Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

A HIERARCHICAL SIMULATION ENVIRONMENT FOR MOBILE WIRELESS NETWORKS

R. Bagrodia
M. Gerla
L. Kleinrock
J. Short
T.-C. Tsai

Computer Science Department
University of California at Los Angeles
Los Angeles, CA 90095

ABSTRACT

A hierarchical simulator has been designed for multimedia communication protocols in a wireless mobile environment. The hierarchical approach integrates performance evaluation of protocols with their implementation. The approach supports scalability studies of the protocols in an efficient manner using coarse grain models that abstract implementation details of the protocol and its execution environment by a few key parameters. Fine grain, low-level models that capture implementation details are used for detailed evaluation of small networks and for automatic implementation on radio platforms. The design, evaluation, and implementation cycle is closed by feeding the measurements from the implementation back into the model to improve its accuracy. The paper describes the use of the environment in the evaluation and implementation of a cluster-based multihop protocol for multimedia traffic.

1 INTRODUCTION

The exploding demand for nomadic computing and communication has led to a significant need for infrastructure to support the mobile user. Applications that could greatly benefit from such support include disaster relief operations, next generation of PCS (personal communication systems) services, and tactical mobile units in the military. A relatively high quality of support for integrated voice, video and data traffic is becoming available over a wireline network; however integrated support for these services over a wireless network is still in its infancy. The need for such an infrastructure underlies the goals of the UCLA WAMIS project. A central component of this project is to develop wireless multimedia instant infrastructure that is capable of establishing instant network infrastructure, of maintaining communication, perhaps via topological reconfiguration, in the presence of node mobility and failures.

As is well known, such protocols are complex to design, evaluate and implement. Their performance depends on a combination of factors that include multimedia traffic patterns, mobility models, application objectives (e.g., maximize throughput, minimize noise/loss), processor characteristics (cpu speed, load), and radio characteristics (bandwidth, power). Before investing significant resources in their implementation, it is generally desirable to model the protocols such that their expected behavior can be evaluated in the projected operational environment. Evaluation of the protocol as a function of these diverse parameters using only analytical models is hard; a detailed performance study is often impossible due to analytical intractability. An alternative to performance prediction is to build and deploy protocol implementations and then test them in actual operation. However, this approach may preclude tests of the protocols over a wide parameter space.

This paper describes an environment to support rapid prototyping of mobile, multimedia protocols, a comprehensive evaluation of their performance over a large parameter space and over large numbers of mobile units, and eventual implementation into operational network systems. The next section describes the networking and node architectures of the physical system that is being simulated. Section 3 describes the modeling environment; Section 4 describes its use to evaluate and implement a cluster-based instant infrastructure. Section 5 discusses related work and Section 6 is the conclusion.

2 PHYSICAL SYSTEM ARCHITECTURE

The design of the multimedia instant infrastructure relies on innovations in the internetworking and subnetworking layers as well as in the interfaces linking the network to the operating system and to the radio. A network operating system for mobile multi-

media nodes is being designed at UCLA as an enhancement to the public-domain KA9Q NOS kernel by Phil Karn. In order to provide a flexible environment for experimentation with multiple radios, modules that implement various network control functions are written in C and supported by the operating system to isolate the architecture and implementation details from the network algorithms. The network control functions comprise numerous software components which can be used to support self-configuring, multihop, multimedia networking. These components can be broken down into end-to-end transport control, internetworking and connectivity control, subnetwork control, and finally link and mobility control. An environment to model each of these components has been designed (Short, et al. 1995) and is described briefly in the next section. Many of the preceding components are needed for the cluster-based communication protocols that are used in the instant infrastructure.

In the packet radio network, each node contains an identical transceiver which can either transmit or receive at any one time. In the spread spectrum code division environment, the receiver should be set to the same code as the designated transmitter before or at the time it transmits. In order to increase the control efficiency and to reduce power interference from neighboring nodes (even hidden nodes), we have adopted a "clustering" architectures (Baker, et al. 1984). The idea is to group some neighboring nodes together into a *cluster*. Each cluster is coordinated by the *clusterhead* (CH) which can directly communicate with all the member nodes in the cluster. The role of CH serves as a regional broadcast node, and as a local coordinator to enhance channel throughput.

Following the clustering approach, we can easily enforce time-division (and even code division) scheduling within the cluster, and can facilitate spatial reuse of time slots and codes across clusters. The store-and-forward multi-hop sessions across different clusters will be carried through by *gateways* (GW), which are nodes that can directly communicate with 2 or more CHs. The proposed clustering architecture can be viewed as a hybrid between traditional packet radio and cellular networks. The clusterhead approach and the support of guaranteed bandwidth connections reminds us of cellular networks. The datagram traffic support, the multihopping and the dynamic reconfigurability follow the tradition of earlier packet radio projects. A fast reservation virtual circuit (VC) scheme is developed for real time traffic in order to guarantee bandwidth. Datagram traffic uses the S-ALOHA scheme with CH acknowledgments. Details of the network protocols and the channel access scheme are described in the paper by Gerla and Tsai (1995).

3 SIMULATION ENVIRONMENT

A general purpose parallel environment is being developed at UCLA for the simulation and prototyping of protocols for mobile computing. Our aim is to decompose the protocols and their execution environment to allow maximum flexibility in experimentation with alternative implementations of a given functionality (e.g., radio characteristics) as well as to support a "plug and play" capability that generates composite prototypes constructed from pieces that model system components at widely differing levels of detail. The simulator is being built on an existing process-oriented, parallel simulation language called Maisie (Bagrodia and Liao 1994). This section describes the simulation environment and the methodology.

3.1 Maisie Language

A Maisie program is a collection of entity definitions and C functions. An entity definition (or an entity type) describes a class of objects. An entity instance, henceforth referred to simply as an entity, represents a specific object in the physical system and may be created and destroyed dynamically. Entities communicate with each other using timestamped messages. Every entity is associated with a unique message-buffer. The language provides an asynchronous send primitive to deposit a message with a current or future timestamp in the message buffer of a destination entity. Blocking and non-blocking receive primitives are also provided by the language to allow an entity to remove messages from its message buffer. The receive construct can be used to remove selective messages from the buffer such that a message is removed only when it is ready to be processed by the destination entity. Appropriate use of selective receives help in the generation of concise model descriptions.

Event scheduling constructs in Maisie are integrated with the send and receive primitives. Thus transmission delays in a physical network can be modeled simply by incrementing the timestamp on the message when it is sent from the source to the destination node. A receive statement can optionally specify a timeout interval, where time is measured using the simulation clock; such a statement may be used to simulate the passage of time corresponding to activities like servicing of a job in a model, or transmission of a message in a network.

A program written in Maisie is independent of

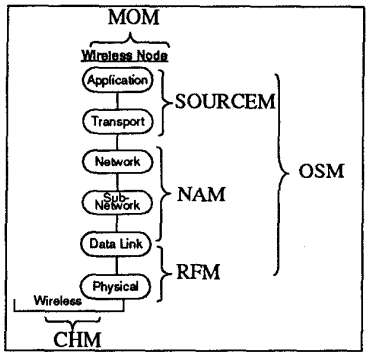

Figure 1: Networking Layers and their Models

any synchronization algorithm. When it is compiled, the analyst can indicate the specific simulation algorithm that is to be used to synchronize execution of the model: sequential, parallel conservative, or parallel optimistic. The compiler generates the appropriate code to interface the model with the corresponding run-time system: a splay-tree based implementation of the global event-list algorithm for the sequential implementation, a null-message or conditional event implementation of the parallel conservative synchronization algorithm (Jha and Bagrodia 1993), or a space-time implementation of the optimistic synchronization algorithm (Bagrodia, Chandy, and Liao 1991). Maisie has been implemented on a variety of sequential and parallel architectures and has been used to model diverse applications from parallel software to VLSI designs.

3.2 Integrated Simulation and Implementation Environment

An integrated environment for the simulation and implementation of network protocols has been built using Maisie (Short, et al. 1995). Fig. 1 displays the relationship between the modeling components and the OSI reference model. The components have been enhanced in various ways to support wireless operations and to integrate wireless and wireline operations. The primary components of the environment are:

- Operating System Models (OSM)
- Application Traffic Models (SOURCEM)
- Network Algorithm Models (NAM)
- Wireless Radio Models (RFM)
- Channel Models (CHM)
- Mobility Models (MOM)

The OSM simulates the relevant portion of the operating system kernel that is involved in interfacing with the application (e.g., delivery of incoming messages) or with the network (e.g., transmission of a remote message). The SOURCEM components can be broken down into the source and destination streams (e.g.: hard disk, keyboard, camera, screen, microphone, or speaker) corresponding to the voice, video and data traffic, the control of these streams via the application, and the transport mechanism (e.g.: TCP, UDP, or Virtual Circuits) which the application chooses to use.

The NAM components are broken down into internetwork models such as IP, subnetwork control such as clustering, and mobility control such as power control, logical link control, and media access control. The MOM components are responsible for movement patterns of the nodes such as the speed in which the nodes move and their motion pattern such as brownian random motion or drift. The CHM components are responsible for the transmission media including the range in which two nodes are able to communicate with each other, and environmental effects such as multi-path fading, shadowing, and interference. The RFM components are responsible for the physical layer modeling of the radio frequency modem and includes the raw channel bandwidth, modulation techniques, and acquisition delays. In the remainder of this section, we describe the facilities for the simulation of clustering based wireless network architecture for instant infrastructure networking.

The infrastructure exists at the subnetwork level in order to control the wireless multihop network transparently for applications and TCP/IP. As shown in Fig. 2, the clustering subnetwork control contains a synchronization module to either adapt to existing clusters, if available, or to create new clusters based upon the clusterhead election algorithm. The TDMA module is responsible for queueing up packets from the IP level in order to provide guaranteed bandwidth delivery of data. The multihop routing algorithm based upon Bellman-Ford's minimum hop is done at the subnetwork level to provide the wireless multihop functionality. The lower layers are responsible for intelligently setting parameters of the radio and adjusting to measurements such as SIR. The CDMA module is responsible for assigning one code per cluster. The LLC module is responsible for providing piggyback and explicit link level acknowledgment since CDMA is used between hops (between different clusters) so passive acknowledgment schemes can not be used.

The distributed power control module, which is based upon work by Chen, Bambos, and Pottie (1994), adjusts the transmit power based upon the

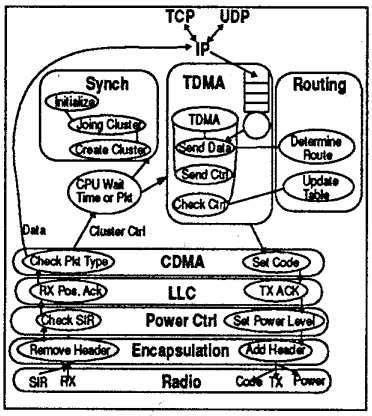

Figure 2: Instant Infrastructure Algorithms

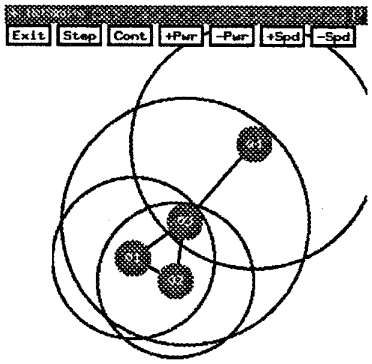

Figure 3: Simulation Environment Control Panel

SIR measured at receiving nodes to minimizes transmit power necessary and reduce CDMA interference and reduce the near-far effect. Various radio parameters such as bandwidth, acquisition time, and error rates were used. As seen in Fig. 3, the simulation environment is able to control various environmental and node parameters such as location, speed of mobility, and transmit power of radio. Based upon these parameters, the channel model determines and transmits packets between appropriate nodes. The simulation environment provides several unique abilities including *the ability of each node to define its own virtual operating environment where parameters such as transmit power, code, and mobility pattern can be set interactively while the simulation is running.*

To facilitate the transition of simulation models into implementations, the interfaces of the modeling environment components must be designed with care. The OSM is responsible for process scheduling, communication, and synchronization as well as for interactions between the applications and the communication device's packet driver. By isolating the interface among these modules, it is possible to interface the packet driver on the node with various network algorithms and operating systems like the KA9Q NOS kernel, the Mach kernel, and Windows. Given a compatible design for these interfaces, it is possible to automatically and transparently migrate the detailed simulation models to operational implementations of network algorithms with different operating system kernels and communication devices. This is a potentially significant achievement as it would provide a direct path for the network model to evolve into operational software.

The rapid prototyping ability of the environment was demonstrated by porting the synchronization, TDMA, and clusterhead election algorithm of the instant infrastructure models to a real system. The refined models used in the simulation study described in the next section were ported, without any redesign or programming, to a set of four 486 laptops running WAMISNOS using two different radios: the UCLA Direct Sequence Spread Spectrum radio operating at 32Kbps (Jain, et al. 1995), and the Proxim RangeLAN2 Frequency Hop Spread Spectrum radio operating at 1.6Mbps.

3.3 Hierarchical Modeling

The integrated design, evaluation, and implementation of the instant infrastructure protocols uses a multi-level modeling methodology where initial abstract or *coarse* models are refined into detailed *fine* models and eventually into protocol implementations. The coarse simulation model is used to develop and test network algorithms in a large, multicluster network setting. Various routing policies and virtual circuit maintenance issues are investigated in the presence of node mobility. In order to obtain run time efficiency and to support interactive protocol design, many implementation details of the protocol are omitted. At this level, the performance measures of interest are the ones related to efficient support of the target multimedia applications, namely: delay (both average and variance), throughput, packet loss, out of sequence packet delivery, and network connectivity (both physical and logical). These performance measures depend not only on network algorithms, traffic pattern and physical network parameters, but are also impacted by the hardware and software characteristics of the mobile node which are typically omitted to keep the models simple and the run-times reasonable.

The refined simulation model captures the implementation details of the protocol and its execution environment that are omitted in the coarser model. Characteristics of the execution environment that are modeled include many of the components from the NOS model described in the previous section including the node OS architecture and CPU load caused by the various applications from the OSM and SOURCEM models, and the impact of CPU load on TDM frame synchronization from the RFM and CHM models. Thus, the fine grain simulator must support an enriched set of performance measures, including CPU loading, synchronization timing, detailed radio modem behavior etc. This model is developed using as inputs the measurements from the actual radio node prototype on which the protocols are being implemented.

The presence of the detailed model characteristics allows the refined models to estimate performance measures not derivable in the coarse grain simulator (e.g., TDM frame synchronization loss and consequent packet loss due to varying application loads). Once these parameters have been estimated in the fine-grain model, they are subsequently introduced as macroscopic parameters in the coarse grain simulator as illustrated in the experimental study reported in the next section. For example, link failure rate (due to synchronization loss) can be defined as coarse grain simulator input. Another function of the detailed simulator is to serve as an intermediate step to protocol software implementation in the radio platform. In fact, as discussed in Section 3.2, the faithful reproduction of the WAMIS operating system environment in the simulator makes it possible to port and run the simulator on WAMISNOS with minimal alterations.

The hierarchical approach is well suited to the interactive design and implementation of the network protocols. It permits us to carry out conceptual algorithm design on large network topologies in an efficient, interactive manner using the coarse grain simulator. In this high level design process, the implementation details are hidden behind a few key parameters which were derived from the fine grain simulator. In turn, the high level network algorithms tested in the coarse simulator are implemented in the fine grain simulator to verify efficiency of operation in the radio node platform. Eventually, the fine grain simulator will serve as a conduit to software implementation on the radios. This design and evaluation cycle is closed by feeding back the measurement results to the simulation model to further improve its accuracy for predicting the performance of large-scale network models.

4 EXPERIMENTAL STUDY

This section describes a performance study that used the multi-level modeling methodology described in the previous section. In the sequel, we show an example of the interplay between the coarse and refined models. More specifically, we show how experiments with the refined models can be used to evaluate link failure rate due to loss of TDM frame synchronization. The latter can be caused by three independent events: (a) CPU overload; (b) control packet loss due to channel propagation effects or interference; and (c) radio mobility. The coarse model can accept as input the link failure rate, but does not include enough detail to compute, for example, the CPU loading caused by the applications, and its effect on link failure rate.

We first describe the detailed experiments which were executed for a 4-node cluster to compute link failure rate due to each of the preceding factors. The first set of experiments studies the impact of CPU load on link failure rate. We recall that in the TDMA access control algorithm, when it is time for a node to transmit a packet, the TDMA process selects and transmits the next packet in the queue. If the CPU is heavily loaded, then the TDMA process will be delayed before it can gain access to the CPU to transmit the packet. To compensate for this delay, a "guard" time is added to the TDMA slot time. In order to study the effects of CPU loading for a given guard time, a model of the CPU scheduler is used, which simulates the effect of the CPU load on frame synch loss. When packets are delayed so that synchronization can no longer be maintained, then a link failure occurs. In Fig. 4, CPU load (assumed to be uniform

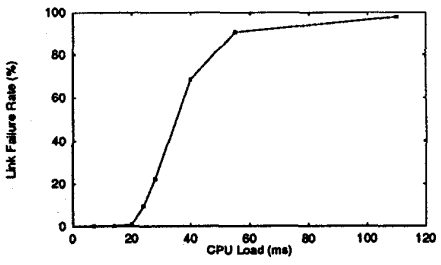

Figure 4: Link Failure Rate vs. CPU Load

for all nodes) is expressed in terms of job scheduling delay (in ms). The slot guard time is 20 ms, which was experimentally determined to be adequate for most applications. However, if the maximum CPU holding time by an application increases beyond 20 ms, the effect on the link failure rate can be quite

dramatic (as shown in Fig. 4). Based on these considerations, slot guard time should be set to the maximum CPU holding time. However, this is not possible in practice because a slot guard time increase contributes directly to channel overhead. A proper compromise must be found via simulation, using the approach here described.

Another set of experiments relates channel packet loss to link failure rate. Fig. 5 shows the impact of a packet loss in a channel on the failure rate of the corresponding link. Every 300ms a node transmits

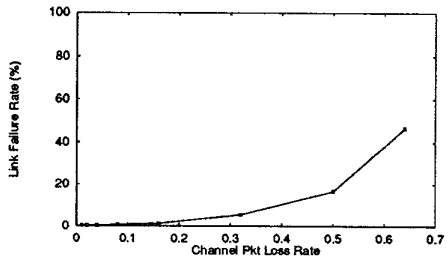

Figure 5: Link Failure Rate vs. Ch. Pkt Loss Rate

a control packet used for synchronization and topology creation. This is done during each node control slot period. With 4 nodes synchronized, each node sends out its control information every 1.2 seconds. This control packet can be lost for several reasons including radio channel factors such as background interference, multipath fading or shadowing which frequently cause bit errors, and packet collision (interference from another radio). Based upon Fig. 5 results, we see that the logical topology synchronization is fairly stable for packet loss rates up to 50%. Experimental results reported in Jain, et al. (1995), show that packet loss rates as high as 16% have been observed, with corresponding logical link failure around 1.2%. Note, however, that even though the logical topology is accurate 98.8% of the time, 16% of the data packet transmission still fail.

A final set of fine grain experiments studies the impact of mobility and transmission range on link failures. In Fig. 6 we can examine the effects of various radio ranges (power levels) and node speeds on link failure rate. As the transmission range is increased, the link failure rate decreases. Conversely, as speed increases, failure rate increases.

Next, we show how frame synch loss due to CPU overload, control packet loss and mobility can be indirectly accounted for in the coarse grain simulator using link failure rate as input parameter.

The coarse grain experiments were based on 20

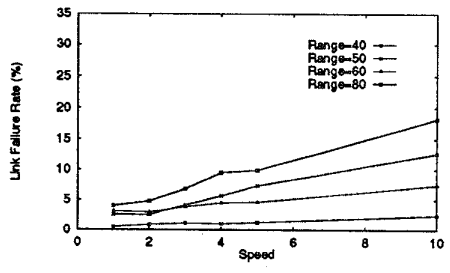

Figure 6: Link Failure Rate vs. Mobility

nodes (radios) placed randomly inside a 100×100 square. During the simulation, in every time frame, a node may randomly move X units in each direction or stay where it is. If a node tries to move beyond the boundary, its position is set at the boundary ("sticky boundary" model). Two nodes can communicate directly with each other if they are within hearing range, which can be modified by adjusting the power level.

The following parameters were used in this study: CPU load = 23 ms, mobility(X) = 5 units/frame, control packet loss rate = 16%, transmission range varying between 20 and 30 units (note that to make 4 node and 20 node experiments consistent the range must be scaled down by a factor $\sqrt{20/4}$ when moving from 4 to 20 nodes, since in both cases the nodes are distributed over a 100×100 area, and yet must have the same average number of neighbors). From the 4 node results presented in Figs. 4 through 6 we derive the "equivalent" link failure rate for the 20 node experiment corresponding to the above input parameters. For example, for a transmission range = 18 units the link failure rate (due to the cumulative contribution of CPU load, packet loss, mobility) results to be 21.2%. We then input this link failure value in the coarse grain simulator to compute the "logical" connectivity in the 20 node network. *Connectivity* for the network is computed as the average connectivity of each node, where the connectivity of a node is defined as the fraction of nodes that it can reach via single or multiple hops. We distinguish between *logical* and *physical* connectivity. In the presence of synchronization loss, a link may physically be up but will logically be considered to be down as it cannot be used to transmit data packets. In other words, logical connectivity is defined as the fraction of node pairs communicating with each other, accounting for the link failure rate due to frame synch loss.

The consistency of the large, coarse grain models and small, fine grain models can be systematically

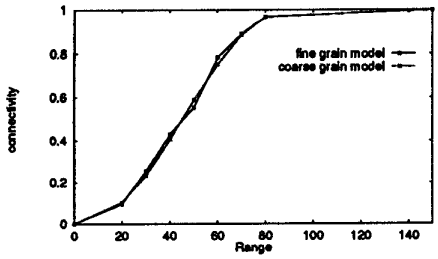

Figure 7: Connectivity vs. Range (4 nodes)

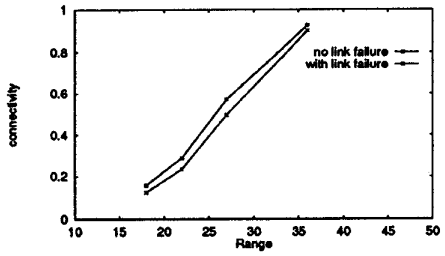

Figure 8: Connectivity vs. Range (20 nodes)

verified by running the coarse model for a small network configuration. Fig. 7 shows logical connectivity versus transmission range in a four node, mobile network configuration. The experiment was carried out with both fine and coarse simulator. There is good agreement between the two sets of results, thus verifying the consistency of the assumption.

Fig. 8 compares physical connectivity (i.e., without accounting for link failures) and logical connectivity (with link failures) for the above mentioned 20 node experiment using the coarse simulator. We note that the impact of link failure (i.e., synch loss) on connectivity is quite substantial, especially in the critical region between 20 and 40. This impact could be captured only through proper interaction of fine and coarse grain simulation.

5 RELATED WORK

A number of commercial tools are available for analysis and evaluation of network protocols including BoNeS, COMNET (CACI), and OPNET (Mil3). All existing tools use sequential model execution and none provide a path towards eventual protocol implementation. We specifically examine the limitations of OPNET for prototyping mobile protocols. Most other systems suffer from similar drawbacks.

OPNET has a number of drawbacks that make it unsuitable for mobile wireless: limited support for node mobility patterns (for instance specifying random node movements), restrictive channel models, rigid modeling methodology which forces the user to specify the models at a much greater level of detail than may be appropriate or feasible for the analyst (the three level methodology must be followed), and long execution times for even moderate sized models. The last and significant problem is that there is no path towards implementation.

Although many university projects are aimed at simulation and analysis of networking protocols, relatively few address integrated network simulation and prototyping (Abbott and Peterson 1992; Montz, et al. 1994). For instance, the approach used in the x-kernel and Scout projects at the University of Arizona is to develop an operating system which can support the implementation of networking protocols (possibly automatically). A simulation environment was developed specially for the x-kernel (Hutchinson and Peterson 1991) which successfully analyzed the performance of the new protocol based upon various simulation model parameters.

6 CONCLUSION AND FUTURE WORK

An environment for hierarchical modeling of multimedia protocols in a wireless mobile environment has been developed at UCLA. This environment supports the iterative transition of simulation models into protocol implementations on operational radio platforms using a multi-level approach. The environment has been designed using the Maisie simulation language. Although Maisie models can be executed on various parallel architectures, the study reported in this paper only used the sequential executions. Parallel implementations of the model are currently under investigation. Our eventual goal is to support an integrated modeling environment, where detailed models of a sub-network are executed *in parallel* with the abstract models, to interactively evaluate protocol performance for large networks over a wide parameter space.

ACKNOWLEDGMENTS

This research was supported by the U. S. Dept. of Justice/FBI, ARPA/CSTO under contract J-FBI-93-112, and by the Advanced Research Projects Agency, ARPA/CSTO, under Contract DABT-63-94-C-0080 "Transparent Virtual Mobile Environment."

REFERENCES

Abbott, M. and L. Peterson. 1992. A language-based approach to protocol implementation. Technical Report No. 92-2, Department of Computer Science, University of Arizona.

Bagrodia, R., K. M. Chandy, and W-T. Liao. 1991. A unifying framework for distributed simulations. *ACM Trans. on Modeling and Computer Simulation* 1(4):348-385.

Bagrodia, R. and W-T. Liao. 1994. Maisie: A language for design of efficient discrete-event simulations. *IEEE Transactions on Software Engineering* 20(4):225-238.

Baker, D. J., A. Ephremides, and J. A. Flynn. 1984. The design and simulation of a mobile radio network with distributed control. *IEEE JSAC* SAC-2(1):226-237.

Chen, S., N. Bambos, and G. Pottie. 1994. On distributed power control for radio networks. In *IEEE International Conference on Communication*, New Orleans, LA, 1281-1285.

Gerla, M. and J. Tsai. 1995. Multicluster mobile multimedia network. *ACM - Beltzer Journal of Wireless Networks*, August (To appear).

Hutchinson, N. C. and L. L. Peterson. 1991. The x-Kernel: An architecture for implementing network protocols. *IEEE Transactions on Software Engineering* 17(1):64-76.

Jain, R., J. Short, S. Nazareth, L. Kleinrock, and J. Villasenor. 1995. PC-notebook based mobile networking: Algorithms, architectures, and implementation. In *Proceedings of the 1995 IEEE International Conference on Communications*, Seattle, WA, 771-777.

Jha, V. and R. Bagrodia. 1993. Transparent implementation of conservative algorithms in parallel simulation languages. In *Proceedings of the 1993 Winter Simulation Conference*, December, 677-686.

Montz, A., D. Mosberger, S. W. O'Malley, L. L. Peterson, T. A. Proebsting, and J. H. Hartman. 1994. Scout: A communications-oriented operating system. Technical Report No. 94-20, Department of Computer Science, University of Arizona.

Short, J., R. Bagrodia, and L. Kleinrock. 1995. Mobile wireless network system simulation. In *Proceedings of the 1995 ACM International Conference on Mobile Computing and Networking*, Berkeley, CA, November, 195-205 (To appear).

AUTHOR BIOGRAPHIES

RAJIVE L. BAGRODIA received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Bombay in 1981 and the M.A. and Ph.D. degrees in Computer Science from the University of Texas at Austin in 1983 and 1987, respectively. He is currently an Associate Professor in the Computer Science Dept. at UCLA. His research interests include parallel languages, parallel simulation, distributed algorithms, and software design methodologies. He was selected as a 1991 Presidential Young Investigator by NSF.

MARIO GERLA was born in Milan, Italy. He received a graduate degree in engineering from the Politecnico di Milano, in 1966, and the M.S. and Ph.D. degrees in engineering from UCLA in 1970 and 1973, respectively. He joined the Faculty of the UCLA Computer Science Department in 1977. His research interests cover the performance evaluation, design and control of distributed computer communication systems and high speed computer networks (ATM, Optical Networks and Wireless).

LEONARD KLEINROCK received the B.S. degree in Electrical Engineering from the City College of New York in 1957 and the M.S.E.E. and Ph.D.E.E. degrees from the Massachusetts Institute of Technology in 1959 and 1963, respectively. He is currently a Professor in the Computer Science Department at UCLA. His research interests focus on performance evaluation and design of many kinds of networks (e.g., packet switching networks, packet radio networks, local area networks, metropolitan area networks broadband and gigabit networks) and of parallel and distributed systems. Dr. Kleinrock is a member of the National Academy of Engineering, a Guggenheim Fellow, and an IEEE Fellow.

JOEL E. SHORT received the B.S. degree in Computer Science from California State University, Chico in 1992 and the M.S. degree in Computer Network Modeling and Analysis from University of California, Los Angeles in 1995. He is currently a Ph.D. candidate in the Computer Science Department at UCLA studying wireless and nomadic system simulation and implementation.

JACK TZU-CHIEH TSAI received his B.S. and M.S. degrees both in Electrical Engineering from the National Taiwan University in 1988, and from the University of Southern California in 1991, respectively. He is currently a Ph.D. candidate in Computer Science at UCLA. His research interests include network protocol design and implementation for wireless radio networks; performance evaluation for distributed computer communication network systems.