Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

PARALLEL SIMULATION ENVIRONMENT FOR MOBILE WIRELESS NETWORKS

Winston Liu
Ching-Chuan Chiang
Hsiao-Kuang Wu
Vikas Jha
Mario Gerla
Rajive Bagrodia

Computer Science Department
University of California, Los Angeles
Los Angeles, California 90095, U.S.A.

ABSTRACT

A parallel simulator has been designed for the evaluation of wireless, multihop, mobile networks. This paper describes the process of porting the simulator from a sequential to a parallel environment. Parallelization is critical in large radio networks, where the complexity of radio propagation models, channel access schemes and interference patterns makes sequential simulation very time consuming – in the order of several hours for 100 nodes experiments. With parallel execution, speedups of up to tenfold have been observed on a 16 processor SP/2, making large network simulations viable.

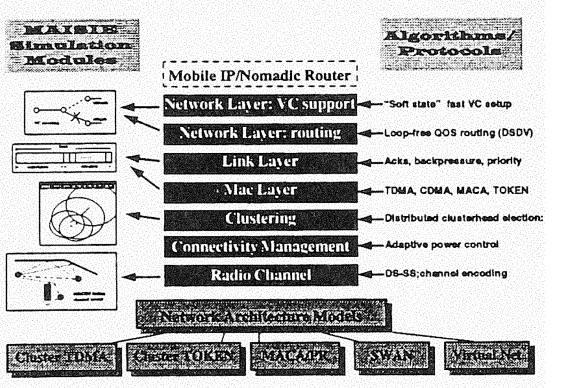

Figure 1: Wireless Network: Protocols and Models

1 INTRODUCTION

The rapid advancement in portable computing and wireless communications technology has led to significant research and investment to create an infrastructure to support mobile users. Such an infrastructure is beneficial to civilian applications such as disaster relief operations as well as military applications such as tactical mobile units. The need for such an infrastructure underlies the focus of the UCLA GloMo project. A major contribution of this project is the development of an instant wireless infrastructure which supports multimedia multi-hop wireless communications in presence of node mobility and failures. Protocols designed for this kind of environment are complex to evaluate analytically due to various factors such as complex channel access protocols, interference effects, traffic input patterns, node mobility , channel propagation properties, and radio characteristics. To alleviate the prohibitive cost of protocol implementation via trial and error, simulation offers a way to evaluate the protocol design before the actual implementation takes place. In simulating such an environment, the simulation execution time is usually directly proportional to the level of detail in the model and the size of the network that one wants to study. Given the complexity of the radio environment, sequential simulation of networks with thousands of nodes requires several days. To make the design more interactive, it is imperative to reduce run time without sacrificing the level of detail in the model. A promising solution is parallel simulation. This paper describes a parallel environment to support rapid prototyping of mobile, multimedia protocols. Sections 2 and 3 describe the wireless network architecture and the radio channel model which have been implemented in the parallel simulator. Section 4 reviews the Maisie parallel simulation environment and discusses the process of porting the model to a parallel architecture. Section 5 reports the results.

2 NETWORK ARCHITECTURE

The network architecture under study is a wireless, mobile, multihop architecture. Unlike cellular systems, there are no fixed base stations connected by a wireline network. The main motivation for mobile wireless multihopping is rapid deployment and dy-

namic reconfiguration. When the wireline network is not available, as in battlefield communications and search and rescue operations, multihop wireless networks provide the only feasible means for ground communications and information accesses. Figure 1 shows the typical protocol layers of the wireless network. In the GloMo project, various alternatives were developed for each layer, leading to several different network architectures as shown at the bottom of Figure 1. The protocol choices at each layer have been mapped into Maisie simulation modules, yielding a modular simulator structure ideally suited for the comparison of multiple architectures. The architecture evaluated in this case study is the Cluster Token. Its key layers are described in the following subsections.

2.1 Clustering

In multihop, mobile wireless networks, the aggregation of nodes into clusters controlled by a cluster head provides a convenient framework for the development of important features such as code separation (among clusters), channel access, routing, and bandwidth allocation (Gerla et al. 1995). Using a distributed algorithm with a cluster, a node is elected to be the cluster head. All nodes within transmission range of the cluster head belong to this cluster. That is, all nodes in a cluster can communicate with a cluster head and (possibly) with each other. The complexity and overhead of clustering rests in the selection of the cluster head. The most important criterion is stability. Frequent cluster head changes adversely affect the performance of other protocols such as scheduling and allocation which rely on it.

In our clustering algorithm, only two conditions cause the cluster head to change. One is when two cluster heads come within range of each other. The other is when a node becomes disconnected from any other cluster. This is an improvement (in stability) over existing algorithms which select the cluster head every time the cluster membership changes. Figure 2 shows a clustering example.

2.2 MAC layer

Clustering provides an effective way to allocate wireless channels among different clusters. Across clusters, we enhance spatial reuse using different spreading codes (i.e. CDMA). Within a cluster, we use a clusterhead controlled token protocol (i.e. polling) to allocate the channel among competing nodes. The token approach allows us to give priority to clusterheads in order to maximize channel utilization and minimize delay. A cluster head should get more chances to

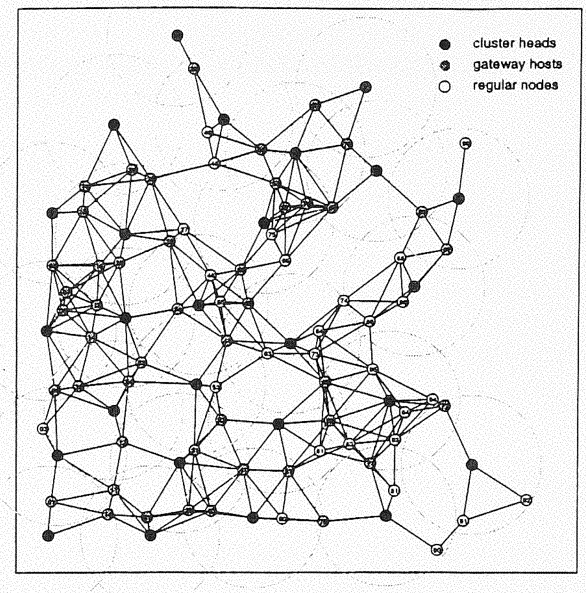

Figure 2: Clustering among 100 Hosts, Transmission Power=100

transmit because it is in charge of broadcasting within the cluster and of forwarding messages between mobile hosts which are not "connected".

The channel is assumed to be time slotted. In each time slot, only one node has the permission to transmit. In some cases the permission token may be lost. One such case occurs when the node with permission moves outside the cluster. Another case is when the host is a gateway (which belongs to more than one cluster). The gateway might be tuned to a different code (i.e. different cluster), thus missing the permission token which is then lost. To overcome these problems, the cluster head reissues the permission token after timeout. Figure 3 shows an example of slot scheduling (and gateway conflicts) in a 5 cluster network. Slot synchronization across clusters is assumed

2.3 Routing

Our model assumes shortest path routing. Routing tables are updated using a distance vector routing algorithm (Bertsekas et al. 1987). Each node maintains a routing table and broadcasts it to its neighbors whenever it gets the channel access permission. It updates its routing table when it receives the routing tables from its neighbors. In a time slot, the node broadcasts its routing table first and then transmit the data packet (if any).

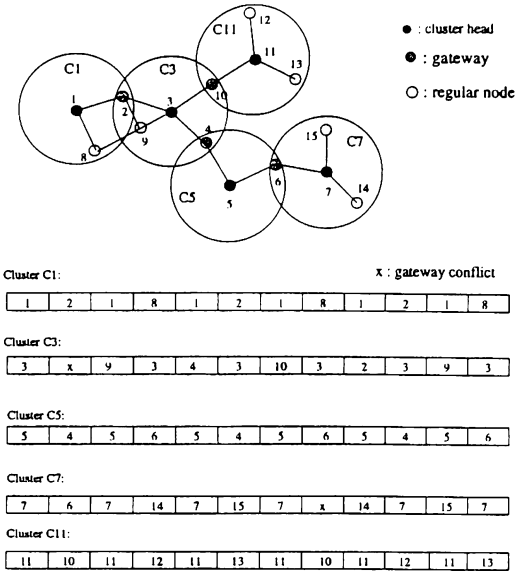

Figure 3: Example of Channel Scheduling

3 RADIO CHANNEL MODEL

An accurate radio channel simulation model is important not only for designing modulation and coding schemes that improve channel efficiency, but also for investigating the impact of channel fading on existing networking algorithms, such as clustering, routing and power adjustment. At present, most radio network protocol simulations are using the free space channel model which basically assumes that attenuation is only a function of transmitter-receiver distance. However, the radio channel characteristics are much too complex to be modeled by simple distance functions. Thus, the results are inaccurate.

To overcome this limitation, our simulator includes a rather sophisticated radio channel model which is an extension of the SIRCIM statistical impulse response model (Rappaport et al. 1990).

The radio channel is characterized by three propagation parameters: free space loss, multipath fading and shadowing. All these parameters are supplied by SIRCIM. SIRCIM provides impulse response characteristics which account for multipath fading. SIRCIM outputs include for example : the distribution of the number of multipath components in a particular multipath delay profile; the distribution of the number of multipath components etc.

SIRCIM provides impulse response at the *signal* level which is suitable for radio designs. For network performance evaluation purposes, we are more interested in received power at the *packet* level. Assuming that the channel is a Direct Sequence-Spread Spectrum channel, we can derive the mean signal power by performing convolution of the spread spectrum random chip sequence with the impulse response of the simulated channel.

Furthermore, in a mobile radio environment, we must model the fluctuation of received power caused by change of positions. To this end, the correlation between received power at different positions must be known. SIRCIM provides only *small-scale* spatial and temporal correlations. We have augmented SIRCIM to account for *large-scale* correlation as well.

In addition to multipath fading, the SIRCIM accounts for the shadowing effect caused by diffraction of radio waves around sharp edges. Shadowing, the slow fading, has been characterized in the literatures by roughly a log-normal distribution, with a standard deviation that depends on the roughness. A common assumption is that shadowing is independent from one location to another. Unfortunately, this assumption is not valid in a dynamic model with mobile users,in which location dependent correlation must be accounted for. In our simulation, we include the correlation model for shadow attenuation developed in (Gudmundson 1991).

The SIRCIM channel module is invoked (with a function call) every time a packet is transmitted in the network. Repeated computation of attenuation is required because the nodes are mobile and therefore change their relative position continuously.

4 SIMULATION ENVIRONMENT

A general purpose parallel environment for wireless network simulation has been designed at UCLA. The simulator is being built on an existing process-oriented, parallel simulation language called Maisie (Bagrodia and Liao 1994). A Maisie program is a collection of entities, each of which 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 communication (send and receive) primitives. Thus transmission delays in a physical

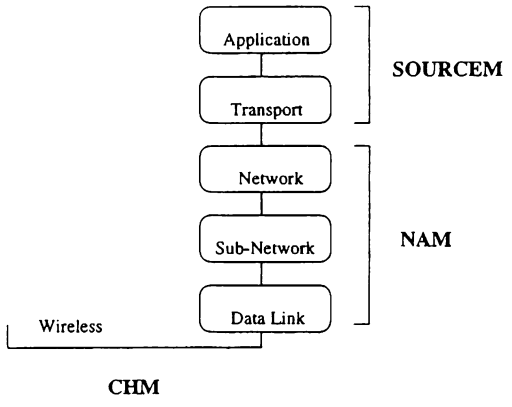

Figure 4: Networking Layers and Their Models

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 Maisie program can be interfaced with a number of simulation algorithms: 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 et al. 1993), or a space-time implementation of the optimistic synchronization algorithm (Bagrodia, Chandy, and Liao 1991).

4.1 Model

The wireless simulator is being used to evaluate the stability and performance of network algorithms as a function of the radio characteristics, network control, and mobility patterns. A detailed description of the simulator and its use in evaluating mobile networks is described in (Bagrodia et al. 1995). This section recapitulates the simulator and describes the modifications that were necessary for efficient parallel execution. The simulator modules that have been used to construct the model described in this paper include: Application Traffic Models (SOURCEM), Network Algorithm Models (NAM), and Channel Models (CHM). Figure 4 depicts the primary layers in the simulator and describes the interface between them.

The SOURCEM components are used to generate the traffic load for the network. The NAM components are broken down into network layer protocol models such as IP, and wireless subnetwork control such as clustering, and mobility control such as power control, logical link control, and media access control. The clustering, cluster-head election, routing and channel access schemes described in the previous section were simulated in the NAM modules. The CHM components are used to provide a set of free space and fading channel models to simulate the wireless medium. The channel models include the effect of multi-path fading, shadowing, and interference for both indoor and outdoor transmissions. The model described in this paper was developed to study the relative stability of the clustering algorithm, and average number of hops and packet delay between a given source and destination pair under both free space and fading channel assumptions.

4.2 Reducing Simulation Execution Time

Simulations of complex systems such as the wireless network described in this paper can quickly become computationally intractable as the number of wireless nodes and the detail of the channel and other component models is increased. Some techniques that have been used to reduce the execution time of the model include: model aggregation, cloning components, hierarchical modeling and parallel model execution

Aggregation (Sanadidi 1993) or coarse grain modeling is an effective method if a component model's accuracy is fairly independent of the amount of detail used in modeling the component itself. Cloning (Sanadidi 1993) is useful in cases where there are many simulated components which tend to have the same behavior throughout the simulation. Hierarchical modeling (Bagrodia et al. 95) combines coarse and fine grain models to reduce simulation time in an iterative fashion. Each of the preceding methods reduces the execution time for the model by reducing the level of detail and hence potentially increasing the inaccuracy in the model. In contrast, parallel execution, if effective, can be used to reduce the execution time without reducing its accuracy. The next section describes the primary modifications that were made to the model to support its parallel execution.

4.3 From Sequential To Parallel Simulation

Parallel execution of the model basically requires that the model be subdivided into N partitions, such that $N \geq P$, where P is the number of processors available for execution of the model. In this paper we assume $N = P$. Each partition executes on a unique processor. The execution across partitions is synchronized through a parallel simulation algorithm. As the synchronization is handled transparently by the Maisie run-time system, in theory it should be easy to modify a sequential model for parallel execution. However, as

described in a companion paper in this proceedings, there are a number of pitfalls that must be avoided in order to get an efficient parallel implementation. In the rest of this section, we first describe the modifications that were made to the initial model (Bagrodia et al. 1995) to yield a **correct** parallel implementation. Subsequently, we describe the modifications that were made to improve its parallel performance.

The first step in the parallelization effort was to eliminate global variables as the distributed memory architecture of SP/2 does not provide coherency support. This appears to be trivial, but can be complex if the model assumes some form of global view of the system for a distributed application and allows global variables to be modified and read at will. The best solution to preserve good performance in the parallel execution, is to redefine the global data structures in a distributed manner. For instance, there may be a global data structure which stores information regarding the VC connection status among all wireless nodes. Such a data structure should be distributed such that a wireless node is only aware of the VC connection status of itself and its immediate neighbors.

The next step is to eliminate unnecessary synchronizations in the model. In the sequential model, an entity was used to advance the time for all other entities in the system by broadcasting a message at the end of each slot interval. Since the time advance (equal to slot duration) is fixed, each entity can advance its local clock asynchronously eliminating the need for a centralized entity. Note that if the slot duration was determined based on some global property of the system, the decentralization would be considerably harder to implement.

For parallel execution, the simulation must be free of zero delay cycles in the communication topology. A model is said to contain a zero delay cycle if it is possible for an entity to receive a causally-ordered message with the same timestamp as another message sent by the entity (Misra 1986). Parallel execution of such models can either lead to deadlocks or instability. A sufficient condition to avoid zero delay cycles is to ensure that the receive time of a message is always greater than its send time; alternatively, it is possible to avoid such cycles by ensuring that an entity does not transmit a message with the same timestamp as an incoming message.

Upon the completion of these changes, the simulation was executed in parallel using the conservative run-time version of Maisie which is based on Chandy-Misra's null message parallel simulation algorithm (Misra 1986). A null message is a synchronization message which advances the clocks of neighbor entities. The initial result yielded slow-downs as opposed to speed-ups! Measurements showed that this was due to an extremely high "null message" ratio (defined as the number of null messages sent for every real message). This poor lookahead was certainly unexpected because the system utilizes fixed slots to synchronize within each cluster. Fixed slot length should itself serve as a good indication of lookahead. The poor lookahead in the initial model can be explained by the following two reasons.

First, the previous intuition of a good lookahead was based on the assumption that messages are only exchanged at the time that a slot starts or ends, but never in the middle. In reality, a number of messages such as routing updates, token arrivals, and data packets may arrive any time within the slot. The order in which these messages are processed sometimes will impact network performance; therefore, lookahead is not as simple as specifying a fixed slot time. Second, the original lookahead was poor because a number of the output messages are conditional, meaning that the entity has not enough information locally to deduce when it will send a message.

The following changes were made to the model to improve its lookahead and the null message ratio. (a) We exploited the fact that given the current state, although an entity does not know which message it will send next, it does know the time at which the message will be sent (if any). (b) Messages, whose sending and receiving time can be set to the start or the end of a slot without affecting the correctness of the simulation, are modified to take advantage of this behavior. By doing this, we were able to derive better lookahead upon receipt of a message. These changes in the model yielded a null message ratio in the range of 4.5 to 5.5 down from 25 in the first attempt. It is usually desirable to have a null message ratio of less than 1, but for the radio broadcast channel , this is the best that we are able to achieve given the fact that wireless network utilization is optimized when each node has on average 6 neighbors (Kleinrock et al. 1978).

Another issue of importance that may affect parallel performance is partitioning. In selecting partitions, it would be desirable to exploit fully both locality, and load balancing. Locality is achieved if neighboring nodes are placed on the same processor, thus reducing the communication overhead compared to neighboring nodes being placed on different processors. Load balancing (i.e. uniform distribution of entities among processors) is important as well because we want to fully utilize all the available computational resources. In our application, there are several strategies: random balanced, local balanced, and clustered balanced. Random balancing assigns enti-

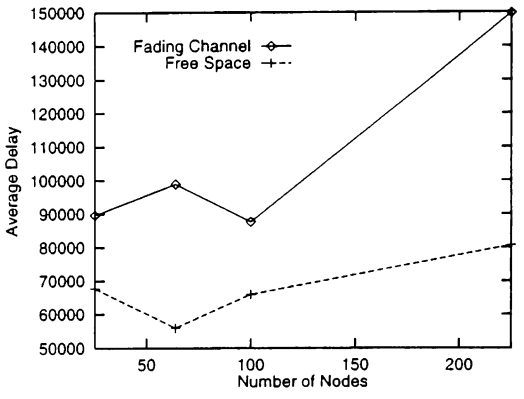

Figure 5: Network Performance

ties randomly to processors with the constraint that each partition has an approximately equal number of nodes. Local balancing tries to assign neighboring entities to the same processor while assigning an approximately equal number of entities to each processor. Clustered balancing first clusters the nodes into super nodes and then assigns the super nodes among different processors. The experiments reported in this paper use the first two partitioning techniques.

5 RESULTS

This section presents the results of parallel execution using Maisie conservative run-time simulator. We are studying the convergence of the distributed clustering algorithm, and throughput, average number of hops and end to end packet delay under both free space and fading channel assumptions. The experiments reported in this paper assume light load in the network with minimal mobility and were executed in exclusive mode on an IBM SP/2 multicomputer. Each node of this architecture is a RS/6000 processor with 128MB of RAM and a peak rating of 225 Mflops. The interconnection bandwidth is 40 Mbps and the hardware latency in the switch is 500 ns. Including software overheads, measurements show a minimum message latency of 30 μs. In order to ensure that the simulation has reached steady state, the baseline simulation time was established to be 250000 for all sets of runs.

5.1 Experiment Configurations

We consider four network configurations with 25, 64, 100, and 225 nodes randomly deployed in a square area (as in Figure 2) using a uniform probability distribution. The experiments serve two purposes, namely, to evaluate network performance and at the same time assess the efficiency of the parallel simulation implementation. Regarding network performance, Figure 2 illustrates a typical solution of the of the distributed clustering algorithm in the 100 nodes example. This solution is reached very rapidly after a few message exchanges among neighbors. Figure 5 shows the average end to end delay for both free and fading channel, for various network sizes. As expected, fading channel delay is higher due to packet loss and retransmission caused by fading.

To evaluate parallel simulation efficiency, the network examples are executed both sequentially, and in parallel on 1, 2, 4, 8, 12, and 16 processors respectively. We carry out three experiments. The first experiment uses random balanced partitioning while the second uses the more intelligent local balanced partitioning. Both experiments use a free space channel model. The third experiment uses the same partitioning strategy as the second, but employs a fading channel model which substantially increases the computational load.

5.2 Speed Ups

For free space channel model experiments, Figure 6, relative speed ups of parallel execution versus sequential execution. The speedup measure is derived by dividing the sequential execution time by the parallel execution time. In both experiments, the speed up generally improves with the number of nodes since a large number of nodes yields better load balancing across processors. The only glitch is for the random balanced partitioning experiment with two processors. This particular random balanced partitioning strategy uses a counter, which is incremented as an entity is created. In this strategy, entities are placed alternatively on one or the other processor in the order in which they were created. Since entities are sequentially deployed in the square grid as they are created, it follows that two successive entities are likely to be neighbors, and yet will fall in two different partitions. This is an example of conflict between locality and load balancing, which leads to high interprocess message cost and degraded performance. Next, we consider local balanced partitioning.

Table 1 compares the locality of messages in the two partitioning schemes used. Locality is defined as the number of messages (packets) that are sent within a processor, expressed as a percentage of the total number of messages. As expected, the local balanced partitioning provides significant improvement in locality. As a result, better speedups are obtained, as shown in Figure 7. Despite the improvement, however, the speed ups are not very high because of a high null message overhead (as explained at the end

Number of nodes	25	64	100	225
Random	11.9	14.4	11.8	19.3
Local	21.3	36.4	40.7	53.4

Table 1: Locality(%) of messages for the two partitioning schemes used, as the number of nodes are changed. The nodes are partitioned and mapped onto 16 processors.

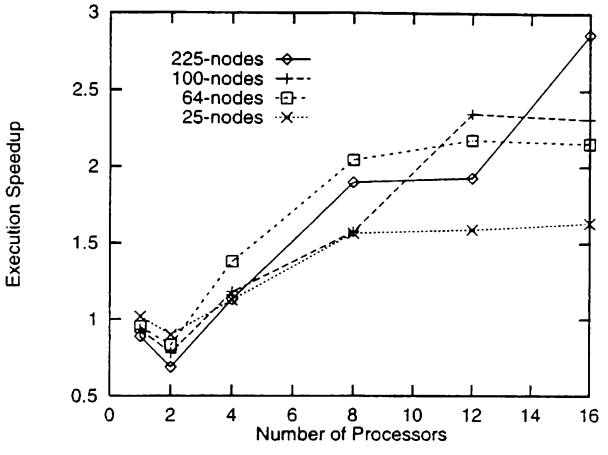

Figure 6: Free Space Channel with Random Balanced Partitioning

of this section).

For the fading channel model experiment, we only used local balanced partitioning since it performed better in the previous set of experiments. In Figure 8, the speed-up again increases with the number of processors and wireless nodes. In some cases, single processor parallel execution outperforms the sequential execution which uses a global event list. As discussed in (Jha et al. 1993), the primary reason for this is the lower context switching overhead. The higher speed up yielded by the fading channel model with respect to the free space model (10.4 vs 3.4) is due to the higher computation versus internode communication overhead. It is expected that future, more advanced radio network models will exhibit an even greater computation versus communication load ratio, due to sophisticated adaptive antennae and spread spectrum encoding schemes. In this experiment, we noticed that the 64 nodes case generally have higher speed up than the 100 nodes case except for the case with 12 processors. This is because for the 12 processors case, the 64 nodes can not be divided evenly among the processors.

Despite improving the lookahead, we found that NMR, the number of null messages sent for every real

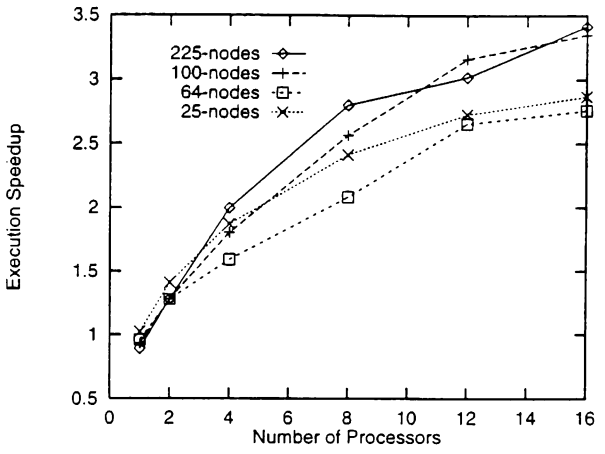

Figure 7: Free Space Channel with Local Balanced Partitioning

message, was relatively high - around 5.5-6.0. Since the computation associated with each real message (packet) is small in case of the experiments without the fading channel model, a high NMR means that the time spent in processing null messages is more than that spent in processing the real messages, resulting in relatively modest speedups. When the fading channel model is introduced, the computation associated with each real message increases substantially. Thus, even though the NMR is still high (around 6.0), the percentage of time spent in processing null messages is reduced, resulting in much better speedups (as good as 10 on 16 processors). Overall, the speed up is very encouraging considering the fact that we are able to reduce execution time from 3+ hours (sequential execution) down to 20 minutes using 16 processors for a network with 225 nodes.

6 CONCLUSION

A parallel simulator has been developed to study a number of protocols at different layers related in the design of mobile wireless networks using Maisie, a general purpose parallel simulation language. The message-passing paradigm used by Maisie provided a natural and simple way to model such networks. To study scalability issues, Maisie offers a parallel simulation approach which reduces execution time while preserving model accuracy. Results for static topology have shown that speed-up tend to increase as the number of nodes in the simulation increases. For the 225 nodes configuration with fading channel model, a speed-up factor of 10 was achieved. This dramatically reduces the turn around time of a simulation run.

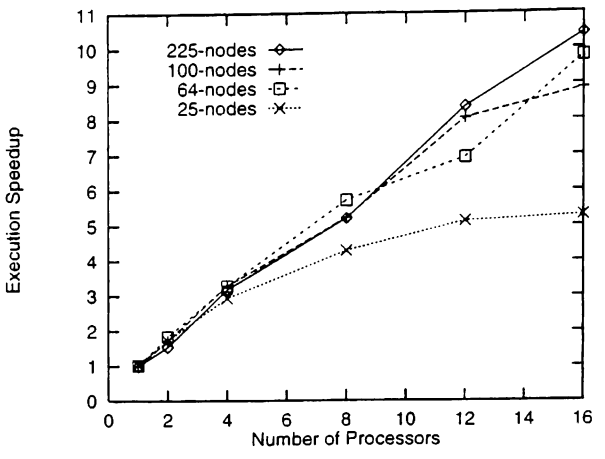

Figure 8: Fading Channel with Local Balanced Partitioning

In porting the simulation, we learned the following lessons: (a) A model designer should not assume the availability of a global snapshot of the system state. This is not an unreasonable constraint since no global snapshot exists in distributed applications. It is just an artifact to simplify the model. (b) Conditional messages are often unavoidable due to the behavior of the protocol that one tries to study. Thus, programmers must take great care to minimize the null messages that may result. One solution is to break up a single receive statement which may receive many different types of conditional messages into a sequence of receives if the arrival order of the various conditional messages is known. (c) Partitioning strategy used to decompose the network was found to affect the performance of the parallel simulator. (d) The general advice to programmers is that the simulator should be written from the beginning as a parallel program. The sequential execution should be considered as a special case. To our knowledge, this is the first parallel simulation environment that supports rapid prototyping of mobile wireless protocols. Work is in progress to parallelize the mobility model, to compare tradeoffs between conservative and optimistic run-time, and to study the behavior of the speed-up factor with respect to network load, degree of mobility and partitioning strategies.

ACKNOWLEDGEMENTS

This research was supported by the U.S. Dept. of Justice/FBI, ARPA/CSTO under contract J-FBI-93-112, by ARPA/CSTO under Contract DABT-63-94-C-0080 "Transparent Virtual Mobile Environment", and by Intel under project "QoS Wireless Networks".

REFERENCES

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. M. Gerla, L. Kleinrock, J. Short, and J. T. Tsai. 1995. A Hierarchical simulation environment for Mobile Wireless Networks. *In proceedings of the 1995 Winter Simulation Conference*, December.

Bagrodia, R. and W-T. Liao. 1994. Maisie: A Language for the Design of Efficient Discrete-Event Simulations. *IEEE Transactions on Software Engineering*, 20(4):225-238.

Bertsekas,D. and R. Gallager. 1994. *Data Networks*, Prentice-Hall.

Gerla, M. and J. T. Tsai. 1995. Multicluster, mobile, multimedia radio network. *Wireless Networks*.

Gudmundson, M. 1991. Correlation Model For Shadow Fading In Mobil Radio Systems. *Electron. Lett.*, 2145-2146.

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

Kleinrock, L. and J. Silvester. 1978. Optimum transmission radii for packet radio networks of why six is a magic number. *Nat. Telecom. Conf.*

Misra, J. 1986. Distributed Discrete-Event Simulation. *Computing Surveys*, 18(1), March.

Rappaport, T. S. and S. Y. Seidel. 1990. SIRCIM: Simulation of Indoor Radio Channel Impulse Response Models, *VTIP, Inc.*.

Sanadidi, M. 1993. A Comprehensive Performance Model of a Massively Parallel Processor. *Proceedings of AT&T Database Day*, October.

AUTHOR BIOGRAPHIES

RAJIVE L. BAGRODIA is an Associate Professor in Computer Science at UCLA.

CHING-CHUAN CHIANG is currently a Ph.D candidate in Computer Science at UCLA.

MARIO GERLA is currently a Professor in Computer Science at UCLA.

VIKAS JHA is a research engineer in Computer Science at UCLA.

WINSTON W. LIU is currently a Ph.D. candidate in Computer Science at UCLA.

HSIAO-KUANG WU is currently a Ph.D. candidate in Computer Science at UCLA.