networks with thousands of nodes and compares the impact of different lower layer protocols on the performance of typical applications.

1. Introduction

High-level design problems for the digital communication infrastructure in the military and commercial environment, are extremely challenging in a number of dimensions: the scale is large, network traffic is a mix of voice, data, and imagery; connectivity can change dynamically in unpredictable ways, and the quality of service requirements are often severe. Fig. 1 presents a sample military communication scenario that includes wireless, wireline, satellite, and airborne communication assets. The total number of communication devices in such scenarios is often in the thousands and for even modest deployments; it can scale up to the tens of thousands. For joint force exercises, having hundred thousand communication units is not unrealistic.

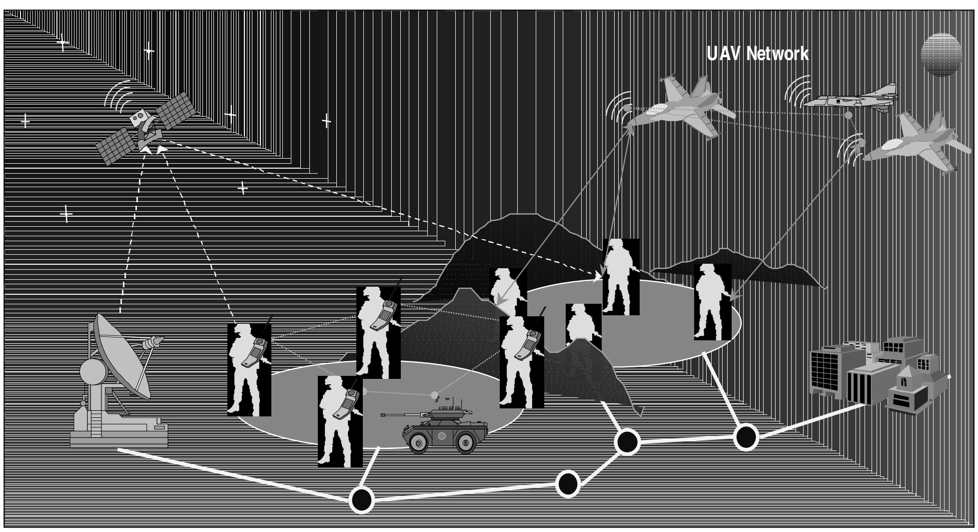

Fig. 1: Warfighter Information Network (WIN)

The multitude of proposed solutions at each protocol layer for wireless, wireline, and satellite networks has led to an explosion in possible design choices for networks such as these. The size of these networks makes experimentation and measurement prior to deployment impossible, yet the risks of deploying these new technologies in critical situations require assurance that they will work. Detailed, high fidelity simulation of the communication infrastructure can provide invaluable insights. Under current funding from DARPA, we are developing a scalable simulation facility whose objective is to simulate networks with upto hundred thousand nodes linked by a heterogeneous communications capability that includes multicast, asymmetric communications using direct satellite broadcasts, multi-hop wireless communications using ad hoc networking, and traditional Internet protocols. The scalability of the simulator to very large networks will be achieved primarily by exploiting parallelism on state of the art parallel computers. We have already demonstrated the feasibility of using parallel model execution to achieve dramatic reductions in execution times of such models. For instance, we have been successful in running a detailed simulation of a large wireless network with 10,000 mobile radios. Using parallel execution, it was possible to reduce the execution time sufficiently such that a model with 10,000 wireless nodes could be simulated on a 6 processor symmetric multiprocessor in less time than a network with half as many nodes using purely sequential execution.

Section 2 provides an overview of related work in the area of network simulators. Section 3 provides a description of PARSEC, the parallel simulation language being used to develop GloMoSim. Section 4 gives an overview of the parallel library of network models that is being developed. The following section presents the results of a few selected experiments to demonstrate the scalability of the simulator to very large networks. Section 6 and 7 present two case studies in the use of the simulator to evaluate the performance of real world applications – TCP performance over MAC layer protocols and replicated file systems in a wireless environment. This is followed by the conclusion.

2. Related Work

A number of sequential network simulators, both commercial and university research projects, have been developed [4, 11]. Well known commercial simulators include Bones from Cadence, COMNET from CACI, and Opnet from Mil3 [1]. None of these tools have been used for the large scale simulations described in this paper. Widely used public domain simulators include NS developed at LBNL and is being used to develop an Internet simulator called VINT at ISI [11]. NS is basically a transport-level simulator that supports several flavors of TCP (include SACK, Tahoe and Reno) and router scheduling algorithms. Models can be described using a variation of the Tool Command Language, Tcl. The VINT effort is a recent start that aims to develop a comprehensive simulator for the Internet; it does not address the use of parallel execution and to the best of our knowledge has not been used for detailed high fidelity simulation of large networks.

Parallel network simulation research has been conducted in the context of wired networks that include ATM networks, LAN simulators, and interconnection networks for parallel computers [3, 5, 8, 9]. Wireless networks on the other hand are inherently different. The signal interference and attenuation concerns are inherently more complicated and typically more computationally intensive for wireless mediums than for wired mediums. Also, the broadcast nature of wireless radio transmission makes communication topology in simulation models relatively denser than for an equivalent wired network. Both node mobility and ordinary communication between nodes causes inter-processor communication, which dramatically increases inter-processor communication topology.

3. PARSEC

PARSEC (for PARallel Simulation Environment for Complex systems) is a C-based simulation language developed by the Parallel Computing Laboratory at UCLA, for sequential and parallel execution of discrete-event simulation models [2]. It can also be sued as a parallel programming language. PARSEC runs on several platforms, including most recent UNIX variants as well as Windows.

PARSEC adopts the process interaction approach to discrete-event simulation. An object (also referred to as a physical process) or set of objects in the physical system is represented by a logical process. Interactions among physical processes (events) are modeled by time-stamped message exchanges among the corresponding logical processes. One of the important distinguishing features of PARSEC is its ability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures. PARSEC is designed to cleanly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute it. Thus, with few modifications, a PARSEC program may be executed using the traditional sequential (Global Event List) simulation protocol or one of many parallel optimistic or conservative protocols. In addition, PARSEC provides powerful message receiving constructs that result in shorter and more natural simulation programs.

4. GloMoSim Library

GloMoSim is a scalable simulation library for wireless network systems built using the PARSEC simulation environment [2]. Table 1 lists the GloMoSim models currently available at each of the major layers. GloMoSim also supports two different node mobility models. Nodes can move according to a model that is generally referred to as the “random waypoint” model [10]. A node chooses a random destination within the simulated terrain and moves to that location based on the speed specified in the configuration file. After reaching its destination, the node pauses for a duration that is also specified in the configuration file. The other mobility model in GloMoSim is referred to as the “random drunken” model. A node periodically moves to a position chosen randomly from its immediate neighboring positions. The frequency of the change in node position is based on a parameter specified in the configuration file.

Layer:	Models:
Physical (Radio propagation)	Free space, Rayleigh, Ricean, SIRCIM
Data Link (MAC)	CSMA, MACA, MACAW, FAMA, 802.11
Network (Routing)	Flooding, Bellman-Ford, OSPF, DSR, WRP
Transport	TCP, UDP
Application	Telnet, FTP

Table 1: Models currently in the GloMoSim library.

In contrast to existing network simulators such as OPNET and NS, GloMoSim has been designed and built with the primary goal of simulating very large network models that can scale upto a million nodes using parallel simulation to significantly reduce execution times of the simulation model. In the next sub-sections, we explore the techniques of node and layer aggregation that are used to achieve this scalability.

As most network systems adopt a layered architecture, GloMoSim is being designed using a layered approach similar to the OSI seven layer network architecture. Simple APIs are defined between different simulation layers. This allows the rapid integration of models developed at different layers by different people. Actual operational code can also be easily integrated into GloMoSim with this layered design, which is ideal for a simulation model as it has already been validated in real life and no abstraction is introduced. For example, a TCP model was implemented in GloMoSim by extracting actual code from the FreeBSD operating system. This also reduces the amount of coding required to develop the model. The simple APIs that are currently implemented in GloMoSim are also presented in a following sub-section.

4.1 Node Aggregation

In PARSEC, a simple approach to designing a network simulation model is to create each network node as an entity. Although this approach is easy to understand, it has scalability problems. If an entity has to be instantiated for each node, the memory requirements would increase dramatically for a model with large number of nodes because each entity requires additional memory to work as an independent process. The performance of the simulation would also degrade due to context switching overheads among many entities. Hence, initializing each node as a separate entity inherently limits the scalability and performance of the simulation.

To circumvent these problems, node aggregation was introduced into GloMoSim. With node aggregation, a single entity can simulate several network nodes in the system. A separate data structure representing the complete state of each node is maintained within the entity. When the simulation code for a particular node is being executed it does not have access to the data structures of other nodes in the simulation. The node aggregation technique implies that the number of nodes in the system can be increased while maintaining the same number of entities in the simulation. In fact, the only requirement is that we need only as many entities as the number of processors on which the simulation is being run. Hence, a sequential simulation needs only one entity in the simulation. With the node aggregation technique, the memory and context switching problems are eliminated.

In GloMoSim, each entity represents a geographical area of the simulation. Hence, the network nodes that a particular entity represents are determined by the physical position of the nodes. For example, suppose we specify a geographical area of 100 by 100 meters in the simulation and set the number of x and y partitions to be 2 for a particular simulation. There would be four partitions in the simulation, where each partition is represented by a single entity. Fig. 2 shows how the terrain would be divided into the four partitions. One particular partition in the simulation would encompass the area represented by the coordinates (0, 0), (49, 0), (0, 49), and (49, 49).

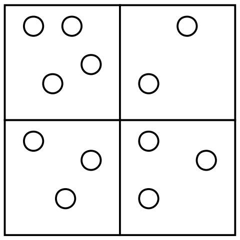

Fig. 2: A simulation with geographical area of 100 by 100 meters, which is divided into four partitions.

In a basic usage of GloMoSim, each entity represents a regular rectangular region (partition). Thus, a partition can have at most eight neighboring partitions. When a network node sends out a message, the message has to be sent to at most the eight neighboring entities in the simulation. This is much simpler than the simple design mentioned previously. If each entity represents a single network node, broadcasting a message from a node is very difficult. The first option to implementing broadcasting is that each entity constantly keeps track of the other entities that are within its power range. This option is difficult since the network topology would constantly change as mobility is introduced into the simulation. The second option is that when a node sends a message, it would be sent to all the other entities in the simulation. The receiving entity will accept the message as long as it is in the power range of the sender. This becomes highly inefficient as the number of nodes in the simulation increases. Hence a simple message transmission becomes very complicated when node aggregation is not used. With node aggregation, each entity can examine which node can receive a packet within the entity and send messages only to the neighboring entities where the packet can be reachable.

4.2 Layer Aggregation

Since GloMoSim is being built using a layered approach, the ability to rapidly integrate models developed at different layers by different people is very important. Hence, the simple approach would be that each layer in the simulation would be represented by a different Parsec entity. Some of the problems of using an entity to represent a single node reappear. As the number of layers in the simulation increases, the number of entities in the simulation also increases. This leads to scalability and performance problems in the simulation. This is not as dramatic as the case where an entity represents a single node since there are relatively few layers in the simulation. Aggregating all the layers into a single entity is still a compelling argument for other reasons.

There are times in a simulation when different layers need to access certain common variables. For example, the upper layers of the simulation need to use the CPU when they are executing any instructions. Since CPU is a shared resource among these layers, a layer has to make sure that the CPU is free before executing any instructions. Hence the upper layers need to have access to common variables which will provide information about the state of the CPU. If these layers are kept as different entities in the simulation there is no elegant method for accessing shared variables. Global variables cannot be used in such a situation as they cause problems with concurrent access during parallel executions.

If the layers are kept as different entities, each layer also has to explicitly keep track of the entity name values for the upper and lower layers. These entity name values are needed for message passing among the various layers. For the parallel conservative runtime, each entity also needs to specify the source and destination set of communicating entities as well as lookahead values. Specifying lookahead for an entity can be a very cumbersome and difficult task. This creates additional work for the protocol developer who is basically interested in modeling a particular network protocol.

For these reasons, we decided to integrate the various GloMoSim layers into a single entity. Each entity encompasses all the layers of a simulation. Each layer is now implemented as three function calls by the protocol modeler. The developer has to provide an initialization function that will be called for each layer of each node at the beginning of the simulation. The next function call provided by the developer is automatically invoked when a particular layer of a particular node receives an incoming packet/event. Based on the contents of the message, the appropriate instructions will be executed. Function calls are also provided for a layer to send messages to its lower or upper layer in the simulation. At the end of the simulation, another developer provided function call is invoked. This can be used to collect any relevant statistics for that layer.

It might appear that the addition of node and layer aggregation would cause great difficulty for protocol developers who are only interested in developing the simulation model for their particular model. But this is not the case as we have created several layers of abstractions in GloMoSim. For the most part, the developer writes pure C code. The presence of the PARSEC runtime and interactions with the runtime are completely hidden from the user. In the experience of our own group, modelers prefer to work within our structured environment of node aggregation rather than the alternative.

4.3 Simple APIs

Simple APIs between every two neighboring models on protocol stacks is predefined to support their composition. These APIs specify parameter exchanges and services between neighboring layers. The simplicity of the APIs allows developers to model their protocols rapidly in an independent fashion. The APIs currently defined in GloMoSim are presented:

<u>Channel Layer – Radio Layer APIs:</u>
Data packet from Channel to Radio:
    Fields: *payload, packetSize*
    These fields refer to the actual data and size of data being received. They have similar meanings when used subsequently for the reception or transmission of packets.
Data packet from Radio to Channel:
    Fields: *payload, packetSize*

<u>Radio Layer – MAC Layer APIs:</u>
Data packet from Radio to MAC:
    Fields: *payload, packetSize*
Data packet from MAC to Radio:
    Fields: *payload, packetSize*
Request Channel Status from MAC to Radio:
    Fields: (none)
    This message is used by the MAC layer to request information about the current channel status.
Report Channel Status from Radio to MAC:
    Fields: *status, flag*
    This message is used by the radio layer to return the current status of the channel as well as the method by which the information is being reported (passively or actively based on the request message sent by the MAC layer).

<u>MAC Layer – Network Layer APIs:</u>
Data packet from MAC to Network:
    Fields: *payload, packetSize, sourceId*
    The *sourceId* refers to the previous hop from which the packet arrived.
Data packet from Network to MAC:
    Fields: *payload, packetSize, destId*
    The *destId* refers to the next hop where the packet will travel.

<u>Network Layer – Transport Layer APIs:</u>
Data packet from Transport to Network:
    Fields: *payload, packetSize*
    The IP header should be a part of the packet that is sent from the transport to the network layer.

Data packet from Network to Transport:

Fields: *payload, packetSize, sourceId*

The *sourceId* refers to the original source where the packet originated. For the packet sent from the network to the transport layer, the IP header is no longer a part of the packet.

<u>Network Layer – Application Layer APIs:</u>

Data packet from Network to Application:

Fields: *payload, packetSize, sourceId*

Data packet from Application to Network:

Fields: *payload, packetSize*

These APIs, which are similar to the APIs used between the network and transport layers, are used for communication between routing daemons (such as OSPF) that are running at the application layer and need to communicate directly with the network layer.

<u>UDP Transport Layer – Application Layer APIs:</u>

Data packet from UDP to Application:

Fields: *payload, packetSize, sourceAddr, sourcePort, destAddr, destPort*

Data packet from Application to UDP:

Fields: *payload, packetSize, sourceAddr, sourcePort, destAddr, destPort*

In these APIs, the *sourceAddr* and *sourcePort* refer to the source address and port number where the packet originates. The *destAddr* and *destPort* refer to the destination address and port number where the packet is going.

<u>TCP Transport Layer – Application Layer APIs:</u>

Open Listen Socket from Application to TCP:

Fields: *appType, localPort*

This API is used by an application type (such as telnet server) to open a listen connection on the given port number.

Connection Open from Application to TCP:

Fields: *appType, localPort, remoteAddr, remotePort*

This API is used by an application to inform TCP to try to setup a connection from the given local port number to the given remote address and port number.

Data packet to send from Application to TCP:

Fields: *payload, packetSize, connectionId*

This API is used by an application to send a packet using on the given connectionId.

Connection Close from Application to TCP:

Fields: *connectionId*

This API is used by an application to close a particular connection.

Listen Socket Open Result from TCP to Application:

Fields: *localPort, connectionId*

This API is used by TCP to inform the application about the result of trying to open a listen connection.

Connection Open Result from TCP to Application:

Fields: *type, localPort, remoteAddr, remotePort, connectionId*

This API is used by TCP to inform the application about any connection that has been opened to a remote address and port number and the associated connection id. The connection type can be passive or active.

Data Sent Result from TCP to Application:

Fields: *connectionId, packetSize*

This API is used by TCP to inform the application about the number of bytes that could be sent due to the data sent request generated by the application for TCP.

Data Received from TCP to Application:

Fields: connectionId, payload, packetSize

This API is used by TCP to inform the application about any data that has been received on a connection.

Connection Close Result from TCP to Application:

Fields: *type, connectionId*

This API is used by TCP to inform the application about the connection getting closed as well as the type of connection close (passive or active).

5. Scalability of GloMoSim

The node aggregation technique gives significant benefits to the simulation performance. As each entity needs to examine packet receptions only for the nodes located in the region it is simulating, using many partitions reduce the total search space for packet delivery. In Fig. 2, for instance, if a packet sent by a node located in Partition (0, 0) cannot reach the border of the partition, no message needs to be sent to the other partitions. Therefore, the other partitions do not have to examine the reception of the packet, which reduce the region to be examined for the packet by a factor of four compared to using single partition. Fig. 3 shows the impact of multiple partitions for the models with 2500 and 5000 wireless nodes. Both simulation models consist of wireless nodes running CSMA at the MAC layer, each of which is randomly placed in 2000 x 2000m free space region. As seen in Fig. 3, the executions for both models become faster as the number of partitions increases. The effect of multiple partitions is larger for the model with 5000 nodes as the reduction in the execution time is related to the number of wireless nodes to be examined for each radio transmission.

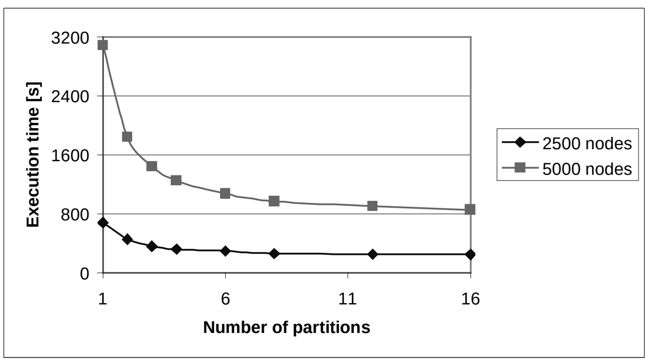

Fig. 3: Execution times against the number of partitions.

GloMoSim is aimed at simulating models that may contain as many as 100,000 mobile nodes with a reasonable execution time. GloMoSim has already been used to simulate 10,000 nodes up to the MAC layer using parallel execution of the model on shared memory architectures. Fig. 4 indicates parallel performance of GloMoSim on a Sun SPARCserver 1000. Speedup rates are calculated based on the sequential execution of each model. The same configuration as the experiments on multiple partitions is used with different number of wireless nodes. Twelve partitions are used for all the executions to balance the workload of each processor. As shown in Fig. 4, GloMoSim achieved better parallel performance for models with higher number of wireless nodes because more activities occur concurrently in those models, which increase the parallelism of models.

Users, especially those who need to simulate large-scale models can benefit from this parallel simulation capability of GloMoSim. Fig. 5 shows increases of execution times against the number of mobile nodes in the model. With the same number of processors, the execution time increases dramatically as the number of mobile nodes in the model increases. However, the execution time with 6 processors for the 10000 node model is shorter than the sequential execution for the 5000 node model. This implies that the user can run the simulation for a model consisting of twice the number of mobile node in the same amount of time with 6 processors.

Parallel simulation requires synchronization of simulation clock among multiple processors. PARSEC simulation environment provides four variations of conservative protocols and an optimistic protocol for the synchronization. The current GloMoSim kernel has parallel execution directives for conservative protocols and will be capable of executing models using optimistic protocols in future. The experiments done in this section used the null message protocol, which is one of the most basic conservative protocols widely used for many applications.

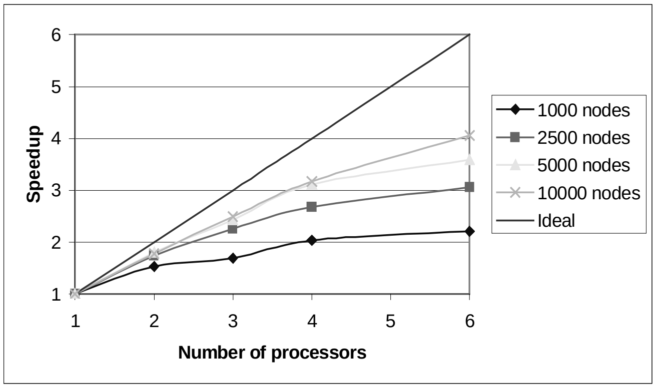

Fig 4: Parallel performance with different number of nodes.

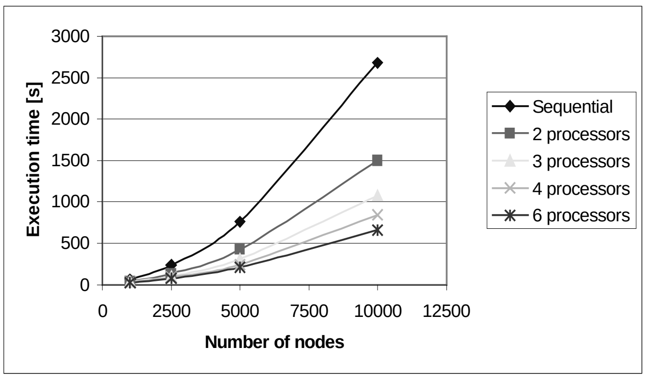

Fig 5: Execution times with varying number of nodes.

6. Case Study: Performance of TCP over MAC layer protocols.

The GloMoSim simulation library has also been used to investigate the performance of TCP and MAC layer in a wireless multi-hop network [6, 7]. Using the simulation library, new insights have been gained about the interactions between TCP and various MAC layer protocols, including CSMA, FAMA and 802.11. These MAC protocols were chosen because they provide an evolution of wireless medium access schemes, starting with carrier sensing (CSMA), then evolving to the utilization of RTS/CTS control frames (FAMA) and finally progressing to collision avoidance and acknowledgements (802.11). The interactions with various network topologies in a mobile environment where node movements are unpredictable have already been studied.

For example, we can consider the effects of the hidden terminal environment based on the topology shown in Fig. 6. Node 1 is in radio range of node 0 and node 2. Nodes 0 and anode 2 are not in the radio range of each other. A single TCP connection is set up from node 0 to node 1 and from node 2 to node 1. The results of the simulation experiments are shown in Fig. 7. As expected, CSMA exhibits capture behavior since both senders are unaware of each other's presence. The connection from node 2 to node 1 monopolizes the channel, with the node 0 to node 1 connection not being able to establish a connection at all. The capture behavior is explained by the interplay of CSMA timeouts with TCP timeouts, coupled with the presence of undetected link losses. When two connections compete for the same channel and one is "pushed back" by the timeouts, the binary exponential backoff nature of both CSMA and TCP timeouts make the situation progressively worse for the loser. FAMA and 802.11, on the other hand, are consistently fair. With FAMA and 802.11, both connections get equal share of the bandwidth with 802.11 attaining aggregate throughput slightly lower than that of FAMA, due to the additional control frame and collision avoidance scheme. The RTS/CTS mechanism and the less aggressive yield time of FAMA play an important role in the hidden terminal scenario. The RTS/CTS allows node 0 and node 2 to coordinate with one another although they are not within range of each other. This permits the idle source to determine when the other is expected to finish utilizing the wireless medium. The yield time deters the source from capturing the channel by permitting the competing source to transmit on the contention free wireless channel. Likewise, 802.11 succeeds in attaining fairness through the RTS/CTS and collision avoidance mechanism. Collision avoidance allows neighboring nodes to coordinate and share the channel. Aggregate throughputs of CSMA (1.8Mbps), FAMA (1.7Mbps) and 802.11 (1.5Mbps) are quite good since the maximum theoretical throughput in a hidden terminal situation is 2Mbps (only one node can transmit at a time).

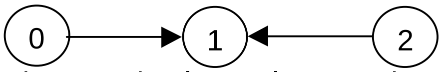

Fig. 6: A simple topology used to measure the effects of the hidden terminal environment.

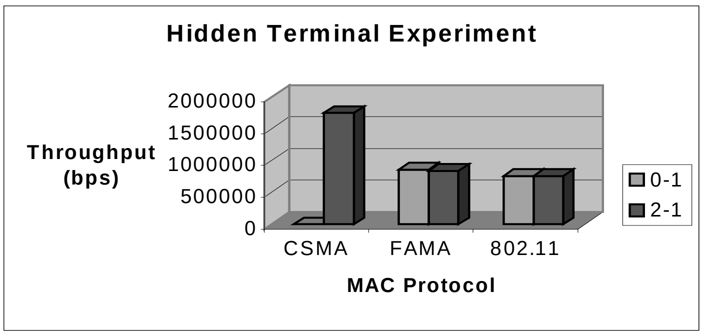

Fig. 7: Throughput (BPS) in Hidden Terminal Experiments.

A more realistic ad-hoc network environment with larger number of nodes is shown with the 81-node grid topology in Fig 8. The dimension of the grid is 100 X 100 meters. Each node is 10 meters apart from its horizontal and vertical neighbors. Furthermore, each node has a radio power range of 30 meters. Static routing is used, as nodes do not move for this particular experiment. FTP connections are established between node 18 to node 26, node 36 to node 44, node 54 to node 62, node 2 to node 74, node 4 to node 76, and node 6 to node 78. Simulation runs are executed for 400 simulated seconds.

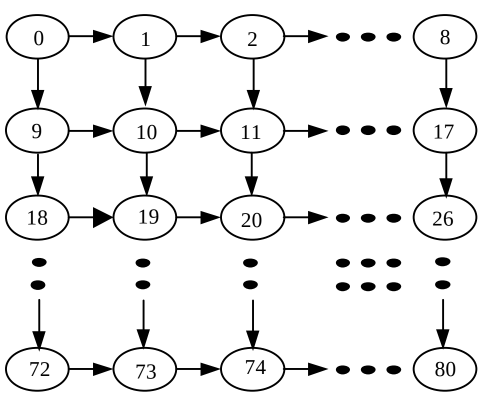

Fig. 8: Grid Topology.

The results for this experiment are shown in Fig. 9. Theoretical maximum throughput for an 8-hop connection with W equal to one packet size is 300Kbps. TCP performance under CSMA virtually collapses with maximum throughput of 2.9Kbps from the connection between node 2 to node 74. The average throughput is 1.6Kbps. The horrific performance is due to interference from neighbor FTP connections as well as interference from the three cross traffic connections. Particularly damaging to multi-hop FTP connections in CSMA is the high loss rate on the links and the lack of link loss recovery. Next, we shift our attention to FAMA. FAMA throughput is much higher than CSMA, with average throughput of 59.7Kbps. The improvement of FAMA is credited to the RTS/CTS control frames and the fair yield time. Therefore, FAMA is better protected against hidden terminal losses. Minimum and maximum throughput is 36.6Kpbs and 76.4Kbps, respectively. Compare to the theoretical maximum of 300Kbps, FAMA is still inadequate. With 802.11, the average throughput is 104Kbps. Minimum and maximum are 7.5Kbps and 222.3Kbps, respectively, which measure favorably to the theoretical maximum. The improved throughput is the direct result of the link level ACKs, which help achieve good aggregate throughput in a high loss rate environment. However, capture is not alleviated.

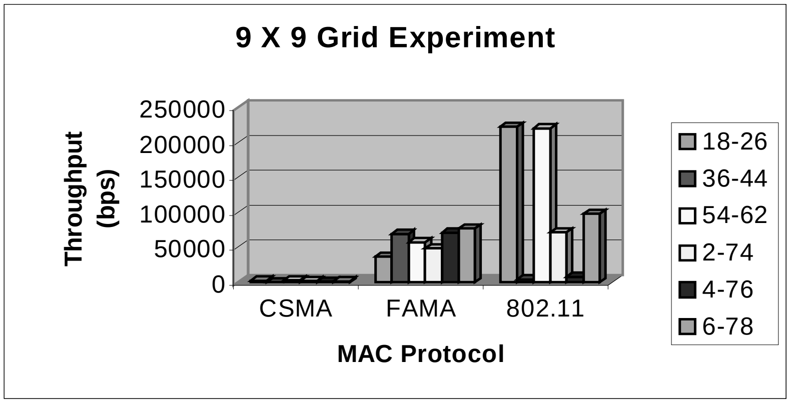

Fig. 9: Throughput (BPS), 9 X 9 Grid Experiments.

7. Case Study 2: Simulation of a Replicated File System

Optimistically replicated file systems have been suggested as an effective method to provide users access to shared files, even when they are temporarily disconnected [13]. Such file systems work on the following principles: replicas, or copies, of a file are stored on multiple computers. Using optimistic replication, a user's read or write operations are directed to the local replica. This can lead to file consistency problems when concurrent updates are made to multiple replicas of a given file. A process called reconciliation, which involves two replicas on separate computers, addresses such inconsistencies. During reconciliation, replicas are compared and updates are propagated between the two computers. When a server generates a reconciliation request it will also specify the target server based on the reconciliation topology, which is distinct from the physical topology in which the servers may be organized. Examples of reconciliation topology include tree, ring, and star topologies.

The design of a scalable replicated file system involves a number of considerations. At the fundamental level, it is necessary to identify the costs paid due to greater availability and reliability; effects of file access patterns on performance; impact of data exchange method, data propagation topology, data synchronization interval, and lastly how all these metrics change as the system is scaled up to hundreds of replicas. Some of the common metrics that have been proposed to evaluate the quality of service offered by a replicated file system include the stale read and write rate as well as cost measurements of CPU and network bandwidth usage. A stale read or write occurs when a user reads or writes out-of-date data [14]. A simulation model [14] for an optimistically replicated file system called Rumor [12] was initially developed using the PARSEC simulation environment by researchers in the OS group to investigate design alternatives. This model used an abstract model of the network as the transmission medium was modeled as a FIFO server.

There is also significant interest in using replicated file services in wireless environments for both military and civilian applications. The communication stack in a wireless ad hoc network is extremely complex. Furthermore, with the large number of protocols designed for the wireless communication at various layers in the protocol stack, there are several possible ways of configuring the communication stack. Most protocols have configurable parameters further increasing the number of possible network configurations. Thus unlike the case with wired networks, where a reasonable number of validated abstract models have been developed for different scenarios, an abstract network model cannot adequately represent the inherent complexities of a wireless communication stack. To obtain accurate simulation results for data replication services, it is imperative to model in detail all the layers in the communication stack from the physical to the application layer. The performance of the Rumor application is sensitive to the choice of lower level wireless protocols (e.g. at the MAC layer) and to the selection of appropriate parameters for a specific protocol (e.g., the size of the TCP retransmission window at the transport layer).

Fig. 10 shows the impact of MAC layer protocol choice on the average reconciliation time. In the absence of mobility, the average reconciliation time is almost independent of the specific MAC layer protocol. With the introduction of mobility, there are noticeable differences in the performance of the replication service under different MAC layer protocols. MACA performs the worst, as it does not sense the carrier when it needs to transmit data. If a particular node is receiving data and needs to transmit at the same time, the incoming packet is lost. It would be impossible to measure the impact of such low level network changes on the application unless we have the ability to run detailed simulations using the GloMoSim library.

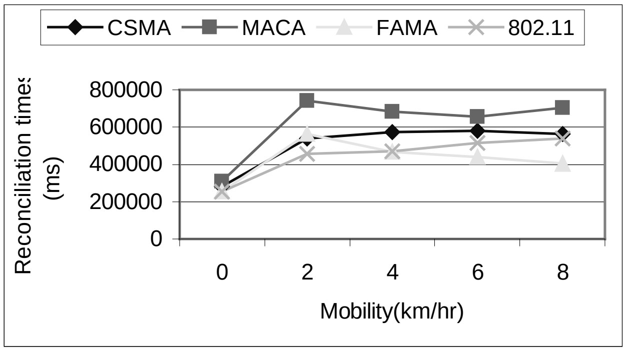

Fig. 10: Average reconciliation time as mobility speed is varied for various MAC layer protocols.

8. Conclusion

Detailed, high fidelity models of large networks represent a significant challenge for the networking community. As the military moves towards deploying a digital communication infrastructure, it is imperative that the performance of the communication devices be thoroughly studied prior to deployment to understand the limits of the network and its ability to handle diverse traffic under stringent operating conditions. This paper presented a simulation library called GloMoSim whose goal is to support accurate performance prediction of large-scale network models using parallel execution on a diverse set of parallel computers. The library has already been used to simulate networks with thousands of wireless nodes and provides a rich set of models for both existing and novel protocols at multiple layers of the protocol stack. It has been used to undertake numerous performance studies among alternative protocols both at UCLA and other organizations. It is available for download at http://pcl.cs.ucla.edu/projects/domains/glomosim.html

References

[1] H. Akhtar, "An Overview of Some Network Modeling, Simulation, &Performance Analysis Tools," Proceedings of 2nd IEEE Symposium on Computers and Communications, 97.

[2] R. Bagrodia, R. Meyer et al., "PARSEC: A Parallel Simulation Environment for Complex Systems," IEEE Computer, October 98.

[3] R. Bagrodia, Y. A. Chen, et al., "Parallel Simulation of a High-speed Wormhole Routing Network," Proceedings of 10th Workshop on Parallel and Distributed Simulations, PADS 96.

[4] S. Bhatt, R. Fujimoto, A. Ogieski, and K. Perumalla, "Parallel Simulation Techniques for Large-Scale Networks," IEEE Communication Magazine, August 98, pp. 42-47.

[5] J. G. Clearly, J. J. Tsai, "Conservative Parallel Simulation of ATM Networks," Proceedings of 10th Workshop on Parallel and Distributed Simulation, PADS 96.

[6] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang, "TCP over Wireless Multihop Protocols: Simulation and Experiments," Proceedings of IEEE ICC 99.

[7] M. Gerla, K. Tang, and R. Bagrodia, "TCP Performance in Wireless Multihop Networks," Proceedings of IEEE WMCSA 99.

[8] T. Holvoet and P. Verbbaeten, "Using Agents for Simulating and Implementing Petri nets," Proceedings of 11th Workshop on Parallel and Distributed Simulation, PADS 97.

[9] P. Martini, M. Rumekasten, J. Tolle, “Tolerant Synchronization for Distributed Simulations of Interconnected Computer Networks,” Proceedings of 11th Workshop on Parallel and Distributed Simulation, PADS 97.
[10] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publishers 1996.
[11] S. McCanne and S. Floyd, UCB/LBNL/VINT Network Simulator – NS (version 2), http://www-mash.cs.berkeley.edu/ns/
[12] P. Reiher, G. Popek, M. Gunter, J. Salomone, D. Ratner, “Peer-to-Peer Reconciliation Based Replication for Mobile Computers,” European Conference on Object Oriented Programming, Second Workshop on Mobility and Replication, June 96.
[13] M. Satyanarayanan, “Coda: A Highly Available File System for a Disconnected Workstation Environment,” Proceedings of the 2nd Workshop on Workstation Operating Systems, September 89.
[14] A. Wang, P. Reiher, R. Bagrodia, and G. Popek, “A Simulation Evaluation of Optimistic Replicated Filing in a Mobile Environment,” Proceedings of the 18th IEEE International Performance, Computing, and Communications Conference, February 99.