Efficient Simulation of Wireless Networks using Lazy MAC State Update *

Young J. Lee and George F. Riley

School of Electrical & Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
Email: {young, riley}@ece.gatech.edu
Tel: (404) 894-9041; Fax: (404) 894-9959

Abstract

Scalable and efficient network simulation methods are the method of choice for evaluating and verifying wireless network protocols on a moderate to large scale. This need becomes obvious when simulating very large–scale wireless networks such as emerging ad hoc sensor networks in which the number of nodes can be the order of thousands or more, and with very high node density. Unfortunately, simulation of such large–scale wireless networks often requires excessively large amounts of computing resources and can be slow to complete. One approach to achieving higher performance in a large–scale network simulation is the use of parallel *or* distributed *simulation techniques. However, the efficient distributed simulation of wireless ad hoc networks is still a daunting task. Therefore, we turn our attention to more traditional sequential simulation methods, and seek to reduce the overhead incurred in the Medium Access Control (MAC) state update propagation between wireless nodes. We introduce a novel method called* LAMP *(LAzy MAC state uPdate), that substantially reduces this overhead, with no loss of accuracy. Using our wireless network simulation tool, we compare the efficiency of the* LAMP *approach to the more traditional approach, and show a performance improvement of up to a factor of eight, with no loss of accuracy.*

1 Introduction

The importance of effective evaluation and verification of wireless network protocols has been growing with the advancements and proliferation of wireless communications and computer technologies. A number of high–quality network simulation environments exist for analysis of the performance of wireless ad hoc networks, including *ns2* [1, 2], *GloMoSim* and its commercial counterpart *QualNet* [3], *OPNet* [4], and the Georgia Tech Network Simulator (*GTNetS*) [5]. All of these tools have detailed models of the IEEE 802.11 [15] wireless MAC protocol, as well as models for physical layer path loss. However, these tools also all suffer from degraded performance when simulating the wireless protocols, when compared to a similar sized *wired* network simulation. There are several contributing factors to the reduced performance for wireless simulation tools, including the complexity of accurate path loss and fading calculations in the physical layer, and the excessive number of simulation events needed to coordinate the MAC state updates between wireless nodes.

When the scalability property of a network protocol is especially of interest, scalable and efficient network simulation methods are required. This need becomes evident when one wishes simulation of very large–scale wireless networks such as emerging ad hoc sensor networks in which the number of nodes can be the order of thousands or more, and the node density can be very high. A traditional network simulation tool such as *ns2* is usually a poor choice in such an environment, due to excessive execution time and memory requirements.

One approach to address the performance issue is the use of *parallel* or *distributed* network simulation techniques, such as those used by *GloMoSim*/*QualNet* [3], *SSFNet* [7], *SWAN* [8], *pdns* [9], and *GTNetS* [5]. For the most part, these parallel simulation tools use a conservative synchronization approach, and rely on *lookahead* [10] to achieve reasonable performance. Usually, the lookahead value in parallel network simulations is obtained from the propagation delay of a signal going through a communication medium, as well as the packet transmission time on the link. However, in wireless networks, this propagation delay is usually very small (order of micro-seconds). Further, the MAC state information must be propagated to peers when the *first bit* of a packet is received by a receiver. Hence,

*This work is supported in part by NSF under contract numbers ANI-9977544, ANI-0136969, ANI-0240477, ECS-0225417, and DARPA under contract number N66002-00-1-8934.

Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS'05)
1087-4097/05 $20.00 © 2005 **IEEE**

the performance improvement of wireless network simulations through parallel simulation techniques has not been significant, and sometimes it is even worse than a *sequential* simulation [6].

In this paper, we introduce a different approach to improving the performance of wireless network simulations. Our approach, called *LAMP*, leads to substantial improvements in overall execution time and reduction in the size of the pending event list.

Our technique is motivated from the observation that informing all *potential* receivers of a given transmission is not necessary. In general, the MAC state of a node in a wireless network is only important *if the node wishes to transmit a packet*. More traditional approaches schedule a packet reception event at all potential receivers of a packet, including *undesignated* [1] receivers. The *designated* receivers of course must be able to sense and consequently receive the packet. However, undesignated receivers do not have to be aware of the transmission unless they are interested in accessing the communication medium in the near future. If one of the undesignated receiver nodes wishes to access the channel later, then it has only to update its MAC state related with the channel access according to the previous transmissions, i.e., according to the *history* of the medium access by other transmitter nodes.

The remainder of the paper is organized as follows. Section 2 gives an overview of the *LAMP* technique, with details given in Section 3. In Section 4, the performance evaluation results of *LAMP* are presented. We review some related work in Section 5, and then finalize the paper in Section 6 with conclusions and future work.

2 Overview of Lazy MAC State Update

As previously mentioned, with the *LAMP* method of delayed MAC state updates, the MAC state of each node is not updated needlessly. It is brought up-to-date only when a node wishes to join communications as a transmitter or a designated receiver. For example, let us suppose that there are three nodes, A, B, and C in a simple wireless network, and they share an IEEE 802.11 [15] wireless medium as shown in Fig. 1. Let us assume that each node is within the transmission range of the others, and node A wishes to send a unicast packet to node B with a preceding Request-To-Send (RTS) – Clear-To-Send (CTS) exchange. With a more traditional approach, every transmission from A and B will incur packet reception events and the relevant MAC state updates at C since C is within the transmission range of A and B. With *LAMP* however, no packet reception event will

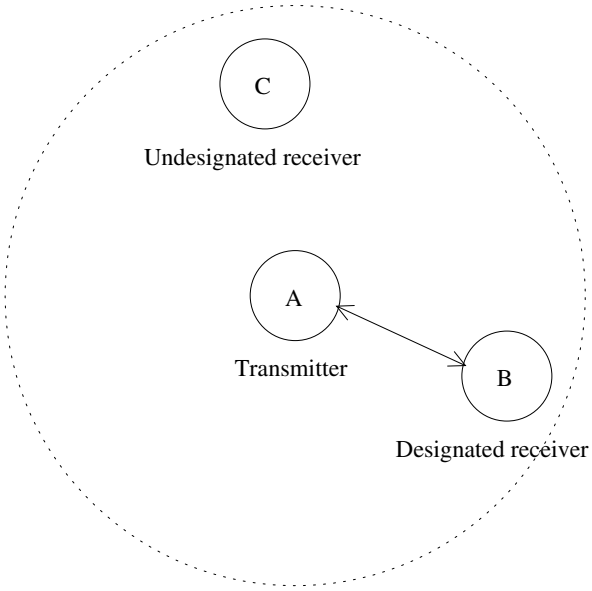

Figure 1. An example of a simple wireless network.

be scheduled at C and no MAC state update occur at C, because it is undesignated. If C wants to communicate with A while the communication between A and B is going on, C must become aware that it cannot access the medium at that time, and does so by updating its MAC state according to the medium access history. Thus, it behaves as if it heard the RTS-CTS exchange and performed the corresponding physical and virtual carrier sensing.

With *LAMP*, a *wireless link event list* is introduced and implemented as a medium access history buffer. This list contains full information necessary to bring the MAC state of undesignated receivers up-to-date. Its detailed structure will be discussed in the following sub-sections.

The wireless link event list is maintained per wireless link object that represents a wireless communication channel. [2] In addition, another list known as a *communicating entity list* is used. This list is used to guarantee that a node can safely change its MAC state without consulting the wireless link event list if it receives a packet. In a wireless network, a situation can occur where a node is both a communicating entity and an undesignated receiver, since a node can be involved in multiple communications at the same time. In such a case, the packet reception event from a communication in which the node is undesignated needs to be scheduled because the node is a communicating entity (as a transmitter or a designated receiver) for another com-

[1] For unicast communications, there is only one receiver, and it is referred to as a designated receiver. The others that are not designated but hear the transmitted signal are called undesignated receivers.

[2] The wireless link is a virtual channel object for scheduling of packet receipt events occurring through the channel. Note that it is not related with the channel state of a node.

munication and consequently needs to keep its MAC state updated.

The communicating entity list contains the nodes that are currently involved in communications. Therefore, any node that wishes to communicate through a medium will be added to this list, as well as all designated receivers. The added entities will be removed from the list as soon as the communication involving them completes.

When nodes want to communicate through a medium, they first must update their MAC states according to the wireless link event list. Then the nodes are added to the communicating entity list. When a node determines that it is allowed to access the medium, it transmits the packet, and records information about the transmission in the wireless link event list. The packet reception event due to this transmission is scheduled at the designated receivers normally, and is also scheduled for all nodes on the communicating entity list, excepting the transmitter itself. The fact that a node is on the communicating entity list means that the node must keep its MAC state updated. This is why the packet reception event should be scheduled at the nodes on the communicating entity list as well as the designated receivers. After the transmission completes, the transmitter and the corresponding designated receivers are removed from the communicating entity list.

In unicast communications, a data packet transmission completes when the sender finishes transmitting it and receives an ACK for the data packet from the view point of the sender. From the perspective of the receiver, the communication ends when the receiver of the data packet finishes transmitting an ACK to the sender. In the broadcast case, the communication ends as soon as the sender finishes transmitting the data packet and the receivers receive it. At the starting point of each communication, the sender and the designated receivers are entered into the communicating entity list, and they are removed from the list at the termination of the communication.

The overall effect of the *LAMP* method is a significant reduction in overall execution time due to the reduced number of MAC state update events, at the expense of the maintenance of the wireless link event list and the communicating entity list. We show later that benefits of the reduced event count are significant as compared to the relatively small overhead of the list maintenance.

3 The Detailed Structure and Algorithm

3.1 The List Structures

One of the key structures of *LAMP* is the wireless link event list. There is one such list for every wireless link object that represents a wireless medium, and it is accessible from each node in the network that shares the medium. Each item in the list consists of the following elements:

- Time that the transmission started
- Time that the transmission ended
- Location of the transmitter node
- Information of the transmitted frame

The first element is the timestamp for the start of a packet transmission event. The second element is used to compute the time of a packet reception event at each receiver. The propagation delay is added to this time to obtain the exact point of time for the packet reception. The third element is used to compute the distance between the transmitter and the receiver so that this distance can be used as one of the inputs to the propagation model, and the signal strength at the receiver can be calculated. The last element specifies the characteristics of the transmitted packet such as frame type and the type-specific information. For example, it will specify if the frame is RTS, CTS, Data, or Acknowledgment (ACK) as well as the corresponding information in the case of the IEEE 802.11 [15] as the MAC protocol.

The wireless link event list is implemented using a double-ended queue, with new items added at the end. Since the generation of the wireless link events are created strictly in timestamp order, this list is naturally sorted by ascending timestamps. When entries are added, old entries are removed when they become so old as to be no longer meaningful. Each node maintains a logical index pointer to the list that indicates the most recent item upon which its MAC state was updated.

The communicating entity list is simply a list of nodes (actually of wireless link interfaces) that are currently involved in communications as a transmitter and/or a designated receiver. Hence scheduling of a packet reception event must be done at each entity on this list if it is within transmission power range of a given packet transmission. In practice, it can be implemented in such a way that it holds a list of pointers to the corresponding objects. This list is also maintained per wireless link object, and each node that shares the medium can access this list.

3.2 Procedures for Lazy MAC State Update

There are a number of housekeeping details needed to properly maintain MAC state using *LAMP*. These can generally be classified into three main categories:

- Updating the MAC state (UPD_MAC)
- Joining the communicating entity list (JOIN_CE)
- Leaving from the communicating entity list (LEAVE_CE)

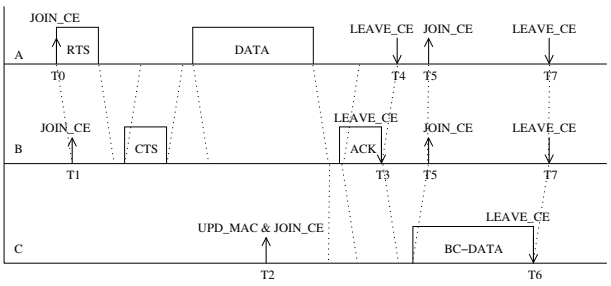

Figure 2. The timing diagram of lazy MAC state update.

The UPD_MAC procedure operates on the wireless link event list, and updates the MAC state of a node according to the packet transmission history. The other two procedures work with the communicating entity list, and they add or remove a node to or from the list.

Fig. 2 shows the timing diagram of the *LAMP* algorithm for the previous example of the simple wireless network described in Fig. 1. At time $T0$, A initiates a unicast communication by transmitting an RTS to B, and then the JOIN_CE procedure is called at A. B detects the first bit arrival of the RTS at time $T1$, joins the communicating entity list, and generates CTS for A. These RTS and CTS frames, however, are not scheduled for reception at C even if it is within the transmission range of A and B because it is not designated, and consequently no carrier sensing function is performed by the MAC layer of C. Let us suppose that there is a broadcast data request for C from the upper layer at time $T2$. C now updates its MAC state according to the wireless link event list that holds the past transmission records and behaves as if it heard the RTS-CTS exchange between A and B. After the update, C also joins the communicating entity list. After the unicast communication completes, and the corresponding LEAVE_CE procedure is performed at time $T3$ for B and at time $T4$ for A respectively, C is allowed to access the medium and transmits the requested broadcast data. This initiates another communication, and thus B and C perform the JOIN_CE procedure at time $T5$ when they hear the transmission, and the LEAVE_CE is requested at the end of the communication, i.e., at time $T6$ when A finishes transmitting the data packet and at time $T7$ when A and B successfully receive it. The detailed operation of each procedure is described as follows.

3.2.1 UPD_MAC Procedure

This procedure is called for just before a node joins the communicating entity list to bring the node's MAC state up-to-date. There are two right points in the method where this must be done. One is from a sender's perspective when a node has received a data send request from the upper layer, and the other is from a receiver's view point when a node has detected the first bit arrival from the sender.

In the UPD_MAC procedure, a node scans the wireless link event list and processes each item in the list. First, it examines if there are any items that need to be processed. If all the items are *old* (meaning they have already been processed), the procedure just returns without performing MAC state update. Otherwise, the corresponding items are processed according to the ascending order of time that each event occurred.

Generally, each update is done according to the following steps:

1. Compute the locations and the distance between communicating entities.
2. Calculate the signal strength after applying the propagation model.
3. Determine if the node can hear the transmission based on the signal strength and the transmission range.
4. Update the MAC state and perform carrier sensing functions as needed.

The MAC state variables involved in step 4 can be, for example, idle time, physical carrier sensing state, and network allocation vector (NAV) specified in IEEE 802.11. In step 4, scheduling of a packet reception event at its own node can happen if the first bit arrival time of the packet is past, and the last bit arrival is in the future. In that case, the node generates a corresponding packet according to the frame information in the wireless link event list, and schedules the reception at its own interface. Also, this procedure deals with overlapping of multiple signals if the *radio capture* capability of the MAC layer is enabled.

3.2.2 JOIN_CE Procedure

In the JOIN_CE procedure, a node is added to the communicating entity list after the corresponding update of its MAC state is done by the UPD_MAC procedure. For unicast communications, the designated receiver is added to the list as soon as the receiver detects the first bit arrival of a packet. For broadcasts, any node that can hear the transmission is added to the list. The transmitting node is also added to the list regardless of the communication type. This procedure first checks if there is already the same entity existing on it before adding a new entity.

Once a node is added to the communicating entity list, every packet that can be heard by the node is scheduled for reception at the node from that time on. The fact a node is on the list means that the MAC state of the node keeps being updated. This is to enable a communicating node to

change its MAC state safely without preceding MAC state update when it has received a packet.

3.2.3 LEAVE_CE Procedure

The LEAVE_CE procedure deals with removal of a node from the communicating entity list. This procedure can be requested at multiple places once a node is added to the list by the JOIN_CE procedure.

Typically, this procedure is called at the end of each successful packet transmission. For the 802.11 protocol, this occurs from a sender's perspective when it has received an ACK frame for the packet, and from a receiver's perspective when the receiver has finished transmitting the ACK. For broadcast communications, it is the moment when the sender completes transmitting a packet and when the receivers successfully get the last bit of the packet.

The LEAVE_CE procedure must be handled carefully when a communication ends with an error. A communication error can occur due to collisions in a wireless network and result in unsuccessful transmission or erroneous packet reception. For that case, each node involved in the communication is withdrawn from the communicating entity list by the LEAVE_CE procedure.

4 Performance Evaluation

In this section, we present the performance evaluation results obtained from several different scenarios. The *LAMP* technique is compared to the traditional simulation method through extensive simulation experiments. The simulation environment is described in detail, and then the simulation results are presented and discussed.

4.1 The Simulation Environment

All of our simulation experiments were done using GTNetS [5]. GTNetS is a scalable simulation tool designed specifically to support large–scale simulations. The design of the simulator closely matches the design of real network protocol stacks and hardware.

The *LAMP* technique can apply to any type of MAC protocols, but we used the IEEE 802.11 [15] protocol for all experiments with two-ray ground reflection [16] as the propagation model.

We did two sets of experiments to measure the performance of the *LAMP* technique. First, we tested it for two extreme cases. One was for unicast only networks, and the other for broadcast only networks. As discussed, the key to *LAMP* is that we schedule packet reception events only at communicating entities so that unnecessary computation regarding MAC state maintenance can be saved. Therefore, the performance gain is expected to be great when there are a small number of communicating entities per transmission in the network. This is the case for the unicast only network. On the other hand, the performance may not be as good as in the former case if there are lots of entities involved in a transmission, which is the case for the broadcast only network since all the nodes within a transmission range are designated receivers in a broadcast. This is the reason that we want to test the new method for the two extreme cases. For this first set of experiments, we put all the nodes in such a way that every node can sense every other's transmission, i.e., every communication can happen in a single hop in order to remove effects of routing protocols. In this set of experiments, we executed simulations varying the number of traffic flows from 5 to 20, and the number of nodes in the simulated network was 100.

For the second set of experiments, we constructed more realistic networks with two mobility characteristics. One is with no mobility, and the other with the random waypoint mobility model [2]. For the latter case, the pause time was 200 seconds, and the node speed was uniformly distributed between 0 and 10 meters/second. In this experiment, the primary variables were node density and traffic volume. We varied the node density from 10 to 100, and the number of traffic flows from 5 to 20. We also performed experiments varying the network topology and traffic patterns. The ratio of the radio transmission range to the network radius was properly set to force multi-hop routing for this set of experiments. The number of nodes in the simulated network was 500.

For both experiments, all traffic was created with a constant-bit-rate (CBR) data source with a rate of 4 packets/second, with a fixed packet size of 512 bytes. We executed 10 simulation runs per scenario and averaged them to represent each data point in the simulation results. Each simulation executed for 500 simulated seconds.

A circular network topology was used for the simulated networks. In this context, the node density is defined as the number of nodes per unit area covered by a transmitter. Thus it can be computed by $D = \frac{N}{\pi R^2} \cdot \pi r^2$, where D is the node density, N the network size, R the network radius, and r the transmission range. For each experiment, the node density was pre-determined, and then the computed network radius was used to generate the network topology with radio transmission range of 250 meters.

We used as performance metrics the following:

- The memory usage
- The number of events processed
- The execution time

For the execution time, we convert it to *speedup* rather than directly specify the value. The speedup metric cap-

tures the performance improvement as compared to traditional methods.

4.2 Two Extreme Cases

Fig. 3 shows the simulation results from the unicast only experiment, and Fig. 4 the broadcast only experiment. As can be seen in Fig 3(a) and Fig. 4(a), the memory usage of *LAMP* is nearly identical to that of the traditional method. On the average, *LAMP* consumed only 57 KB more than the traditional method, which is less than 1 % with respect to the total memory footprint. This is because the only memory overhead incurred by *LAMP* is the use of the wireless link event list and the communicating entity list whose sizes are relatively small. This phenomenon can be observed throughout all of our experiments, and can be explained in the same way.

As expected, in Fig. 3(b), we can see that *LAMP* significantly reduces the number of events processed during the simulation. The event count was reduced by a factor of 15 for 5 traffic flows, a factor of 6 for 20 traffic flows, and a factor of 7 on the average. It can be also observed, for both methods, that the number of events increases with the number of traffic flows as expected. However, the total event count for *LAMP* shows a relatively gentle slope as compared to the traditional method.

As can be seen in Fig. 4(b), both simulation methods produced an identical number of events for the broadcast only experiment. This is as expected, since every node within a transmission range is a designated receiver in broadcast communications. With the *LAMP* method, therefore, a packet reception event was scheduled exactly in the same way as in the traditional method.

For the traditional method, it can be observed that the processed events in the unicast only experiment were much more than those in the broadcast only experiment. This is due to the increased number of packets for unicast communications using additional frames such as RTS, CTS, and ACK.

Fig. 3(c) shows speedup achieved by *LAMP* with respect to the traditional approach. The simulation with *LAMP* was up to about 5 times faster for 5 traffic flows, and about 2 times faster for 20 traffic flows. On the average, *LAMP* demonstrated speedup of about 3 in these experiments.

In Fig. 4(c), on the other hand, we can see that speedup by *LAMP* is less than one, which means that the simulation with *LAMP* was actually slightly slower than the one with the traditional method. Even though the difference is small (about 10 % slower on the average), one may wonder what made this difference while the both methods processed exactly the same number of events during the simulations. This question can be answered as follows. With *LAMP*, even though there is no action taken by the MAC_UPD procedure as every node is designated, there is still some computational overhead incurred by the JOIN_CE and LEAVE_CE procedures. These procedures are requested at the beginning and at the end of each communication respectively. For the unicast only experiment, this overhead was outweighed by the computation savings obtained from the lazy MAC state update. This was, however, not the case for the broadcast only experiment where there was no such computation benefit.

4.3 The 500-Node Experiments

Figure 5 and Figure 6 show the simulation results from the 500-node experiments with no mobility. The first was obtained using various node densities with a fixed number of traffic flows while the second varied the number of traffic flows with fixed node density.

Fig. 5(b) shows that *LAMP* substantially reduces the number of events processed during the simulation. It is worth noting that the number of events for *LAMP* remains almost constant with node density while the event count for the traditional method increases. For the traditional method, this phenomenon seems reasonable since a packet reception event needs to be scheduled at all nodes within a transmission range even for unicast communications. For *LAMP* however, the number of nodes that need scheduling of a packet reception event for unicast communications is no longer a function of node density. This implies that *LAMP* is scalable with respect to the node density of a network, and is responsible for the nearly flat graph with varying node densities.

In Fig. 6(b) we can see, for both methods, that the events increase with the number of traffic flows. But the slope of the line graph for the two method is quite different. The event count for the traditional method quickly increases with traffic flows, which is not the case for *LAMP*.

As can be seen in Fig. 5(c), a speedup of about up to 8 was achieved by *LAMP*. On the average, the simulation with *LAMP* was about 6.5 times faster than the traditional method. Speedup increases with node density because time for processing events with the traditional method increases with node density, while time for processing events with *LAMP* is largely unaffected by node density.

Fig. 6(c) shows speedup obtained from the 500-node experiment varying the number of traffic flows. We can see that speedup decreases as the number of traffic flows increases. The increased number of traffic flows translates to the increased number of nodes that need packet reception scheduling per unicast transmission. This is the main reason for the decreasing speedup with traffic flows. Another reason for this result is explained by the fact that the increased traffic volume incurs more collisions, packet losses, and occasional perceived link failures, resulting in more routing

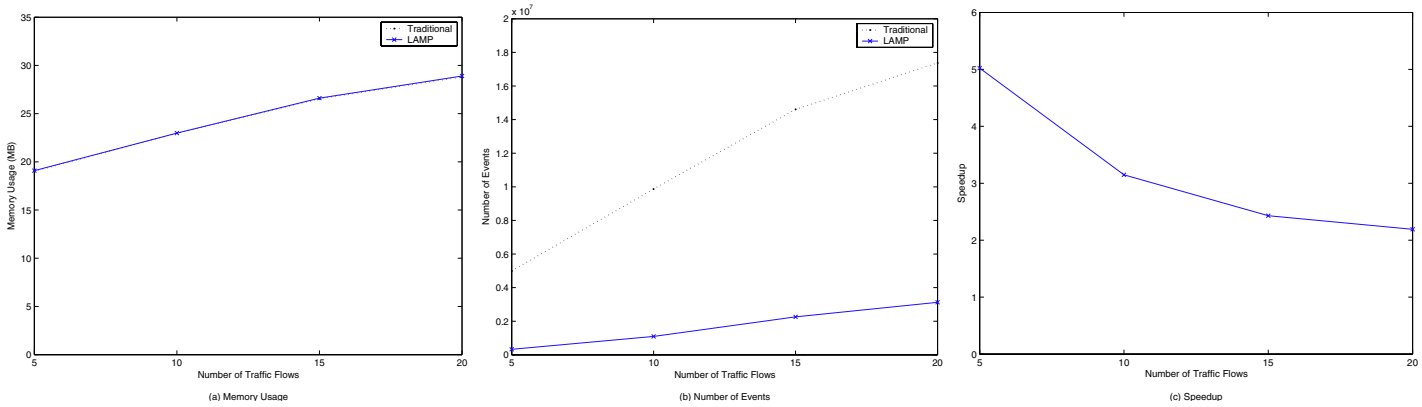

Figure 3. *Unicast only (100 nodes with varying traffic flows).*

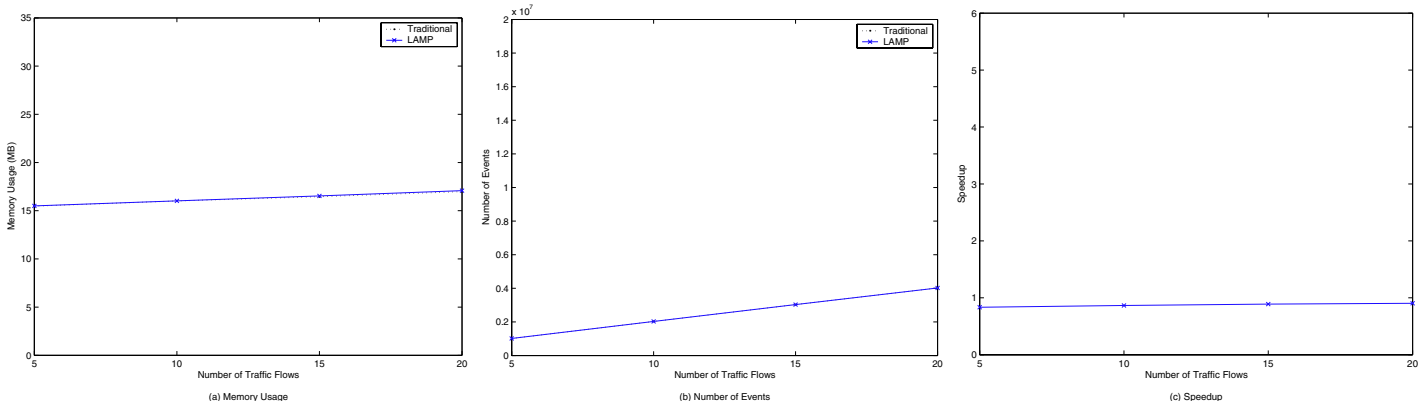

Figure 4. *Broadcast only (100 nodes with varying traffic flows).*

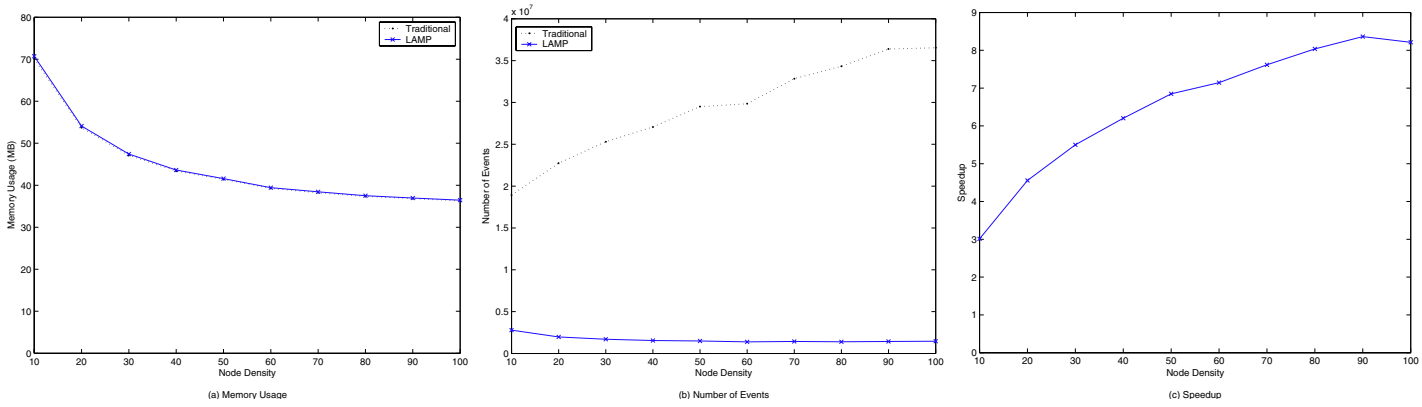

Figure 5. *500 nodes with no mobility (varying node densities with 10 traffic flows).*

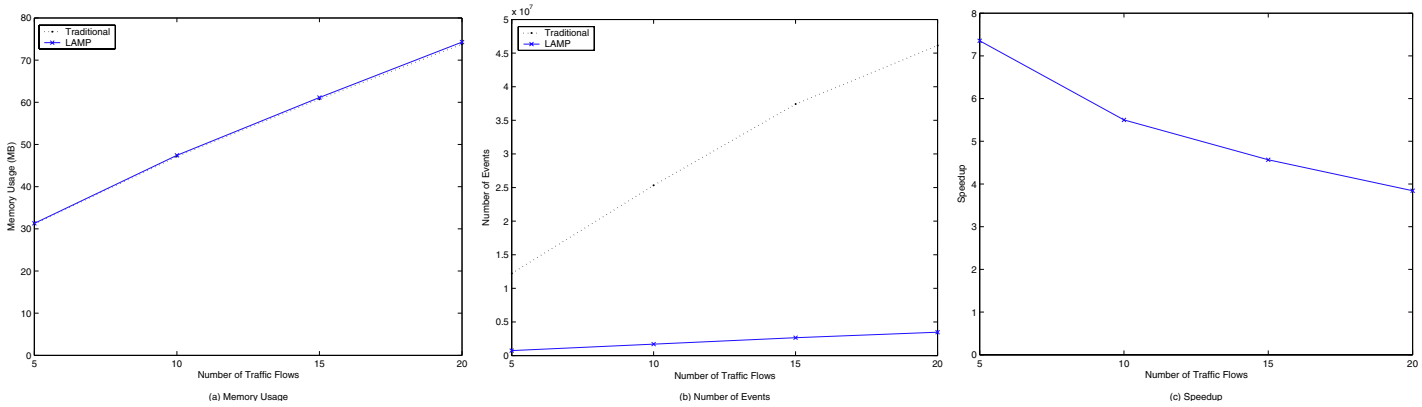

Figure 6. *500 nodes with no mobility (varying traffic flows with node density of 30).*

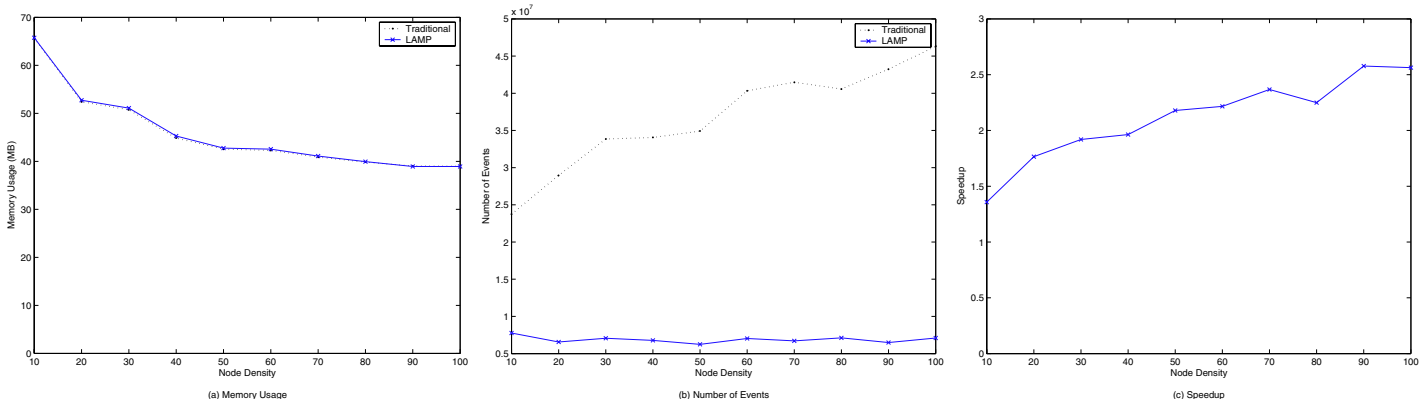

Figure 7. *500 nodes with mobility (varying node densities with 10 traffic flows).*

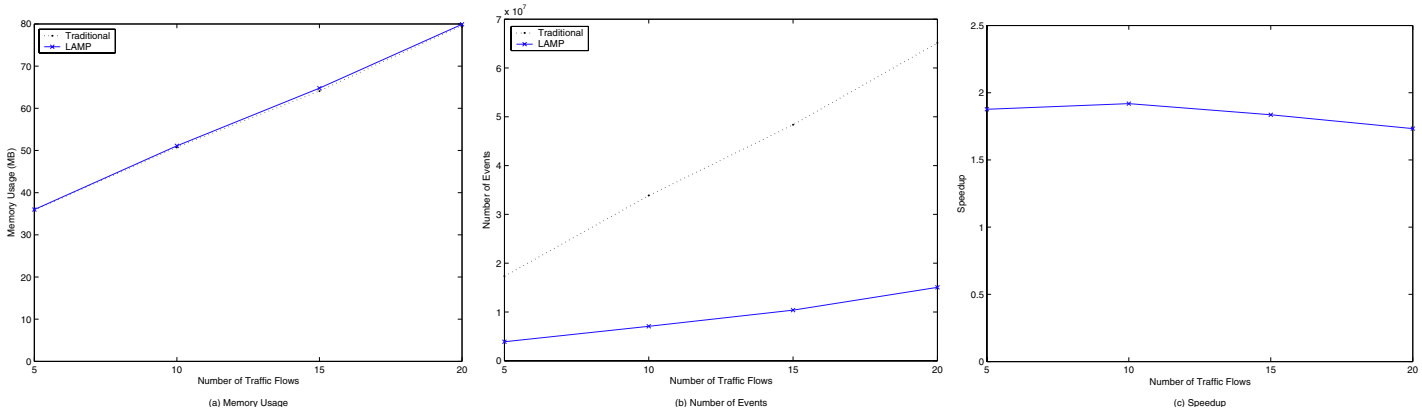

Figure 8. *500 nodes with mobility (varying traffic flows with node density of 30).*

message traffic. As discussed in the two extreme cases, the increased number of broadcast packets can adversely affect the performance of *LAMP* relative to the traditional method. For this experiment, speedup of up to 7 (for 5 traffic flows) was achieved. On the average, speedup of about 5 was achieved by *LAMP*.

Fig. 7 and Fig. 8 show the simulation results from the 500-node experiments with mobility. The first was obtained using various node densities with a fixed number of traffic flows while the second varied traffic flows with fixed node density. In general, as can be seen in these figures, the overall tendency is very similar to that for the previous experiment with no mobility.

The graphs in Fig. 7(b) look similar to those in the preceding case with no mobility. The number of events processed in the simulation with the traditional method grows with node density while the number for *LAMP* remains relatively stable. This again confirms that *LAMP* is scalable with respect to the network node density. For both methods, the absolute number of events increased as compared to the prior experiment with no mobility. This results from the increased number of routing related messages due to mobility induced link failures and subsequent additional routing messages.

The same tendency can be observed in Fig. 8(b). For both methods, the number of events processed during the simulation increases with the number of traffic flows, as discussed previously.

Fig. 7(c) shows the speedup values achieved by the *LAMP* method with mobility. Overall, speedup decreased as compared to the one with no mobility. The simulation with *LAMP* was nearly 2.6 times faster than the one with the traditional method, and about 2.1 times faster on the average. This reduced speedup as compared to the no mobility case is a result of the increased portion of broadcast packets due to route discovery.

Fig. 8(c) shows speedup obtained from the 500-node experiment with mobility varying the number of traffic flows. The maximum speedup was about 1.9, and the average speedup of 1.8 was achieved from this experiment. The reason for decreasing speedup with traffic flows was discussed above.

Even though it was omitted due to lack of space, the simulation with the *LAMP* method produced the identical simulation results as compared to the simulation with the traditional method, which means that *LAMP* can be used for simulation without loss of accuracy.

5 Related Work

There have been several other efforts to improve the performance of sequential wireless simulations and to enhance the existing simulation tools.

In [11], a simplified MAC model is developed and used. In this work, the claim is that a detailed model is both unwanted and unnecessary for protocol design purposes. Thus, this work is limited to the use of simulations for the purpose of higher layer protocol verification.

In [6, 12], a *network gridding* technique is proposed, where the physical network is divided into several partitions. With this approach, a radio signal is not allowed to propagate over the grids in which nodes are out of range from the transmitter. This technique is implemented at the layer that deals with channel propagation.

In [13], a *staged* approach is proposed, where a grid-based method is used to compute neighbors, and several optimizations are suggested to eliminate computational redundancies. Redundant computations are avoided by function caching. This approach differs from ours and the work in [6] in that our work focuses on optimizations in a single protocol stack layer, whereas theirs works across layers.

In [14], the *lazy event scheduling and corrective retrospection* technique is proposed, where a non-receivable signal is not scheduled for reception. Thus, it is determined based on the strength of a signal whether or not to schedule a packet receipt event. In ours, on the other hand, it is determined based on the state of a node, i.e., whether a node is a communicating entity that is involved in active communications.

All these works are complementary to our work. In addition, our work is unique in that it uses a *pull* technology for the MAC layer, in which necessary information is retrieved and computed on a demand basis. This is in contrast to existing wireless simulations where a *push* based approach has been adopted.

6 Conclusions and Future Work

We presented a novel simulation technique for mobile wireless networks, which is efficient in terms of the execution time and the event list size, and can be achieved with a sequential simulation. Our technique is motivated from the observation that scheduling of a packet reception event at undesignated receivers is not always necessary, since undesignated receivers do not have to be aware of the transmission unless they are interested in accessing the communication medium. The necessary information can be determined by the lazy MAC state update algorithm as needed. The new method is applicable to any kind of MAC protocols, but we implemented it for the IEEE 802.11 [15] MAC sub-layer for our simulation experiments.

Extensive simulation experiments were carried out to assess the performance of the proposed simulation technique, and the simulation results show that the new technique tremendously reduces the simulation events processed during the simulation and consequently shortens the execution

time significantly.

Even though this work improves the efficiency and scalability of wireless network simulations, other issues such as a large amount of computing resources due to very large–scale network simulations still need to be addressed. Toward this end, we are planning to explore parallel and distributed simulation techniques as well.

References

[1] The network simulator – ns-2, available at `http://www.isi.edu/nsnam/ns/`.

[2] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *ACM Int'l Conference on Mobile Computing and Networking (MobiCom'98),* October 1998.

[3] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "GloMoSim: A scalable network simulation environment," *Technical Report, UCLA Computer Science Department - 990027,* 1999.

[4] `http://www.opnet.com/`.

[5] G. Riley, "The Georgia Tech Network Simulator," *ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools'03),* August 2003.

[6] M. Takai, R. Bagrodia, K. Tang, and M. Gerla, "Efficient wireless network simulations with detailed propagation models," *ACM/Kluwer Wireless Networks, Vol. 7, No. 3, pp. 297-305,* 2001.

[7] J. Cowie, D. Nicol, and A. Ogielski, "Modeling the global Internet," *Computing in Science & Engineering, Vol. 1, No. 1, pp. 42-50,* 1999.

[8] F. Perrone, D. Nicol, J. Liu, C. Elliot, and D. Pearson, "Simulation modeling of large-scale ad-hoc sensor networks," *Simulation Interoperability Workshop (SIW 2001),* 2001.

[9] G. Riley, R. Fujimoto, and M. Ammar, "A generic framework for parallelization of network simulations," *Int'l Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'99),* October 1999.

[10] R. Fujimoto, "Parallel discrete event simulation," *Communications of the ACM, Vol. 33, No. 10, pp. 30-53,* 1990.

[11] J. Liu, D. Nicol, L. Perrone, and M. Liljenstam, "Towards high performance modeling of the 802.11 wireless protocol," *Winter Simulation Conference (WSC 2001),* December, 2001.

[12] V. Naoumov and T. Gross, "Simulation of large ad hoc networks," *ACM Int'l Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'03),* September, 2003.

[13] K. Walsh and E. Sirer, "Staged simulation for improving the scale and performance of wireless network simulations," *Winter Simulation Conference (WSC 2003),* December 2003.

[14] Z. Ji, J. Zhou, M. Takai, and R. Bagrodia, "Scalable simulation of large-scale wireless networks with bounded inaccuracies," *ACM Int'l Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04),* October, 2004.

[15] IEEE Computer Society, "802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," June 1997.

[16] T. Rappaport, "Wireless communications: principles and practice," Prentice Hall, Upper Saddle River, 2nd edition, 2001.