ROUTING AND EFFICIENT EVALUATION TECHNIQUES FOR MULTI-HOP MOBILE WIRELESS NETWORKS

A Dissertation
Presented to
The Academic Faculty

By

Young-Jun Lee

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
December 2005

ROUTING AND EFFICIENT EVALUATION TECHNIQUES FOR MULTI-HOP MOBILE WIRELESS NETWORKS

Approved by:

Dr. George F. Riley, Advisor
Asst. Professor, School of ECE
Georgia Institute of Technology

Dr. Geofferey Li
Assoc. Professor, School of ECE
Georgia Institute of Technology

Dr. Henry L. Owen
Professor, School of ECE
Georgia Institute of Technology

Dr. Richard M. Fujimoto
Professor, College of Computing
Georgia Institute of Technology

Dr. Raghupathy Sivakumar
Asst. Professor, School of ECE
Georgia Institute of Technology

Date Approved: August 1, 2005

To my parents,

Sang-Keun and Sook-Hee Lee

ACKNOWLEDGMENTS

First of all, I would like to express my hearty gratitude to my advisor, Dr. George Riley, for the kind assistance and guidance he has showed me during the pursuit of my PhD degree. He has provided an excellent environment and valuable feedback for my PhD research. Without his help, I could not have completed this dissertation successfully.

I would also like to thank Dr. Henry Owen, Dr. Raghupathy Sivakumar, Dr. Geoffrey Li, and Dr. Richard Fujimoto for reviewing this dissertation and serving on my PhD dissertation committee.

I am grateful to my colleagues in the MANIACS Laboratory at Georgia Tech for their kind help and support.

Finally, I would like to express my sincere gratitude to my parents, Sang-Keun and Sook-Hee Lee, and my younger brother, Young-Jin, for their love, encouragement, and prayer. Without their love and support, I could not have begun my graduate study and finished my PhD degree successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this dissertation, routing protocols, load-balancing protocols, and efficient evaluation techniques for multi-hop mobile wireless networks are explored.

With the advancements made in wireless communication and computer technologies, a new type of mobile wireless network, known as a mobile ad hoc network (MANET), has drawn constant attention. In recent years, several routing protocols for MANETs have been proposed. However, there still remains the need for mechanisms for better scalability support with respect to network size, traffic volume, and mobility. To address this issue, a new method for multi-hop routing in MANETs called Dynamic NIx-Vector Routing (DNVR) is proposed. DNVR has several distinct features compared to other existing on-demand routing protocols, which lead to more stable routes and better scalability.

Currently, ad hoc routing protocols lack load-balancing capabilities. Therefore they often fail to provide good service quality, especially in the presence of a large volume of network traffic since the network load concentrates on some nodes, resulting in a highly congested environment. To address this issue, a novel load-balancing technique for ad hoc on-demand routing protocols is proposed. The new method is simple but very effective in achieving load balance and congestion alleviation. In addition, it operates in a completely distributed fashion.

To evaluate and verify wireless network protocols effectively, especially to test their scalability properties, scalable and efficient network simulation methods are required. Usually simulation of such large-scale wireless networks needs a long execution time and requires a large amount of computing resources such as powerful CPUs and memory. Traditionally, to cope with this problem, parallel network simulation techniques with parallel computing capabilities have been considered. This dissertation explores a different type of method, which is efficient and can be achieved with a sequential simulation, as well as a parallel and distributed technique for large-scale mobile wireless networks.

CHAPTER 1

INTRODUCTION

1.1 Background

A mobile ad hoc network (MANET) is a collection of mobile nodes that forms a network structure without the help of any existing infrastructure or central administration, such as static base stations in cellular networks. With the advancements made in wireless communication and computer technologies, the use and application of mobile ad hoc networks are increasing over time. For example, military operations or disaster relief efforts are usually performed without any pre-existing infrastructure. Also, commercial applications that need cooperative mobile data exchange can benefit from this mobile ad hoc networking technology [1]. Moreover, this type of network may provide an inexpensive alternative to cellular networks [2]. A MANET working group [3] was created by the Internet Engineering Task Force (IETF) to standardize these efforts in this field.

In recent years, several routing protocols for mobile ad hoc networks have been proposed. The primary goals of such routing protocols are to establish a correct route from a source node to a destination node in an efficient way, to maintain the discovered routes, and to react to the network topology changes effectively. Ad hoc network routing protocols can be classified into two major categories according to the way routes are acquired and maintained: table-driven (proactive) and on-demand (reactive) protocols.

In table-driven protocols, each node exchanges network topology information with other nodes by propagating update messages to have a consistent view of the network. Each node maintains routing tables to keep the routing information. The table-driven protocols provide accurate and fresh (up-to-date) routes since every node monitors and captures any change in the network topology, and each node propagates the corresponding information throughout the network. In general, immediate route acquisition is possible in these protocols, but they incur high routing overhead with a number of routing messages because

of topology changes and mobility. The protocols in this category include DSDV [4], WRP [5], ZHLS [6], STAR [7], OLSR [8], and TBRPF [9].

In on-demand protocols, routes are found and maintained on a need basis. The source node requests routes only when required, and the discovered routes are maintained only if desired. Each node maintains routing tables or caches to keep the routing information. This type of protocol is known to be more efficient than table-driven ones in general because it discovers and maintains routes only when there are explicit needs. The protocols in this category include DSR [10], AODV [11], TORA [12], and SSR [13]. Among these routing protocols, the most commonly studied protocols are DSR [10] and AODV [11].

Destination-Sequenced Distance-Vector Routing (DSDV) [4] bases its table-driven routing protocol on the distance vector algorithm. Each node periodically exchanges its routing information and forwards packets in hop-by-hop manner. This is an enhanced version of traditional distance vector routing protocols applied to mobile ad hoc networks. The routing overhead remains almost constant under a variety of mobility scenarios because this is a table-driven protocol. This protocol works only for the bi-directional links. In DSDV, however, the signaling overhead is very high due to the route advertisement messages that each node sends out periodically. This protocol also shows a poor packet delivery ratio under high mobility [50].

In Temporally-Ordered Routing Algorithm (TORA) [12], a distributed on-demand routing protocol based on the *link reversal* algorithm was proposed to reduce signaling overhead. This protocol tries to avoid communication overhead aiming at minimizing the signaling overhead when it reacts to the network topology changes. Thus, the route optimality is not the first concern in TORA. However, the link reversal algorithm produces short-lived loops in case of routing failure, and the protocol shows poor packet delivery performance and high routing overhead under the scenario with high mobility nodes and high volume of traffic sources [50].

In Dynamic Source Routing (DSR) [10], packets are delivered using *source routing*.

A DSR packet contains a complete ordered list of IP addresses for nodes through which it should pass. DSR is an on-demand protocol and therefore finds a path to a destination only when it wishes to send to the destination. The main advantage of this type of protocol is that the intermediate nodes on a path do not need to maintain up-to-date routing information, which removes the need for route advertisement and explicit neighbor-detection mechanisms. Further, it is straightforward to obtain loop-free routes with this approach. However, the size of a packet header containing route information grows with the length of a path, which results in high packet overhead and raises scalability concerns. Obviously, this could have a negative impact because of limited bandwidth in wireless networks. The authors in [14] attempted to reduce this packet overhead using *implicit source routing*. It successfully reduces the packet overhead due to source routing, but shows very similar performance to DSR in other metrics such as packet delivery and latency. Furthermore, DSR does not have a mechanism to invalidate the cached routing information in a timely fashion, which can lead to a routing misbehavior because of stale routes, resulting in poor performance in highly mobile networks.

Ad hoc On-demand Distance Vector (AODV) [11] adopts the *distance vector* routing algorithm. Each node exchanges routing information to attain the up-to-date view of the network only when involved in an active routing path, which makes it an on-demand routing protocol. AODV forwards packets in a hop-by-hop manner. It uses *HELLO* messages to maintain local connectivity at each node. One of the advantages of the protocol is that it scales well to large networks. However, even though the neighbor management associated with local connectivity is done only during an active routing phase, it does increase the signaling overhead of the entire network.

There exist other routing protocols that adopt different routing metrics instead of the minimum hop count [15, 13]. In Power Aware Routing (PAR) [15], a power-related metric was used. In Signal Stability Based Adaptive Routing (SSR) [13], link quality and location stability were used to find a route. However, these protocols suffer from the same problems

as in the routing protocols on which they base.

There also exist multicast routing protocols [16, 17, 18, 20, 21, 22] to support multicast routing in MANETs. There are other enhancements to multicast routing for MANETs [19, 23, 24, 25].

The Quality of Service (QoS) support in MANETs is another issue that has been constantly gaining considerable attention [26, 30]. Several routing protocols have been proposed to support QoS in MANETs [27, 28, 29, 31, 32, 33, 34, 35, 36].

Mobile wireless ad hoc networks usually consist of mobile nodes that are not reachable through a single hop. Therefore, the main focus of ad hoc routing protocols has been to support the wireless multi-hop routing capability. These wireless links usually have lower capacity than wired links. Hence, congestion can be a normal phenomenon rather than an exception in mobile ad hoc networks.

Currently, ad hoc routing protocols lack load-balancing capabilities. Thus, they often fail to provide good service quality, especially in the presence of a large volume of traffic as the network load concentrates on some nodes, resulting in a highly congested environment. This congested environment causes several undesirable effects such as long packet latency, poor packet delivery, and high routing overhead. It also causes excessive consumption of the network resources, such as bandwidth and power, that are usually scarce in these networks.

Recently, several studies have been performed in the ad hoc networking domain to balance the network load, to mitigate congestion, and to provide stable packet delivery.

In Dynamic Load-Aware Routing [56], the routing load of the intermediate nodes is used as the primary route-selection metric. A route request (RREQ) message keeps recording queue occupancy information of each node it visits, and the destination selects a path it considers the best based on the queue occupancy information recorded in the RREQs. This scheme, however, lacks path diversity since it is a single-path mechanism. Therefore, its load-balancing capability is limited.

In Alternate Path Routing [57] and Multi-Path Routing [58], path diversity is a main concern. To utilize path diversity, multiple paths are found per destination and are maintained at source nodes and used in turn for routing. The basic idea of these schemes is to distribute traffic among multiple paths. These protocols, however, need to maintain complex states to dynamically select a routing path among the multiple discovered paths. These multi-path protocols also incur additional routing overhead because they maintain more than one route per destination compared to single-path protocols [59]. Moreover, it is known that multi-path routing is effective when the alternate paths are disjointed, which is not easy to achieve in mobile ad hoc networks [60, 61, 62, 63].

All of these schemes can be classified as *end-to-end* approaches since source and/or destination nodes are responsible for selecting and maintaining single or multiple paths.

It is important to evaluate and verify wireless network protocols effectively. The ns-2 network simulator [87] is one of the most widely used simulation tools. Recently, wireless and mobility extensions to ns-2 have become available [50, 88], where detailed models for wireless propagation and medium access control (MAC) layer were added. This network simulation environment has been used broadly for wireless networking research.

When the scalability property of a network protocol is especially of interest, scalable and efficient network simulation methods are required. This need seems evident when simulating very large-scale wireless networks such as emerging ad hoc sensor networks. A normal network simulation software such as ns-2 [87] cannot be used for this purpose because simulation of such large-scale wireless networks usually requires very long execution time and a large amount of computing resources such as powerful CPUs and memory as well.

Traditionally, to cope with this problem, *parallel* network simulation techniques with

parallel computing capabilities have been considered [65, 67, 69]. There exist several parallel network simulators [89, 90, 91, 66, 92, 64, 48, 68, 86]. There are various time management techniques for parallel simulation, and basically they can be classified into a conservative synchronization method [70, 71, 72, 73, 74, 75] or an optimistic synchronization method [76, 77, 78, 79, 80]. There also exist hybrid approaches [81, 82, 83, 84, 85, 86].

Most of the parallel simulation tools mainly rely on *lookahead* [93], which is the ability to predict the earliest time of messages that can be generated in the future. Usually, the lookahead value in a parallel network simulation is obtained from the propagation delay of a signal going through a communication medium. However, in wireless networks, this propagation delay is very small (order of micro-seconds). Hence, the performance improvement of wireless network simulations through parallel simulation techniques has not been noticeable, and sometimes it is even worse than a *sequential* simulation [90].

There have been several research efforts to improve the speed performance of sequential wireless simulation and to enhance scalability of the existing simulation environments.

In [94, 95], 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.

The *network gridding* technique is proposed in [90, 99], 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 [100], 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.

The *lazy event scheduling and corrective retrospection* technique is proposed in [101]. In this work, 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.

1.2 Motivation and Solutions

1.2.1 Routing

There have been several routing protocols for MANETs proposed in recent years. All of these protocols prefer stored route information (at route caches or routing tables) in a route discovery phase. Even though this behavior can save some routing overhead, it often fails to sense the up-to-date network topology, which leads to the discovery of invalid paths, resulting in additional packet drops and more delays, especially in case of high mobility.

In addition, a node in a MANET tends to communicate with much more peers than those in a wired network due to the dynamic nature of the network. Therefore, frequent *address resolution* services are requested, which may degrade the performance of routing protocols in terms of packet latency and delivery [10, 41, 42] since some packets can be lost during the address query process.

Moreover, all the protocols show degraded performance in the face of a large network size, a large volume of traffic, or high network mobility [43, 44, 45]. Thus, there still remains the need for mechanisms for better scalability support with respect to network size, traffic volume, and mobility.

To address these issues, a new method for multi-hop routing in MANETs, which is called *Dynamic NIx-Vector Routing (DNVR)*, is proposed. DNVR basically inherits its routing functions from the wired NIx routing [37]. DNVR behaves in a reactive fashion to acquire loop-free routes and maintain them as do other reactive routing protocols.

However, DNVR has several distinct features compared to other existing on-demand routing protocols. First, the stored route information is validated before being used, and the up-to-date network topology is probed in an efficient way during a route discovery phase. Second, the routing states are managed in a timely fashion, and a conservative route discovery strategy is adopted by suppressing route requests. Thus, only a few routes are maintained per destination. Third, address resolution is obviated by using a NIx and MAC

addresses for routing purposes. Finally, a NV routing header is a compact form of source route, which significantly reduces the packet overhead due to source routing. By virtue of these features, DNVR provides more stable and scalable performance with respect to network size, traffic volume, and mobility.

1.2.2 Load Balancing

Due to the dynamic nature and the presence of wireless links, congestion is a normal phenomenon rather than an exception in a MANET. Currently existing ad hoc routing protocols, however, lack load-balancing capabilities, which leads to poor performance in the face of a large volume of network traffic. Under such an environment, network traffic can concentrate on some nodes, resulting in congestion that can cause several undesirable effects such as long packet latency, poor packet delivery, and high routing overhead as well as excessive consumption of the network resources, such as bandwidth and power, that are usually scarce in a MANET.

To address these issues, a novel load-balancing technique for ad hoc on-demand routing protocols is proposed. The new technique makes each node forward RREQ messages adaptively according to its load status. In this scheme, nodes that are considered as overloaded do not permit additional traffic flows to set up through them so that they can be excluded from the requested paths within a specific period. These nodes allow additional traffic flows as soon as they become no longer overloaded.

To control RREQ messages adaptively, the new technique utilizes interface queue occupancy and workload. In the new technique, a threshold value is maintained per node, which is a criterion for decision of how to react to a RREQ message. The threshold value of a node dynamically changes according to the load status of the node based on the queue occupancy and the workload change within a specific period.

The new load-balancing technique significantly reduces packet latency as well as routing overhead. In addition, it does not adversely affect but rather improves the network throughput. The new technique also successfully balances the network load among nodes.

Moreover, it operates in a fully distributed manner and utilizes only local information, and it can easily be incorporated with existing on-demand routing protocols to work on top of them.

1.2.3 Evaluation

Scalable and efficient network simulation methods are required to evaluate and verify wireless network protocols effectively, especially to test their scalability properties. 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 on the order of thousands or more, and the node density can be very high. Usually, simulation of such large-scale wireless networks needs a long execution time and requires a large amount of computing resources such as powerful CPUs and memory.

Using *parallel* or *distributed* network simulation techniques is one approach to address the performance issue. The parallel simulation of wireless networks, however, suffers from the small lookahead problem and consequently degraded speed performance. This problem results from the inherent small propagation delay of wireless networks because a lookahead value in a parallel wireless network simulation is mainly obtained from a propagation delay of a signal going through a wireless communication medium.

To address these issues, a novel approach to improving the performance of wireless network simulation is proposed. The new technique called *LAzy MAC state uPdate (LAMP)* is motivated from the observation that notifying all *potential* receivers of a given transmission is not always necessary. In general, the MAC state of a node in a wireless network is only important *if the node wishes to access the medium*. In the new technique, the scheduling of a packet receipt event and the corresponding MAC state update are deferred until there is an explicit need. Therefore, *LAMP* leads to substantial improvements in overall execution time and reduction in the size of the pending event list.

Even though *LAMP* improves the efficiency and scalability of wireless network simulation, other issues such as a large amount of computing resources due to very large-scale

network simulation still need to be addressed since *LAMP* is originally designed for sequential simulation of wireless networks.

To cope with this problem, the *LAMP* technique is extended to a parallel version. A parallel and distributed wireless simulation environment is constructed using GTNetS [48]. This simulation environment is based on the conservative synchronization method for time management. The enhanced version of GTNetS using parallel *LAMP* is also implemented for scalable simulation of wireless networks. The *LAMP* technique extended for parallel simulation also tremendously reduces the simulation events processed during the simulation and significantly shortens the execution time.

1.3 Dissertation Outline

The remainder of the dissertation is organized as follows. In Chapter 2, the DNVR protocol for MANETs is described in detail, and its performance is evaluated and compared through simulations. In Chapter 3, the novel load-balancing technique for MANETs is introduced, the detailed algorithms are explained, and its performance evaluation results are presented and discussed. Chapter 4 details the *LAMP* technique for efficient simulation of wireless networks, and its performance is assessed and compared through extensive simulations. Chapter 5 discusses the extension of *LAMP* for parallel simulation of wireless networks, and its performance evaluation results are presented. Finally, Chapter 6 concludes the dissertation.

CHAPTER 2

DYNAMIC NIX-VECTOR ROUTING FOR MOBILE AD HOC NETWORKS

2.1 Introduction

A mobile ad hoc network (MANET) is a collection of mobile nodes that forms a routing structure without the help of any existing infrastructure or central administration. In recent years, several routing protocols for mobile ad hoc networks have been proposed. The primary goals of such routing protocols are to establish a correct route from a source node to a destination node in an efficient way, to maintain the discovered routes, and to react to the network topology changes effectively.

In this chapter, a new routing protocol, which is called *Dynamic NIx-Vector Routing (DNVR)* for mobile ad hoc networks, is introduced. DNVR is a pure on-demand routing protocol. It acquires a loop-free route and maintains it on a demand basis as do other on-demand routing protocols [10, 11, 12].

The DNVR protocol, however, has several distinct features from other on-demand routing protocols as follows:

- Validation of the stored route information,
- Utilization of probes for efficient sensing of the network,
- Management of routing states in a timely fashion,
- Suppression of route requests for a conservative route discovery,
- Removal of address resolution,
- Use of a compact form of source routes.

DNVR effectively validates the stored route information before using it as well as efficiently detects the up-to-date network topology in a route discovery phase by utilizing a

unicast packet functioning as a probe. To accommodate a high degree of mobility, the routing states are invalidated in a timely manner. DNVR adopts a conservative route discovery strategy by suppressing route requests at destinations, and thus only a few routes are maintained per destination. In addition, DNVR uses MAC addresses instead of IP addresses to identify routing neighbors, which eliminates the need for additional mechanisms such as the *Address Resolution Protocol (ARP)* [38] to resolve a neighbor's IP address. [1] DNVR uses a compact form of a source route since it inherits the NIx-Vector routing [37] as a basic routing method, which reduces packet overhead significantly. By virtue of these features, DNVR provides more stable and scalable performance compared to existing routing methods, with respect to network size, traffic volume, and mobility.

2.2 Overview of Wired NIx-Vector Routing

The wired *NIx-Vector (NV)* routing protocol was introduced to provide an efficient routing method in the Internet [37]. The motivation for NV routing was provided by the observation that the specification of the IP address for each next hop in a routing path is an excess of information. The NV routing method enables the relevant routing information to be specified with a small number of bits per hop, and thus the same amount of routing information can be specified in a much smaller space. The NV routing makes source routing a feasible option in the Internet.

The NIx-Vector routing method starts with the concept of a *neighbor index (NIx)*. Each router has an ordered set of routing neighbors. For example, if a router has three routing neighbors, it has a set of {0, 1, 2}. The router selects the next hop from this set. The NV consists of the concatenation of the all *neighbor index (NIx)* values selected on the path from a source to a destination.

A NV is constructed during a creation phase while a packet goes from a source to a destination. During this phase, routers make normal routing decisions and record the

[1] The use of a MAC address for identifying a routing neighbor simply removes ARP, and it operates only from the perspective of per-hop behavior. Note that it does not violate the layered protocol approach.

neighbor index decisions in the packet header by simply concatenating the *NIx* value to the existing vector and incrementing the length by the appropriate number of bits. When the packet arrives at the destination, the NV is complete and is returned to the source. Once the NV is available at a source, it is included in all subsequent packets from the source to the destination, and the routers use it to efficiently forward the packets by extracting the appropriate number of bits from the vector and decrementing the length.

This method works well in wired networks, since each router needs a constant number of bits to record the NIx, and that number is expected to remain somewhat static over time. For routers that do expect to add next-hop neighbors over time, using some extra bits will allow a multiplicative increase in the number of routing neighbors that can be recorded. For example, if a router currently has 10 routing neighbors, then it needs 4 bits to record the NIx. But if that router is expected to have up to 64 neighbors in the future, it could simply use 6 bits to record and extract its *NIx* value.

Figure 1 shows a simple network as an example. Each router has varying numbers of neighbors, and each router numbers those neighbors sequentially, as shown. Suppose the path for a packet is 0–1–3–6–9–12. The resulting NV for this path is `0 10 10 011 1` (binary).

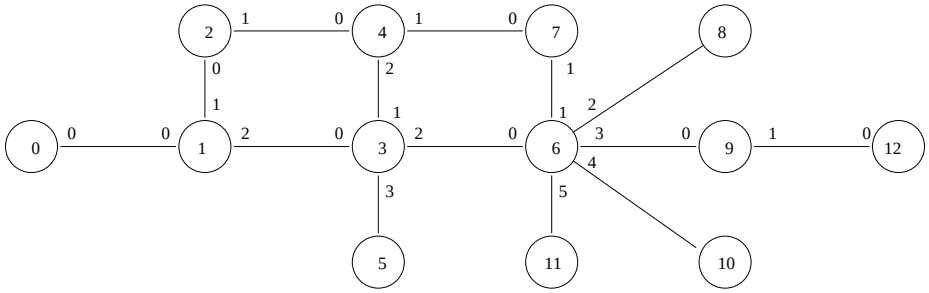


Figure 1. Simple routing path.

Once a NV from a source to a destination is created and becomes available to the source, it can be subsequently used for efficient $O(1)$ routing decisions. Each router extracts the

appropriate number of bits from the received NV, which specifies the correct NIx for the next hop. This NIx is used to index a table of the next hop IP address, interface number, and (optionally) layer 2 address. The packet is then forwarded to this neighbor with no further processing. The extracted NIx values at each router between 0 and 12 would be 0, 2, 2, 3, and 1, resulting in the correct path 0–1–3–6–9–12. To specify the same information using *Strict Source Routing*, the IP address of each next hop must be listed in the packet header. In this example, the packet header would contain the list of IP addresses 1–3–6–9–12, which is 20 bytes (or 160 bits) of information.

2.3 Packet Overhead Reduction

To achieve bandwidth efficiency, DNVR adopt the NV concept for routing packets in ad hoc networks. In DNVR, a neighbor is defined as a node that actively participates in a routing action. Thus, it is a potential next hop of a node. In addition, the number of routing neighbors is dynamic due to the mobility in an ad hoc network, which makes it difficult to record a NIx with a constant number of bits. To cope with this, A *dynamic NIx* is introduced whose length can be specified with an additional prepended field called *color*. The structure of this dynamic NIx is detailed in the later section. For the remainder of this discussion, *NIx* is used to term *dynamic NIx*.

The size of a NIx ranges from 4 to 18 bits, and consequently it provides a concise representation of a routing path resulting in low overhead. For example, if the hop count of a path is ten, the size of routing information in a packet header ranges from 5 to 23 bytes (or 40 to 180 bits) in DNVR while a DSR routing header needs 40 bytes (or 320 bits) to represent the same information (IPV4 case). Note that the NIx size is not affected by the addressing scheme of the different IP protocols (version 4 or 6). Moreover, each NIx is removed from the header once it is used, since the NIx that has been read and used at each node is no longer meaningful. In DSR, each IP address in a source routing header is not removed after it is used since it could be used later for returning routing messages. Hence,

the size of routing information in DNVR gets smaller as a packet is routed, and its average size is much less than that of DSR.

Figure 2 shows the result of a simple analysis on the packet overhead in terms of bandwidth consumption per hop by a packet header (including the IP header). For DSR, the overhead is $(160 + 32 \cdot L) \cdot r$, for DNVR, $\{160 + \frac{1}{L} \cdot \sum_{k=1}^{L} 10 \cdot (L - k)\} \cdot r$, and for an ideal case, $160 \cdot r$, where L is the hop count, and r is the transmission rate. Miscellaneous parts of a packet header were ignored. Note that the DNVR calculation represents the upper bound with the number of neighbors up to 127.

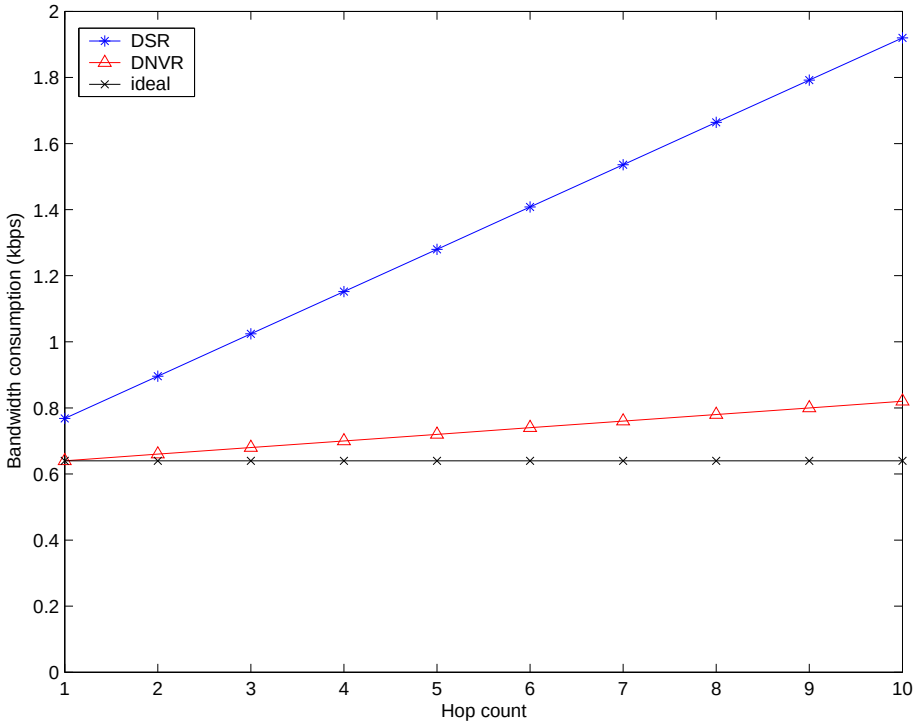


Figure 2. Packet overhead.

In this analysis, it was assumed that the transmission rate is four packets/sec and the IP header size is 20 bytes. As shown, the bandwidth consumption by a DSR packet header grows rapidly as the length of a path grows, which resulted from the nature of source routing. On the other hand, the bandwidth consumed by a DNVR packet header increases

slightly with the hop count running closely to that of the ideal case in which the IP header is the only overhead.

2.4 Removal of ARP

In every routing protocol for MANETs, address resolution services are requested once a next hop for a packet is determined at each intermediate node to resolve the IP address of the next hop to its MAC address. DNVR uses MAC addresses instead of IP addresses to identify each next-hop neighbor internally, which completely removes the need for address resolution mechanisms. Thus, a NIx extracted from a routing header can be directly mapped to a MAC address through a data structure called *neighbor table*. The structure of this neighbor table is described in the later section.

2.5 Data Structures of DNVR

The DNVR protocol mainly consists of two parts: NV creation and mobility management. In the NV creation phase, a path from a source to a destination is found, and a NV for the path is constructed. In the mobility management phase, DNVR detects routing failures and performs mobility management functions.

DNVR utilizes three types of data structures: NIx-Vector, neighbor table, and NV Forwarding Information Base (NV-FIB). Each structure is detailed as follows.

Figure 3 shows the NIx structure. A NIx has a color field and a neighbor index field. The color field specifies the number of bits for the neighbor index field. The neighbor index field represents the actual index of the neighbor table. The next hop can be determined from this index value at each forwarding node. The length of the color field is fixed (currently four) while the index field length is variable. The value to be stored in the color field N_b can be computed from (1).

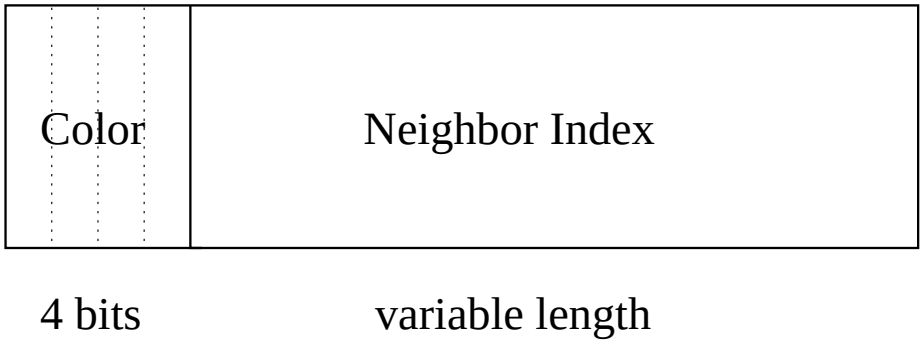


Figure 3. NIx structure.

$$N_b = \lfloor log_2 index \rfloor + 1 \tag{1}$$

where $index \in \{1, 2, 3, \dots\}$. The index starts from one due to the presence of a *hidden bit* [47] in the neighbor index field, where the bit 1 is assumed to always precede the number specified in the neighbor index field. Thus, the actual length of the index field is $N_b - 1$ bits since the color N_b counts the hidden bit, and the index value is obtained by prepending the hidden bit 1 to the number extracted from the index field. Given a four-bit color, the length of the index field can range from zero to 14 bits.

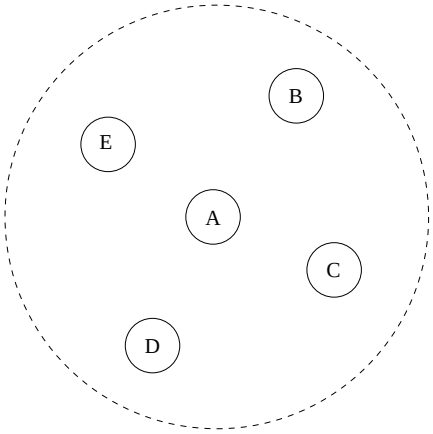
The four-bit width of the color field means that a node can accommodate up to 32K (2^{15}) neighbors. Even though this number seems to be large enough to handle every node as a routing neighbor in a realistic mobile ad hoc network, a slight increase in the color field will give more space. For example, if five bits for the color field rather than the four (only one bit increase) are used, a node will be able to accommodate up to about two billion (2^{31}) neighbors.

A NV is simply a sequence of NIxes with a NV length field. The NV length field represents the number of bits of the NV excluding the length field itself. The length of the NV decreases, while a packet with a NV is routed along a path since each NIx is removed from the NV as it is used. The NV length field also can be used to check whether or not the NV is valid. If the NV length is not positive, then the NV is invalid and cannot be used for routing. Thus, the packet with the invalid NV should be dropped immediately.

Each neighbor table entry consists of three fields: next hop, interface number, and lifetime. The entries are indexed by the *NIx* value and thus have $O(1)$ access time. In routing table based approaches, $O(logN)$ access time is typical.

The next hop field contains the MAC address of the next-hop neighbor's interface. This MAC address uniquely identifies a neighbor's interface and is used as a link-layer unicast address. The interface number specifies which interface to use for communicating with the relevant neighbor. In general, this interface is the one from which a node received routing protocol messages. The lifetime value specifies how long the associated entry will remain valid. An entry is deleted when the lifetime expires. Also, an entry can be invalidated even though the lifetime has not expired if the neighbor is no longer reachable.

Figure 4 shows an example of a simplified neighbor table. The interface number and the lifetime fields are omitted for simplicity in this example. Node *A* has four one-hop neighbors, *B*, *C*, *D*, and *E*. *A* assigns a unique index value to each neighbor. It can be observed that the index begins with one, and each index has a hidden bit 1 that is parenthesized.



Neighbor Table of Node A

Index	Neighbor
1 − (1)	B
2 − (1)0	C
3 − (1)1	D
4 − (1)00	E

Figure 4. An example of a neighbor table.

A NV-FIB is a table storing NVs and the relevant information. When there is a request for sending a data packet, DNVR searches its NV-FIB for a NV with the packet destination. A NV-FIB is indexed by a *path id*. A path id is a set of a source IP address, a destination

IP address, and a NV reply number. The NV reply number is generated only by a node that replies to NV request messages. Whenever a node replies with a NV reply, it increments its own NV reply number by at least one and puts the value in the returned message. In DNVR, a path can be uniquely identified by this path id.

Each NV-FIB entity has five elements: path id, NV, metric, state, and lifetime. The NV is a representation of the path identified by the path id. The metric field holds the corresponding information regarding the path such as hop count, load factor, etc. The state field indicates the validity of the NV. If it is invalid, the NV cannot be used to route data packets, but still can be used for route maintenance purposes.

The meaning of the lifetime depends on the state field. A NV is invalidated when (i) the lifetime expires with a valid state or (ii) the link to the next hop is determined to be broken. As soon as the NV switches to the invalid state, the lifetime field is refreshed with a new value. When this new lifetime expires, the NV is deleted permanently.

2.6 NIx-Vector Creation

The NV creation process relies on a flooding scheme using local broadcasts as do most ad hoc routing protocols [10, 11]. Even though recent studies [51, 52, 55] indicate inefficacy and unreliability of local broadcasts, and there have been several efforts to cope with this problem [53, 54], the main focus of this work does not lie on the topic.

A NV creation process is invoked when a node has packets to send and does not have a NV for the destination. The node then stores the packets in a buffer for pending packets and initiates a NV creation process by broadcasting a NV request (NVREQ) message. When a node receives a NVREQ and it is the specified destination, it returns a NV reply (NVREP) message to the source. Otherwise, the node propagates the NVREQ by broadcasting or unicasting if the request has not been processed by the node before. This action is described in detail later.

In this work, it is assumed that every link involved in a network is bidirectional. In

other words, it is assumed that the link layer protocol does not allow packet transmission over unidirectional links. The IEEE 802.11 [46] falls into this category. Thus, symmetric routing is a basic assumption where the reverse of a path from node *A* to node *B* is the path from *B* to *A*. The detailed operation is as follows.

A NV request (NVREQ) message carries a source IP address, a destination IP address, a NVREQ sequence number, routing metric information, a reverse NV, and the MAC address of a forwarding node's interface. When a node receives a NVREQ message, it first checks if it has already processed the request by examining a unique NVREQ identifier, a pair of <source IP address, NVREQ sequence number>. If the node has already processed the request, it drops the request and does not rebroadcast the packet. Otherwise, the node then processes the routing metric field of the NVREQ by incrementing the hop count by one.

The node then reads the MAC address of the forwarding (previous hop) node's interface from the NVREQ and adds it to the neighbor table. This action naturally returns a neighbor table index for the added item. This index is converted to a NIx as follows: the number of bits for the index, N_b, is computed by (1), and N_b is then prepended at the index as color. This NIx is concatenated to the *reverse NV* field of the NVREQ. The reverse NV is used to return a NVREP to the source later. The node then replaces the corresponding field of the NVREQ with the MAC address of its interface from which it received the request and rebroadcasts the request through the interface. This process is repeated until the NVREQ reaches the destination node.

When the NVREQ message finally reaches the destination, a NV reply (NVREP) message is constructed and returned to the originator of the request. DNVR adopts a conservative route discovery strategy. Whenever a destination node replies to a NVREQ, it records how many times it replied to the same NVREQ and checks if the value does not exceed a *reply threshold* value. If the node already replied the threshold value times, it simply drops the request. By doing so, only a few paths are maintained per destination in DNVR. For example, if a node has a reply threshold value of *one*, the node replies only to the NVREQ

that arrives first.

A NVREP message carries a source IP address, a destination IP address, a NV reply number, routing metric information, a NV, and the MAC address of a forwarding node's interface. Before constructing the NVREP, the destination node constructs a NIx from the NVREQ and completes the reverse NV by concatenating the NIx. The destination node then puts in the corresponding field of the NVREP the MAC address of its interface from which it received the request. The routing metric information is copied to the relevant field of the NVREP from the NVREQ, and the NV reply number maintained at the node is incremented and put in the corresponding field of the NVREP. Finally, the reverse NV in the NVREQ is copied to the NV routing header of the NVREP, which is used to return the NVREP to the requester. This routing action using NV is detailed in the later section.

The requested NV is constructed while the NVREP is routed along the reverse path from the destination to the source. Each node on the path reads the MAC address field of the NVREP and adds it to the neighbor table. The resulting NIx is concatenated to the NV field in the NVREP. The NV and the reverse NV (the NV in the routing header) are then copied and stored at the NV-FIB of each intermediate node. The NV represents the requested downstream path, while the reverse NV represents the upstream of the path. Each NV is stored with a relevant path id (Figure 5). This process is repeated until the NVREP arrives at the request originator. The requester extracts the NV from the NVREP and stores it with the corresponding information at the NV-FIB. The requester then can transmit the pending packets to the destination using that stored NV.

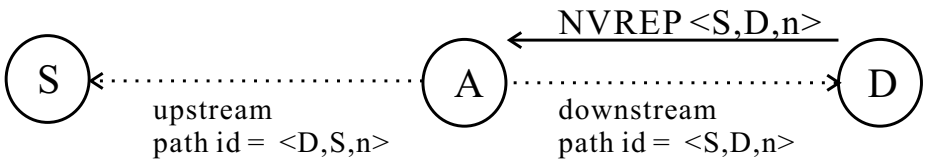


Figure 5. Forward and reverse NVs.

For loop detection, it is enough to uniquely identify a NVREP. A NVREP can be identified by its requester, the responder, and the NVREP number, i.e., a triple <source IP address, destination IP address, NVREP number>, which is a path id (Section 2.5). Let us suppose that node S requested a NV for node D, and thus D replies with a NVREP number n in Figure 6. In this example, the NVREP can be identified by $< S, D, n >$. Each intermediate node on the path stores a relevant routing state with the path id when it receives the NVREP.

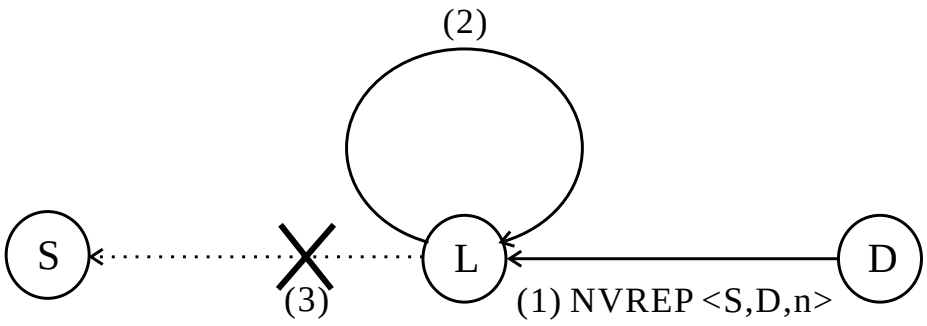


Figure 6. Loop detection in DNVR.

If a node, say L, finds a state already existing with the same path id while processing the NVREP, it assumes a loop is detected and does not forward the NVREP toward the requester S. The routing states that have been stored at other nodes between L and D on the path are never brought up to the active state and eventually removed. Thus, any route with loops is neither found nor used in DNVR.

Figure 7 shows a conceptual diagram of the NV creation process. The source node S issues a NVREQ toward the destination node D. Each intermediate node forwards the NVREQ, and it finally reaches D by this flooding process. As soon as D receives the NVREQ, it replies with a NVREP to S. Each intermediate node that receives the NVREP stores the NV for D and forwards it to the previous hop. S finally receives the NVREP and consequently stores the requested NV for D.

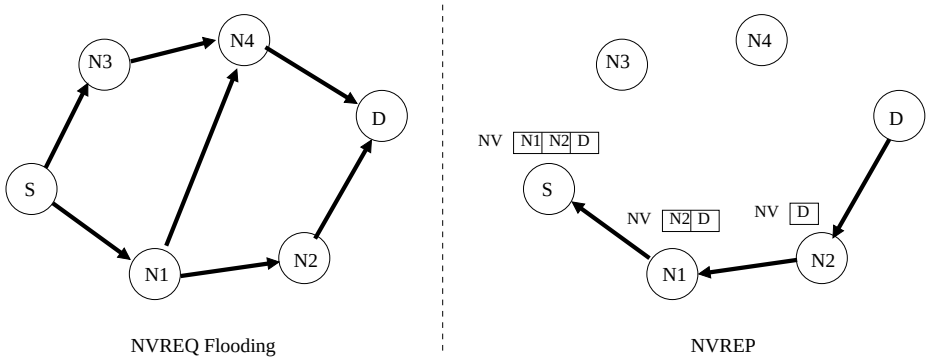


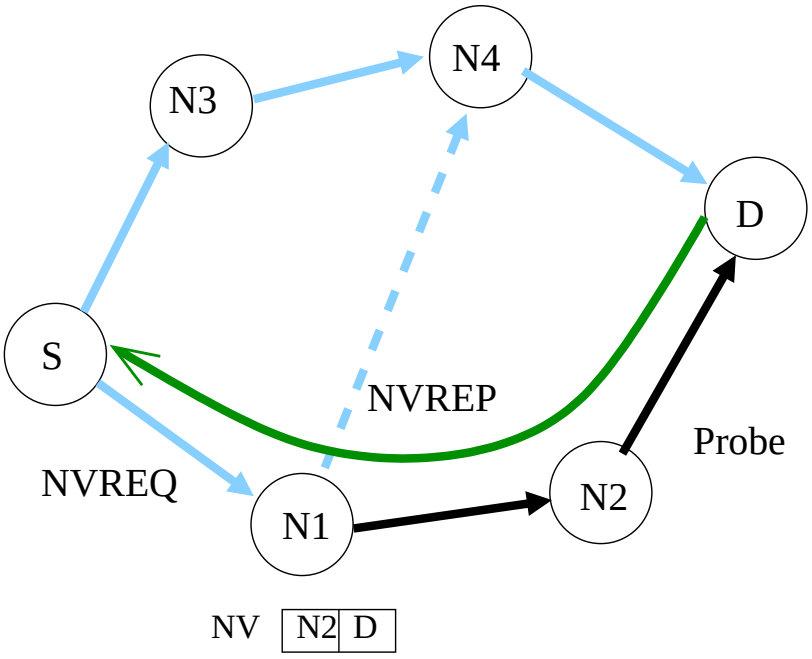
Figure 7. NV creation process by NVREQ flooding and NVREP.

2.7 Validation of Stored Route Information

When a NVREQ arrives at a node that is not the request target but does have a NV for the target, DNVR shows a different behavior from other on-demand routing protocols. Instead of immediately replying to the request, the intermediate node forwards the request by *unicasting* to the target using the stored NV. By doing so, the stored route information is validated before it can be used, and the up-to-date network topology can be probed. On the other hand, the immediate reply relies on the stored route state that may be inaccurate, and it can incur additional packet drops and delays. In DNVR, the unicast NVREQ message serves as a *probe*.

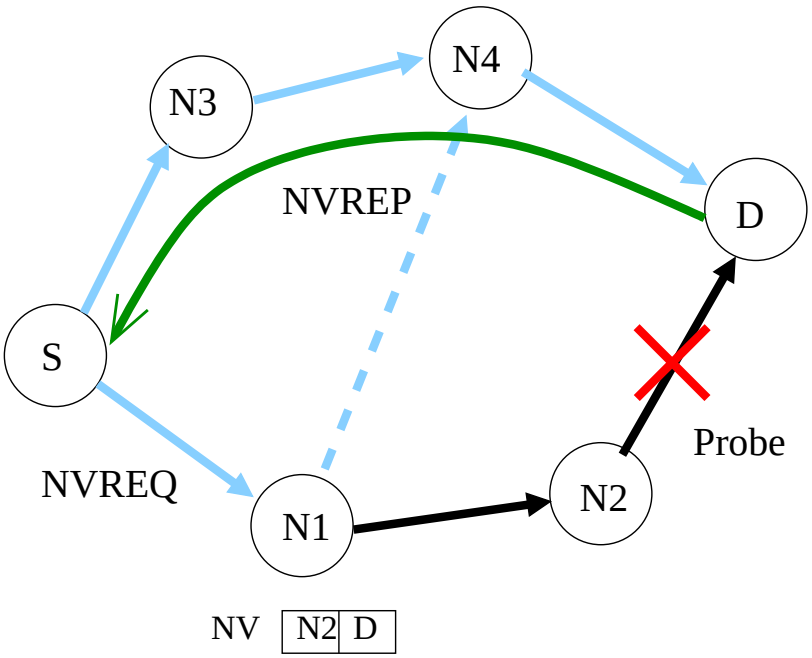
If the probe packet carrying the NVREQ successfully reaches the target, it means that the probed path is still valid and the corresponding NV can be used for routing. Figure 8 shows an example for this case. In the example, the node $N1$ received a NVREQ for the destination D, and it has a NV for D. $N1$ then unicasts a probe to D using the stored NV. For this example, the probed path is still valid. Thus, the probe will arrive at D, and consequently D will send a NVREP to S along the probed path.

Otherwise, that path is no longer valid, and thus other paths should be found. Figure 9 shows an example for this case. In the example, the node $N1$ has a NV for D and received a NVREQ for the destination D. $N1$ then sends a probe to D using the store NV. The probed



Valid Route

Figure 8. Validation of stored routes (a valid route).



Invalid Route

Figure 9. Validation of stored routes (an invalid route).

path is not valid in this example because the link between $N2$ and D is broken. Therefore, the probe will not reach D. However, D receives a NVREQ through another path, S–$N3$–$N4$–D. Thus, a NVREP is replied back to S, and S can send packets through the newly discovered path.

Moreover, with this method, more accurate route metrics (other than hop count) can be obtained since every reply comes from the target through all the intermediate nodes on the path.

It appears at first glance that this different behavior of validating the stored route information will take additional time to acquire a route and consequently cause more packet latency. Even though it might slightly increase the route acquisition time, it does not contribute to the overall delay performance of the protocol very much while resulting in higher packet delivery performance. This is verified through the simulation, which is presented later.

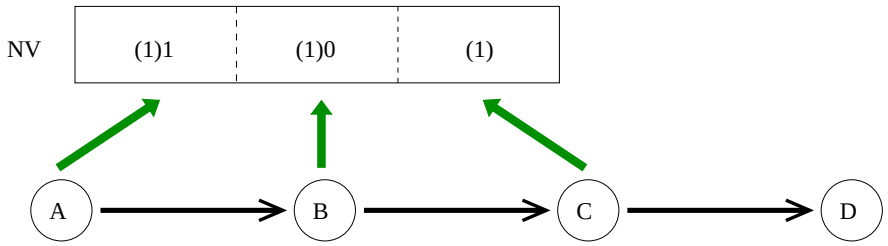
2.8 Routing and Mobility Management

2.8.1 Routing Using NVs

When a node receives a data packet with a NV, it first reads the four-bit color field from the NIx in the NV. Let us call the value of the color field N_b. The node then reads $N_b - 1$ bits afterward. The hidden bit 1 is then prepended to the extracted bits, and the resulting bits constitute the neighbor index for the next-hop neighbor. This neighbor index returns the MAC address of the next-hop node's interface from the neighbor table. After the node gets the MAC address, it forwards the packet to the corresponding neighbor by link layer unicast if the packet is not destined for the node.

Figure 10 shows an example of routing a data packet using a NV. In the example, the color field of each NIx in the NV is omitted for simplicity. Thus, the value specified in each NIx field is the neighbor index. The parenthesized number represents the hidden bit. The node A received a data packet with a NV. A then reads its portion of NIx `(1)1` from the NV, which returns B as the next hop from its neighbor table. Thus, A forwards the packet to B.

In the same way, the packet is sent from *B* to *C*, and then from *C* to *D*, the final destination of the packet.



Index	Neighbor
1	–
2	–
3	B
4	–

Index	Neighbor
1	–
2	C
3	–
4	–

Index	Neighbor
1	D
2	–
3	–
4	–

Figure 10. Routing using a NV.

2.8.2 Mobility Management

The mobility management function of DNVR can be decomposed into two parts: route maintenance and neighbor management. Route maintenance deals with how to detect routing failures, to report the routing errors to corresponding nodes, and to find new routes. Neighbor management is concerned with how to detect neighbors and when to add or invalidate neighbors in the neighbor table at each node.

2.8.2.1 Route Maintenance

DNVR assumes notification of packet transmission failure from the link layer. In other words, it utilizes the link-layer notification function in case a packet transmission fails. However, if this link-layer feedback function cannot be supported for some reasons, DNVR can optionally utilize overheard packets to detect the routing failure.

When a node detects a broken link, it notifies relevant nodes of the failure by originating NV error (NVERR) messages. This error message contains (i) the path id for the path that

has been invalidated due to the error and (ii) the IP address of the node that detected the routing failure. The error-detecting node searches its NV-FIB for NVs containing this dead link. Each found NV is invalidated, and the source of that invalidated NV is notified with a NVERR message of the failure. To return the NVERR, the path id for the reverse path (upstream to the source) is constructed by exchanging the source with the destination part of the downstream path id, and then the error-detecting node searches its NV-FIB by the reversed path id to retrieve the NV for the upstream path. The found NV is used to forward the NVERR to the source.

When each intermediate node receives the NVERR, it first checks if it has a stored NV with the path id specified in the NVERR and then invalidates the NV if it has one. Finally, the source node becomes aware of the routing failure and invalidates the corresponding NV as soon as it receives the NVERR. The source node will invoke a new NV creation process if there are more data packets for the destination.

2.8.2.2 Neighbor Management

In DNVR, routing neighbors at each node are managed in a reactive fashion. It does not actively detect neighbors or monitor the current state of existing neighbors. The neighbors are passively detected and monitored.

A neighbor is detected and added to the neighbor table only during a NV creation phase. Once a neighbor is detected and added, that neighbor can be used as a next hop for routing. The lifetime value is refreshed whenever the neighbor is used for routing. A neighbor is invalidated when the lifetime for the neighbor expires, or the node experiences a transmission failure over the link to the neighbor. This action results from three possibilities: (i) the neighbor is no longer used for routing, (ii) the neighbor has moved out of the node's transmission range, or (iii) the link to the neighbor is highly congested. In any case, the node invalidates the neighbor. In case of (ii) or (iii), the node will experience a failure to route a packet and send a NVERR message toward the source node for the packet.

2.9 Performance Evaluation

In this section, the performance of DNVR is evaluated and compared to the well known DSR protocol, which is believed to be one of the most efficient on-demand routing protocols. The simulation models for the MAC and the routing protocols are explained, and the performance results are presented and discussed.

2.9.1 The Simulation Environment

All of the simulations were performed using the *Georgia Tech Network Simulator (GTNetS)* [48]. 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. Moreover, the simulator is implemented completely in object-oriented C++, which leads to an easy extension for new or modified behavior of existing simulation models. For more information, refer to the GTNetS web page at [49].

The distributed coordination function (DCF) of the IEEE 802.11 [46] standard was used as the MAC protocol in the simulation. Each routing protocol model has a send buffer of 64 data packets with a 30 second timeout. After the timeout, the packet is expunged from the send buffer. The send buffer holds pending packets while waiting for route replies. In addition, each wireless interface of a node has a queue known as an *interface queue*. The interface queue can hold a maximum of 50 packets. This queue gives higher priority to routing protocol messages than data packets. The DSR simulation model was developed to follow the protocol specification [39] except for the promiscuous listening capability. However, [40] reports that the performance of DSR is similar in both cases of promiscuous and non-promiscuous listening mode.

For the simulations, 150 different scenarios were constructed. Each scenario is a set of mobility patterns, traffic patterns, number of traffic flows, and pause time. The mobility model used was random-waypoint [50]. The sizes of the network simulated were 50, 100, and 200. [2] For the 50-node model, 10, 20, 30, and 40 traffic flows were used with a rate

[2] DSR is designed for networks of up to about 200 nodes [39]. For a fair comparison, experiments for

of four packets/sec. For the 100-node model, 10, 20, 40, and 50 traffic flows were used with a rate of four packets/sec, excepting in the 40 and 50 flow scenarios where three and two packets/sec were used respectively. For the 200-node model, 20 and 40 flows with a rate of four and two packets/second were used respectively. All traffic was created with a constant-bit-rate (CBR) data source, and all packets were 512 bytes.

In the simulation, each mobile node is placed within a rectangle of 1500 m $\times$ 300 m for the 50-node model, a rectangle of 2200 m $\times$ 600 m for the 100-node model, and a rectangle of 3200 meters $\times$ 900 meters for the 200-node model. Initially, a mobile stays in a randomly selected location during the pause time, then selects a random target and moves to the destination with a uniformly distributed speed between 0 and 20 meters/second. Once it reaches the target, the mobile stays again during the pause time. Each simulation executed for 500 simulated seconds.

2.9.2 Performance Results

To reflect various aspects of the routing protocols, the following performance metrics were used:

- Packet delivery ratio (PDR)
- Packet latency (end-to-end delay) [3]
- Normalized routing overhead (packet count)
- Normalized total overhead (byte count)

The packet delivery ratio is the ratio of the number of received data packets at the destinations to the number of data packets generated by the CBR sources. This metric captures the throughput of the network. The packet latency is the average time taken to transfer a data packet from a CBR source to its target. The route acquisition time is also reflected in the packet latency.

larger networks were not performed.

[3]The ARP effect was not modeled in this simulation. It might cause additional delays for protocols that need address resolution if modeled.

The normalized routing overhead is the ratio of the number of routing messages generated by the routing protocol to the number of received data packets at the destinations. This metric is a measure of how many routing messages are needed to receive one data packet. It captures the efficiency of the routing protocol.

However, the normalized routing overhead is not enough to represent the efficiency of a routing protocol for several reasons. First, it measures the ratio by packet count. The routing message size is dynamic, and can be somewhat different from a routing protocol to another. For example, DSR uses source routing to unicast a packet, which results in large control messages. In addition, the *data packet overhead* is not considered in the routing overhead. The data packet overhead can be defined as the portion that a packet header occupies in a data packet. To capture these all together, the normalized total overhead is introduced. The total overhead is the byte count for the routing messages and the header of data packets. The header includes the IP header and the routing header of DSR and DNVR. Then, the total overhead is normalized to the byte count of the received data packets. Hence, the normalized total overhead is better at representing the efficiency of a routing protocol. This metric is a measure of how many bytes of routing messages and data packet headers are needed to receive a byte of data.

In the simulation results, each data point represents an average of 30 runs with different mobility patterns and traffic patterns. To insure a fair comparison, however, an identical set of mobility and traffic patterns was applied to both routing protocols in each simulation run. Each result was plotted over the *mobility factor*. The mobility factor was introduced to represent the network mobility with respect to the pause time. It is defined as $(\textit{simulated time} - \textit{pause time})/\textit{simulated time}$. Thus, the value 0 means no mobility, and the value 1 the highest mobility.

2.9.2.1 50-Node Model

As can be seen in Figure 11 and Figure 12, the overall performance of DSR and DNVR is similar for 10 and 20 traffic flows. DNVR shows slightly higher PDR at zero pause time in

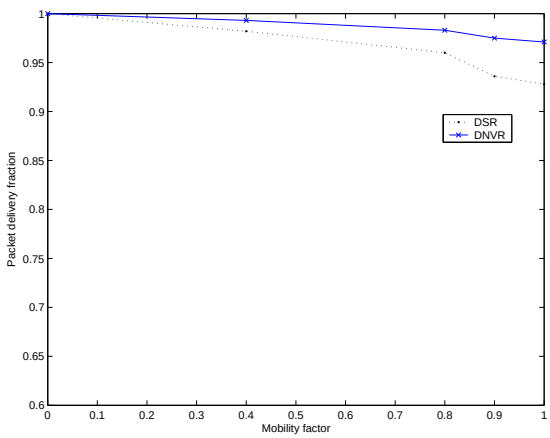
both 10 and 20 flow cases. But both protocols show above 90 % PDR in every case. Also, the delay performance of each protocol is very similar for 10 and 20 flows.

As can be seen in Figure 13(a), DNVR outperforms DSR with mid to high mobility in terms of PDR for 30 flows. DNVR shows about 16 % higher PDR than DSR at zero pause time. This result shows that DNVR scales better to a larger number of traffic sources with high mobility than DSR in terms of PDR. This claim is supported by the simulation result from the 100-node model too. Figure 13(b) shows that DNVR demonstrates about 15 – 20 % smaller delay than DSR for 30 flows without regard to mobility.

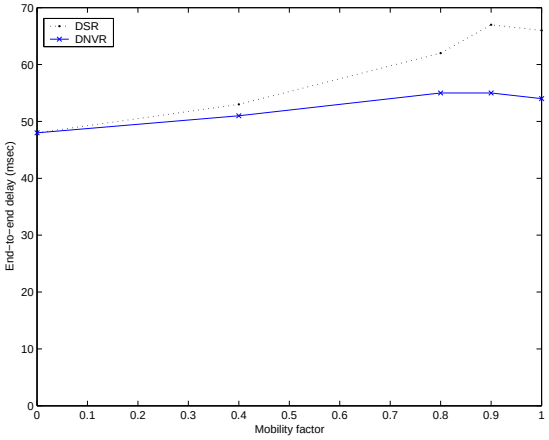
An interesting observation can be seen in Figure 13(b). As can be seen in Figure 11(b) and Figure 12(b), the end-to-end delay slightly increases as mobility increases for 10 and 20 flows. For 30 flows, however, the end-to-end delay decreases with mobility as seen in Figure 13(b). This result is due to a high congestion level of the network. The traffic flows tend to be distributed more evenly with high mobility than low mobility. Thus, the high mobility helped dissolve the network congestion in this case. The similar phenomenon is reported in [40, 55].

The routing overhead and total overhead measures show an interesting result. Figure 14(a), Figure 15(a), and Figure 16(a) show that DSR generated fewer routing messages per data packet than DNVR. In all cases, DNVR shows 1.6 – 6.2 times higher routing overhead than DSR. This is because DSR acquires as many routes as possible in one route discovery cycle. In consequence, a DSR node has much more route information than a DNVR node resulting in less frequent route requests when a route is invalidated due to mobility or congestion.

Figure 14(b), Figure 15(b), and Figure 16(b) reveal, however, that the efficiency of each protocol is similar. These figures show that DNVR is even better than DSR in terms of normalized total overhead (byte count) in all cases. DNVR shows about 7 – 17 % less overhead than DSR with mid to high mobility. Even though DSR generates fewer routing messages (in terms of packets), the routing message size is bigger than that of DNVR due

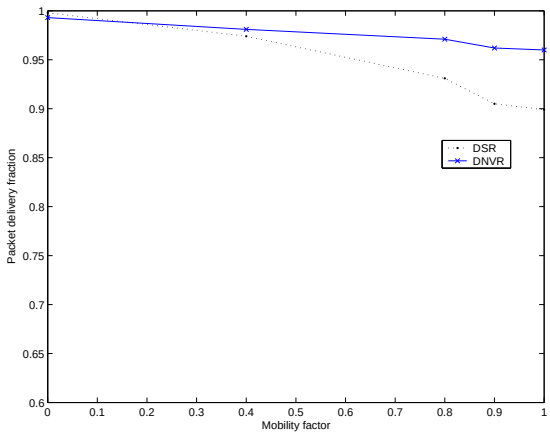


(a) Packet delivery ratio

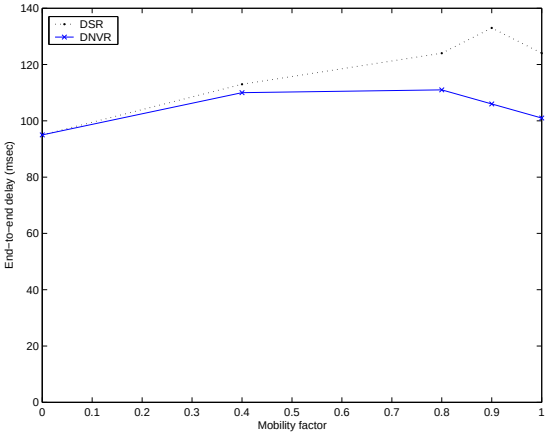


(b) End-to-end delay

Figure 11. Average PDR and delay (10 flows/50 nodes).

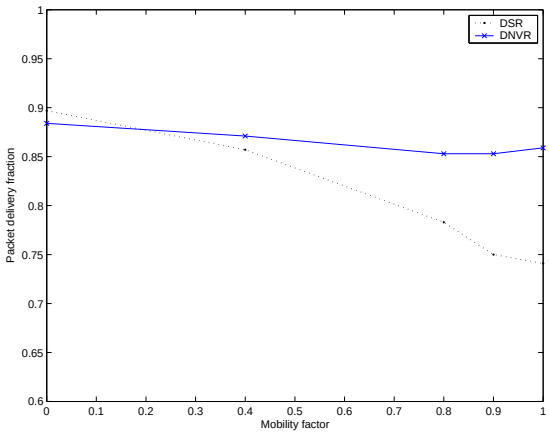


(a) Packet delivery ratio

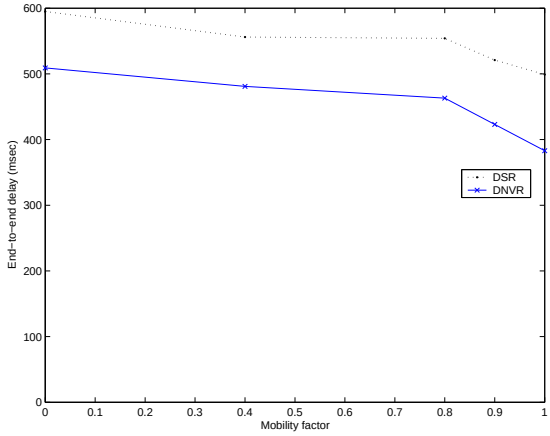


(b) End-to-end delay

Figure 12. Average PDR and delay (20 flows/50 nodes).

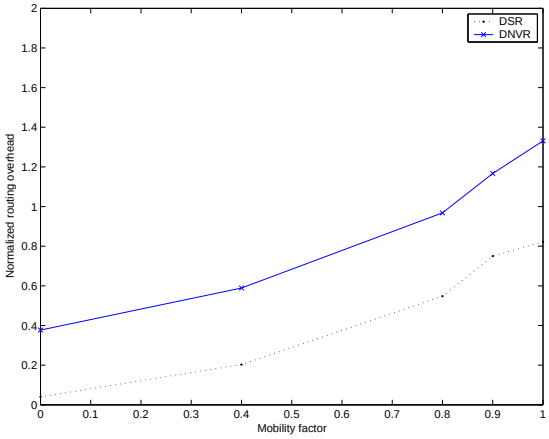


(a) Packet delivery ratio

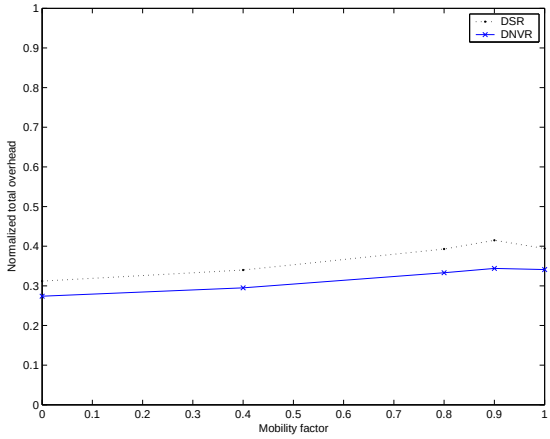


(b) End-to-end delay

Figure 13. Average PDR and delay (30 flows/50 nodes).

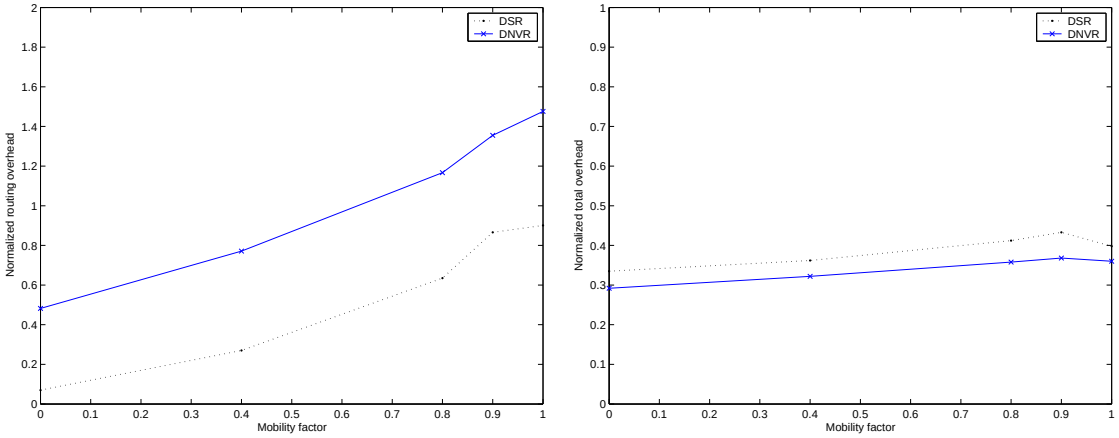


(a) Routing overhead



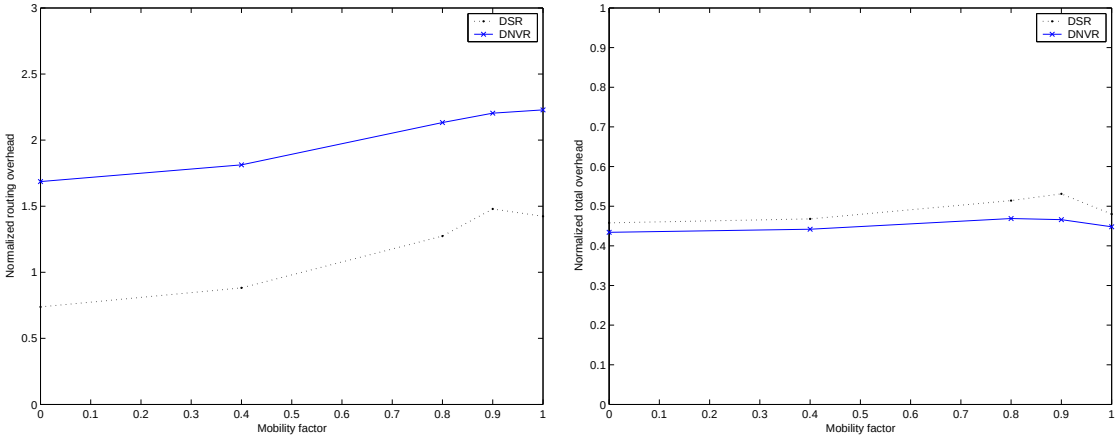
(b) Total overhead

Figure 14. Normalized routing and total overhead (10 flows/50 nodes).



(a) Routing overhead (b) Total overhead

Figure 15. Normalized routing and total overhead (20 flows/50 nodes).



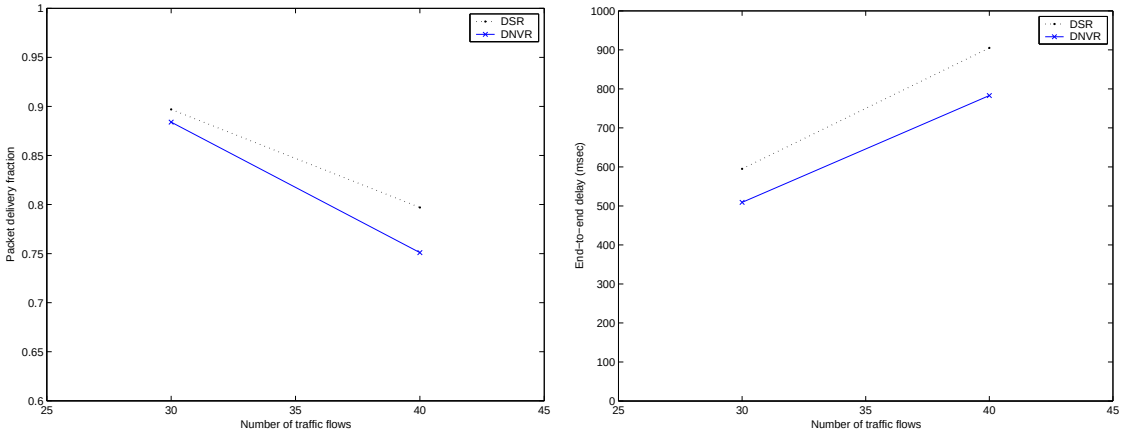
(a) Routing overhead (b) Total overhead

Figure 16. Normalized routing and total overhead (30 flows/50 nodes).

to the source routing header of DSR messages. The same logic applies to the data packets. In general, a DSR data packet has a much longer routing header than that of DNVR. These facts cause somewhat higher total overhead in DSR compared to DNVR.

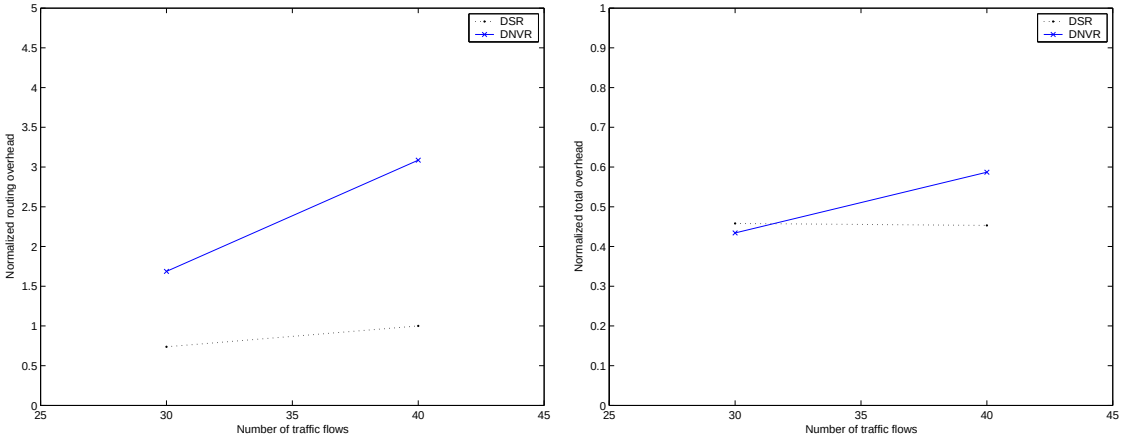
Figure 17 shows average packet delivery ratio and latency of each protocol under a static network with a different number of traffic flows. As can be seen in the figure, the network is highly congested. Under this circumstance, as can seen in Figure 17(a), DSR outperforms DNVR in terms of PDR. There is no mobility-induced packet loss because the network is static. Thus, every packet loss is due to congestion. When a packet loss event occurs, DSR can utilize other paths to route packets with no additional overhead because a DSR node obtains as many routes as possible in a single route discovery phase. This can be beneficial especially when a network is static because all paths remain topologically valid. As shown in Figure 17(b), DNVR shows smaller delays than DSR. The increased delay of DSR partly attributes to its packet salvaging capability. With no mobility, it is more likely that a DSR node has multiple valid routes to a destination, which incurs frequent packet salvaging.

Figure 18 shows normalized routing load and total overhead of each protocol. It can be observed, in Figure 18(a), that the routing load of DSR remains relatively stable while that of DNVR grows with offered traffic load. The total overhead shows the similar tendency as can be seen in Figure 18(b). Under a highly congested environment, as explained above, DSR can benefit from multiple routes for a destination without incurring additional overhead because all paths are found in a single route discovery phase and remain valid as there is no topology change due to mobility. On the other hand, DNVR tries to find another path in case of packet loss and issues many RREQ packets, increasing its routing overhead. However, this is a very special case because a mobile network is usually subject to topology changes resulting from mobility, and a high degree of network-wide congestion due to overly offered traffic load is not a normal phenomenon.



(a) Packet delivery ratio (b) End-to-end delay

Figure 17. Average PDR and delay (static/50 nodes).



(a) Routing overhead (b) Total overhead

Figure 18. Normalized routing and total overhead (static/50 nodes).

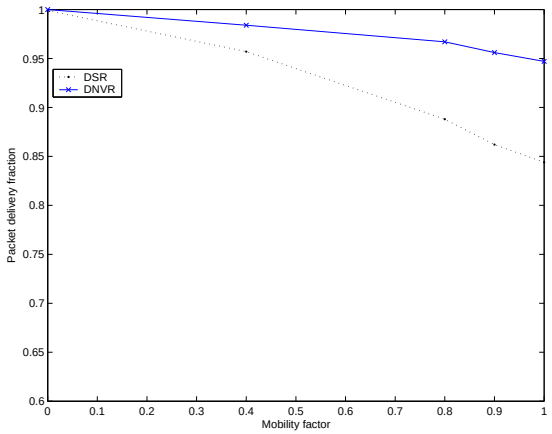
2.9.2.2 100-Node Model

For the 100-node model, DNVR demonstrates better performance in terms of PDR compared to DSR with mid to high mobility. With low mobility, both protocols show very similar packet delivery performance. As shown in Figure 19(a), Figure 19(c), Figure 20(a), and Figure 20(c), the PDR of DSR drops more steeply than that of DNVR as mobility increases. The PDR differential between two protocols gets even bigger with the number of traffic flows. At zero pause time, DNVR shows about 12 % higher PDR than DSR for 10 flows, and about 30 % higher for 40 and 50 flows. This result supports the claim made in the previous section that DNVR scales better than DSR to a large number of traffic flows in terms of packet delivery performance.

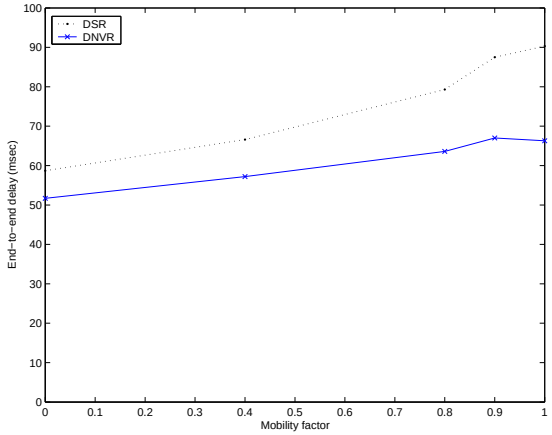
As can be seen in Figure 19(b) and Figure 19(d), both protocols show very similar delay performance for 10 and 20 flows. Figure 20(b) and Figure 20(d) show that, for 40 and 50 flows, DNVR demonstrates smaller delay than DSR with high mobility, but bigger delay with low mobility. DNVR shows about 16 % smaller delay than DSR at zero pause time, but about 18 % bigger delay with no mobility in 40- and 50-flow cases. However, the overall delay performance of each protocol is very similar on average.

As shown in Figure 21(a) and Figure 21(c), DSR shows somewhat lower routing overhead than DNVR for 10 and 20 flows. Figure 22(a) and Figure 22(c) show that, for 40 and 50 flows, DSR demonstrates much lower routing overhead than DNVR in terms of packets. As explained in the previous section, however, both protocols show similar total overhead performance in terms of bytes. As can be seen in Figure 21(b), Figure 21(d), Figure 22(b), and Figure 22(d), DNVR has somewhat lower total overhead than DSR for 10 and 20 flows while DSR shows a little bit lower total overhead compared to DNVR for 40 and 50 flows. On average, DNVR shows 17 % lower overhead than DSR for 10 and 20 flows, and DSR shows 9 % lower overhead than DNVR for 40 and 50 flows.

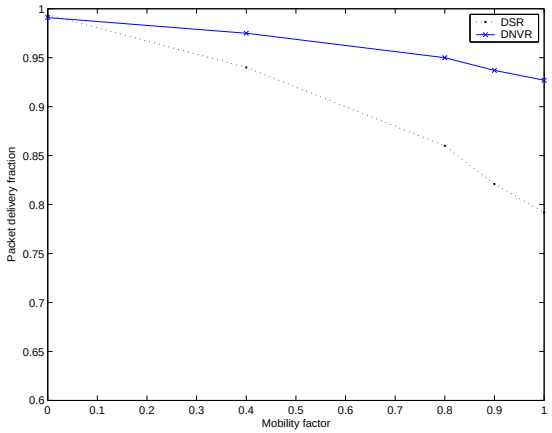
The reason that DNVR has somewhat higher total overhead than DSR for 40 and 50 flows can be explained as follows. The PDR differential of two protocols gets bigger for



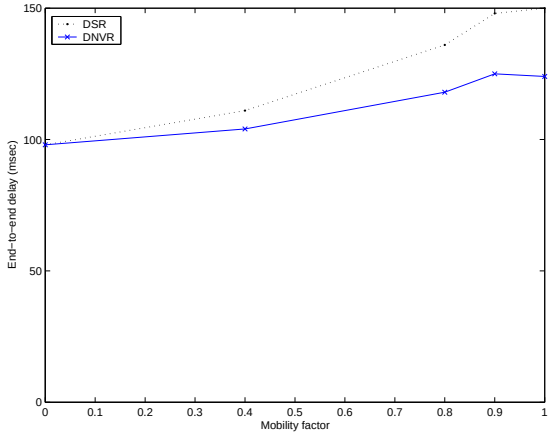
(a) Packet delivery ratio (10 flows)



(b) End-to-end delay (10 flows)

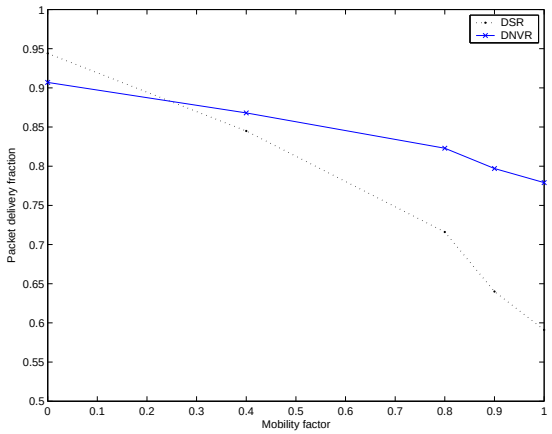


(c) Packet delivery ratio (20 flows)

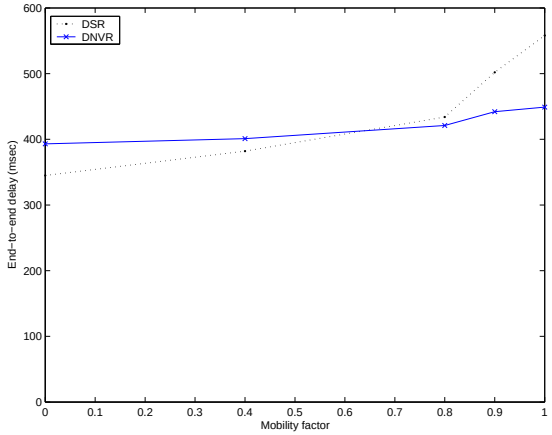


(d) End-to-end delay (20 flows)

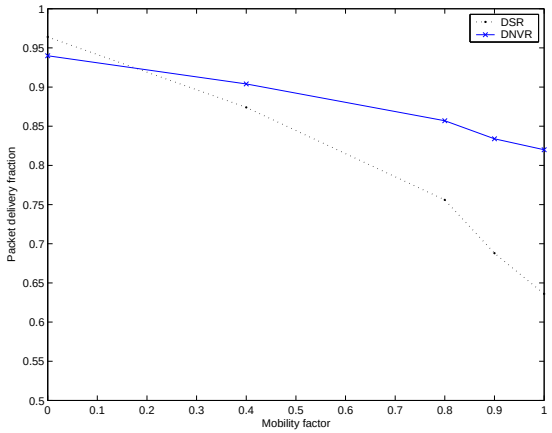
Figure 19. Average PDR and delay (10 and 20 flows/100 nodes).



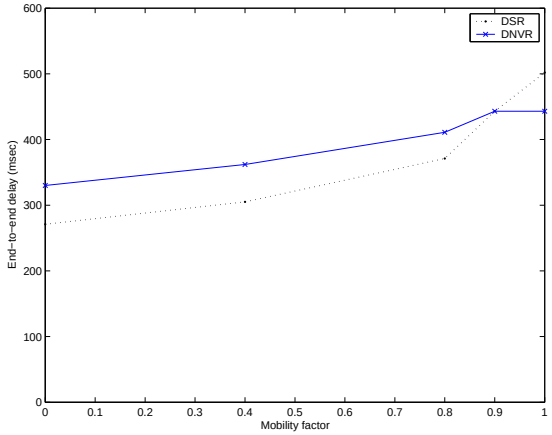
(a) Packet delivery ratio (40 flows)



(b) End-to-end delay (40 flows)

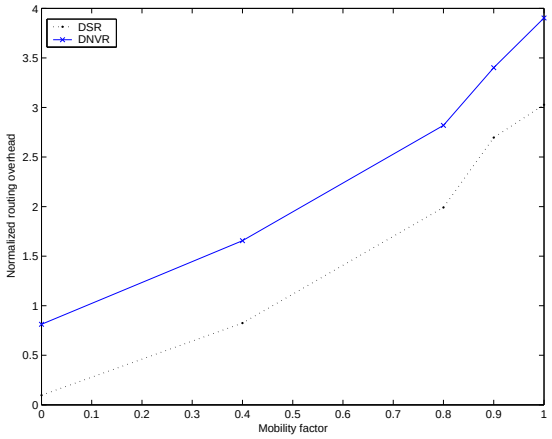


(c) Packet delivery ratio (50 flows)

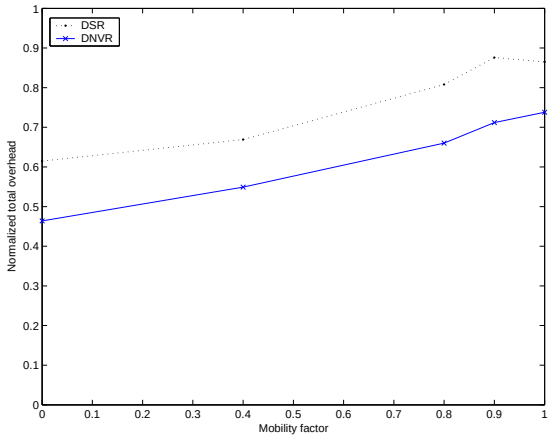


(d) End-to-end delay (50 flows)

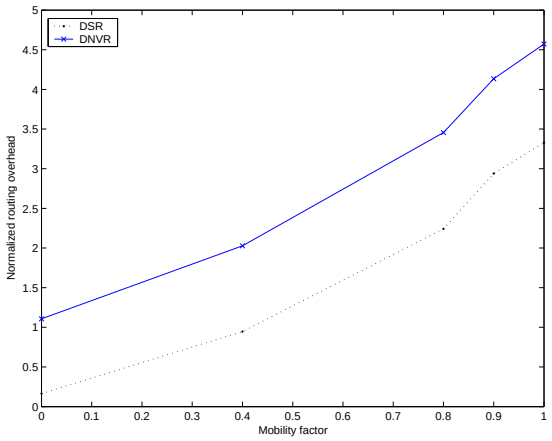
Figure 20. Average PDR and delay (40 and 50 flows/100 nodes).



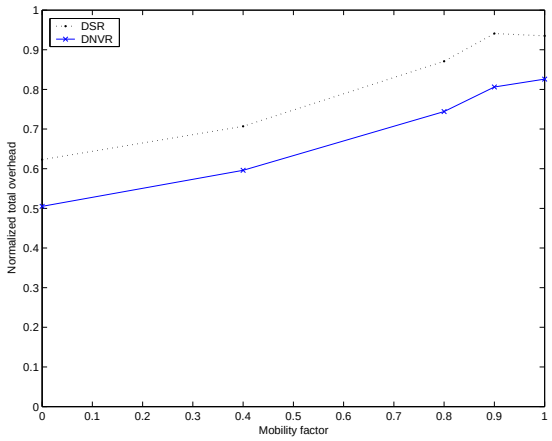
(a) Routing overhead (10 flows)



(b) Total overhead (10 flows)

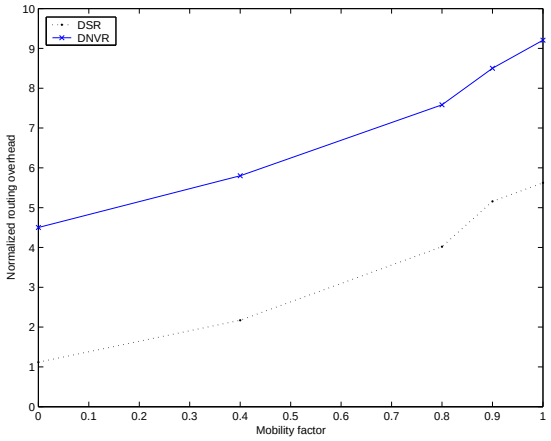


(c) Routing overhead (20 flows)

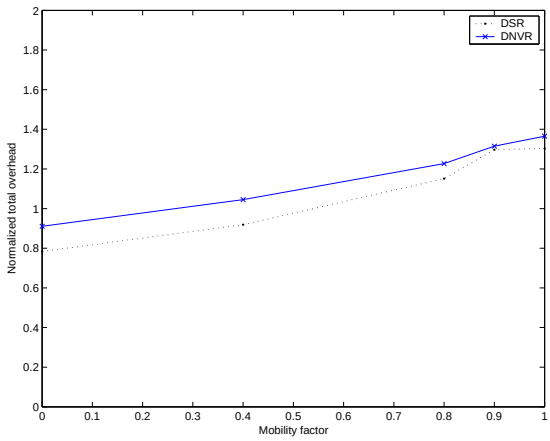


(d) Total overhead (20 flows)

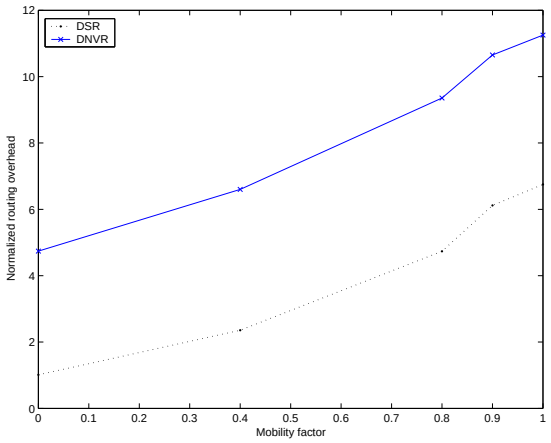
Figure 21. Normalized routing and total overhead (10 and 20 flows/100 nodes).



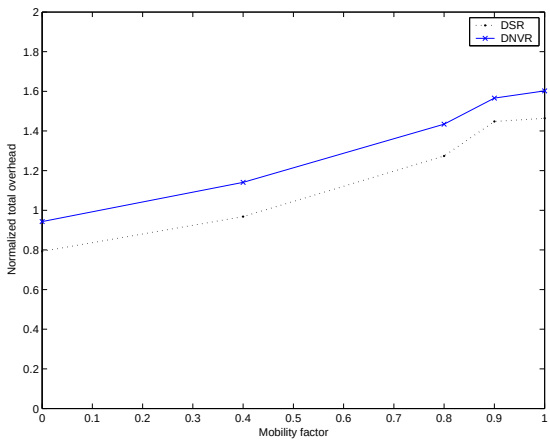
(a) Routing overhead (40 flows)



(b) Total overhead (40 flows)



(c) Routing overhead (50 flows)



(d) Total overhead (50 flows)

Figure 22. Normalized routing and total overhead (40 and 50 flows/100 nodes).

40 and 50 flows than for 10 or 20 flows. For example, the PDR of DNVR at zero pause time is about 12 % higher than that of DSR for 10 flows, but about 30 % higher for 40 and 50 flows. With that, the data packet overhead tends to contribute more to the total overhead with PDR increase. Given these, DNVR has much more data packets delivered, and thus has a bigger portion of data packet overhead in the total overhead than DSR in this case, which ends up with somewhat higher total overhead in DNVR. But as can be seen in Figure 22(b) and Figure 22(d), the differential is small.

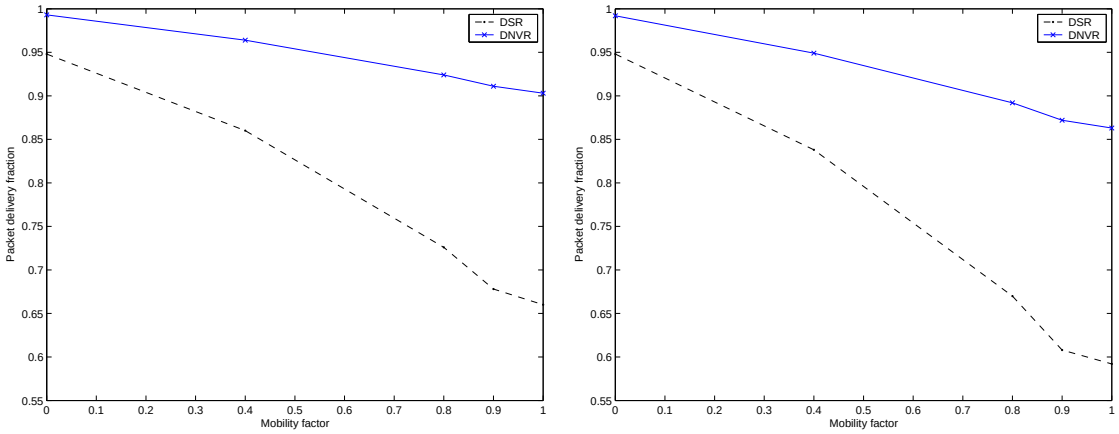
An interesting observation in this experiment is that DSR shows relatively stable performance in terms of packet delivery ratio and delay, contrary to the simulation result from [55]. That work reports that DSR suddenly becomes unstable with a large number of nodes and traffic sources, which was not observed in these experiments.

2.9.2.3 200-Node Model

As shown in Figure 23, both protocols show above 90 % PDR with no mobility. As the network mobility increases, however, the two protocols show much different aspects. The PDR differential becomes even bigger as the network mobility and the traffic volume increase, which shows that DNVR scales better than DSR with respect to mobility and traffic volume. With the highest mobility, the PDR of DNVR is 37 % higher than that of DSR for 20 flows and 46 % higher for 40 flows. On average, DNVR demonstrated 23 % higher PDR than DSR.

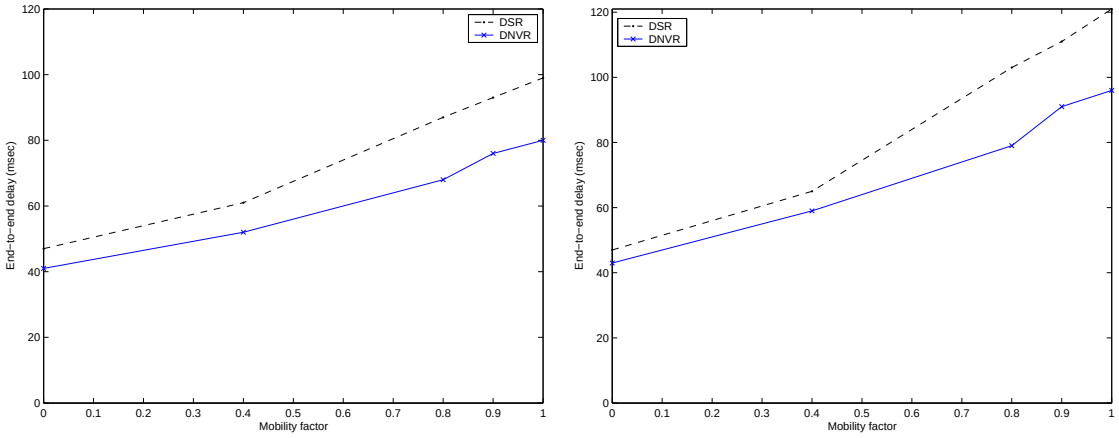
As can be seen in Figure 24, DNVR shows smaller packet latency for every mobility factor. The delay performance gap becomes larger as mobility increases. DNVR demonstrated up to a 23 % smaller delay than DSR. On average, DNVR showed about an 18 % smaller delay than DSR. It also turned out that DNVR consumed about 35 % less time to acquire routes than DSR. But the portion of each protocol's route acquisition time out of the total delay was similar, which was around 30 %.

Figure 25 shows routing overhead for 20 and 40 traffic flows, and it can be observed that the overhead increases with mobility for both protocols. This results from the increased



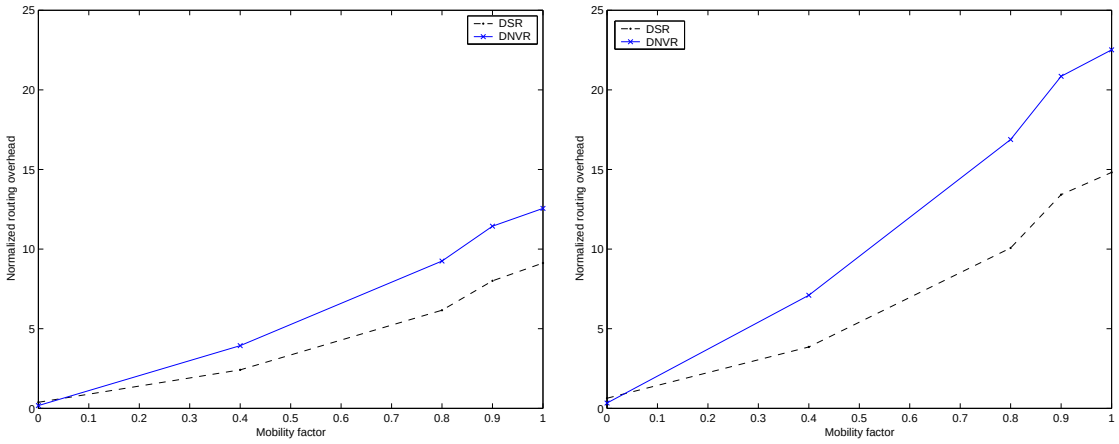
(a) 20 flows (b) 40 flows

Figure 23. Average PDR (20 and 40 flows/200 nodes).



(a) 20 flows (b) 40 flows

Figure 24. Average delay (20 and 40 flows/200 nodes).



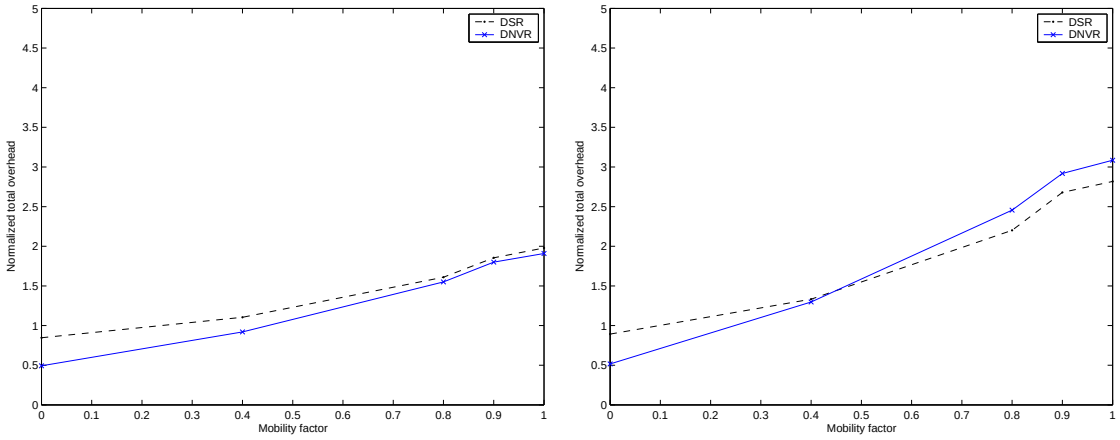
(a) 20 flows (b) 40 flows

Figure 25. Normalized routing overhead (20 and 40 flows/200 nodes).

number of mobility induced route failures and consequent routing messages. However, the overhead of each protocol shows a different aspect regarding the number of traffic flows. The differential of the overhead gets bigger with the number of flows. For DSR, existence of more flows means population of more route information, which can increase hit rate of the route cache in a node. The effect of this becomes salient when there are more route requests. Thus, DSR issued relatively less routing messages than DNVR in higher mobility with 40 flows compared to the case with 20 flows.

Figure 26 reveals that the efficiency of both protocols is very similar. As can be seen in Figure 26(a), the total overhead of DNVR is somewhat lower than that of DSR for 20 flows without regard to mobility. Figure 26(b) shows that, for 40 flows, DNVR produces a little bit lower overhead than DSR with mid mobility and slightly higher overhead with high mobility.

In the presence of high mobility and a large volume of traffic, it is possible that the network seems partitioned temporarily to the routing protocol when it fails to find a route to a specific destination. In this case, on-demand routing protocols attempt route request retries repeatedly. DSR experiences these route discovery failures less frequently than



(a) 20 flows (b) 40 flows

Figure 26. Normalized total overhead (20 and 40 flows/200 nodes).

DNVR due to the high hit rate of route caches even though these replies from the route caches contain stale routes. On the other hand, DNVR issues and consequently propagates more route request messages in this case. This is the reason that DNVR shows a little bit higher total overhead in this case. On average, DNVR showed a 10 % lower overhead than DSR for 20 flows and 4 % higher for 40 flows. Overall, both protocols demonstrated very similar protocol efficiency without regard to mobility and traffic volume.

2.10 Conclusions

In this chapter, a new routing method for mobile ad hoc networks was presented. DNVR behaves in a reactive fashion to acquire loop-free routes and maintain them as do other reactive routing protocols. However, it is differentiated from other existing on-demand routing protocols in that (i) the stored route information is validated before being used, and the up-to-date network topology is probed in an efficient way during NV creation, (ii) the routing states are managed in a timely fashion, (iii) a conservative route discovery strategy is adopted by suppressing route requests, and thus only a few routes are maintained per destination, (iv) address resolution is obviated by using a NIx and MAC addresses for

routing purposes, and (v) a NV routing header is a compact form of source route, which significantly reduces the packet overhead resulting from source routing. By virtue of these features, DNVR provides more stable and scalable performance with respect to network size, mobility, and traffic volume.

It was shown via simulation that the new method scales well to a large network under various scenarios, especially with a high degree of mobility and a large volume of traffic. DNVR was compared to DSR through extensive simulations, and it was shown that DNVR is as efficient as DSR in terms of normalized total overhead while achieving higher packet delivery and smaller packet latency in most cases. DNVR can be another design choice among the existing on-demand routing protocols for mobile ad hoc networks in that it is designed to provide scalability and high bandwidth efficiency as well as low overhead.

CHAPTER 3

A WORKLOAD-BASED ADAPTIVE LOAD-BALANCING TECHNIQUE FOR MOBILE AD HOC NETWORKS

3.1 Introduction

Mobile wireless ad hoc networks usually consist of mobile nodes that are not reachable through a single hop. Therefore, the main objective of ad hoc routing protocols has been to pursue the wireless multi-hop routing capability. The wireless links in these networks usually have lower capacity than wired links. Hence, congestion is a normal phenomenon rather than an exception in mobile ad hoc networks.

Currently existing protocols for ad hoc routing lack load-balancing capabilities. Thus, they often fail to demonstrate good performance especially in the face of heavy traffic as the network load concentrates on some nodes resulting in a highly congested environment. Congestion causes several undesirable results such as long packet latency, poor packet delivery, and high routing overhead. It also results in excessive consumption of the network resources such as bandwidth and power that are usually scarce in these networks.

In this chapter, a simple but very effective method is introduced to achieve load balance and congestion alleviation in a completely distributed way. The new scheme is motivated by the observation that ad hoc on-demand routing protocols flood route request (RREQ) messages to acquire routes, and only nodes that respond to those messages have a potential to serve as intermediate forwarding nodes.

Even if each routing protocol has different features from another, most on-demand protocols share a common route discovery mechanism, which is due to their on-demand behavior. Ad hoc on-demand protocols mainly rely on flooding to find a path to a requested destination. They issue a RREQ message toward the destination when a source node does not have a valid route to the destination, and each node that receives the request responds by forwarding it. This forwarding action leaves a state in the RREQ message (source routes)

or in the forwarding node itself (routing tables) that is used to compute a route. If a node does not join this RREQ forwarding action within a specific period, it can completely be excluded from the additional communications that might have occurred for that period otherwise. Thus, a node can decide not to serve a traffic flow by ignoring the RREQ for that flow.

In the new scheme, RREQ messages are forwarded selectively according to the load status of each node. Overloaded nodes do not allow additional communications to set up through them so that they can be excluded from the requested paths within a specific period. Each node begins to allow additional traffic flows again whenever its overloaded status is dissolved.

The new scheme utilizes interface queue occupancy and workload to control RREQ messages adaptively. Each node maintains a threshold value, which is a criterion for deciding whether or not to respond to a RREQ message. The threshold value dynamically changes according to the load status of a node based on its interface queue occupancy and its workload within a specific period.

In on-demand routing protocols, not every node that has responded to a RREQ is guaranteed to be on the discovered path. This is because there usually exist multiple paths for the same source-destination pair in a mobile ad hoc network. However, it is obvious that only nodes that have joined the RREQ forwarding action could have a potential to be on the found path. In other words, a node can completely be excluded from a path if the node drops the RREQ in a route discovery phase for the path. In normal ad hoc routing protocols, this does not happen as long as a node receives a RREQ.

The new scheme enables a node to join the RREQ forwarding action selectively. Each node maintains a threshold value. This threshold value is a criterion for each node's decision of how to react to a RREQ message. When a node receives a RREQ, the node takes a simple action according to the threshold value. If the interface queue length of a node is greater than the threshold value, the node simply drops the RREQ. Otherwise, the node

forwards the RREQ by rebroadcasting it. By doing so, additional traffic flows are not allowed to set up through overloaded nodes, and therefore, the overloaded nodes are naturally excluded from the newly requested paths.

The threshold value is initially set to a pre-determined value. The threshold value keeps changing according to the load status of a node. If a node experiences overload to an extent, its threshold value decreases. When the node senses that its load has been low for a long enough period, it is considered an indication that the node's overloaded status is dissolved, and its threshold value returns to the initial value. From that point on, the node allows additional communications to set up through it as long as it is not overloaded. The detailed operations of the new scheme are presented in the following section.

Figure 27 shows an example for normal forwarding of RREQ. In the example, the node $N1$ is already serving two traffic flows. After a while, there is a RREQ from $S2$. With normal RREQ forwarding, $N1$ can reply with a RREP to $S2$ as soon as it receives the RREQ. Finally, all the traffic will concentrate on $N1$, and the node can be overloaded causing congestion.

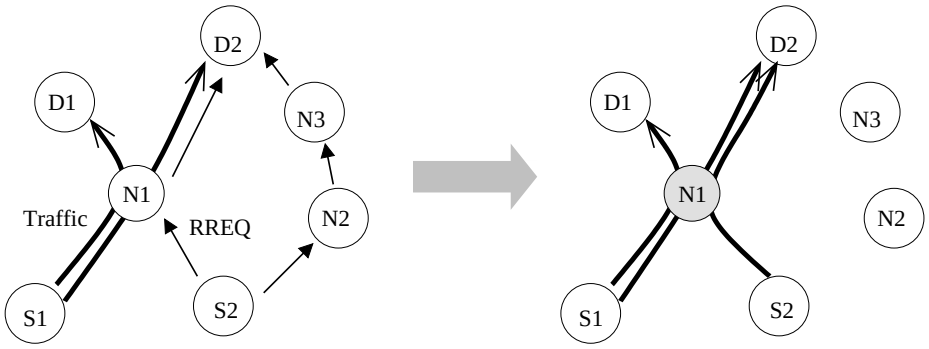


Figure 27. Normal forwarding of RREQ.

Figure 28 shows an example for selective forwarding of RREQ. In this example also, two traffic flows are already passing through the node $N1$, and there is a RREQ from $S2$. In this case, however, $N1$ ignores the RREQ from $S2$. Finally, the new traffic flow will detour

without going through $N1$. Thus, the network traffic can be more evenly distributed with the selective forwarding of RREQ.

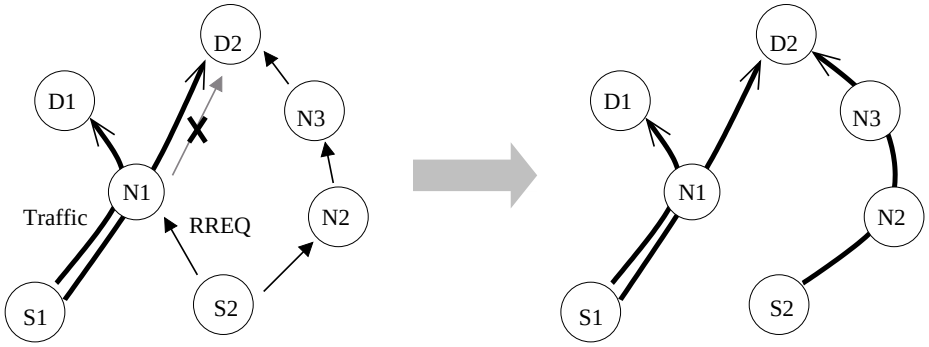


Figure 28. Selective forwarding of RREQ.

3.2 Detailed Operations

In this work, the *workload* of a node is defined as the area under the graph when the interface queue length of the node is plotted over time. Thus, the unit of workload is packet · seconds. This workload is incremental since the area is accumulated over time. In the new scheme, the queue occupancy and the workload increment are used as input parameters for calculating the threshold value.

The threshold value of a node is initially set to the maximum threshold value (max_{th}), and a node is not allowed to have the threshold value greater than max_{th} or less than the minimum threshold (min_{th}). The minimum and the maximum threshold values are pre-determined.

The threshold value of a node ranges from min_{th} to max_{th} as described above. Hence, if the queue length at the moment a node received a RREQ is greater than max_{th} or less than min_{th}, the RREQ is dropped or forwarded deterministically. However, if the queue length is between min_{th} and max_{th}, the threshold value is updated first, and then a corresponding decision is made according to the updated threshold. This situation is detailed in Figure 29.

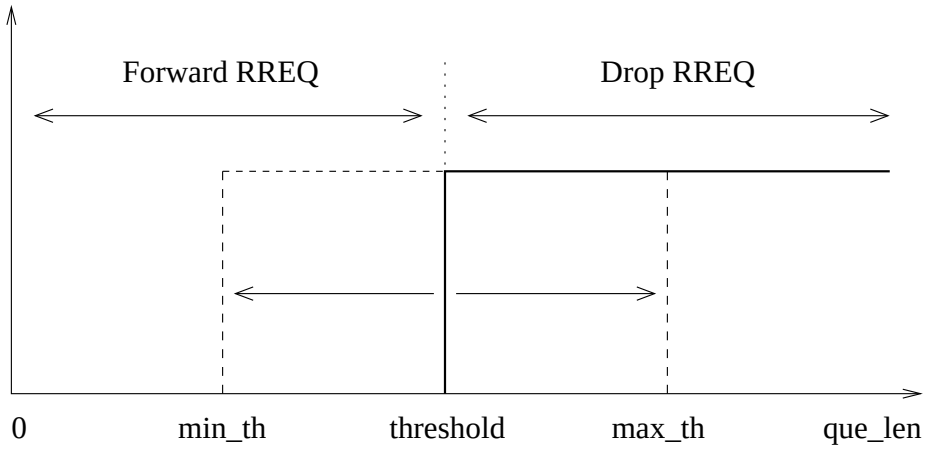


Figure 29. Adaptive threshold adjustment.

It is reasonable for a node to drop RREQs more frequently if the node experiences severer overload, and this feature can be achieved by making the possibility of dropping a RREQ higher, i.e., by decreasing its threshold value gradually as the node's overloaded status lasts.

In the new scheme, a node is considered overloaded if the following conditions are met simultaneously:

- The current queue length is greater than the average of min_{th} and max_{th}.
- The *outstanding workload* ($workload_{out}$) is greater than the workload threshold ($workload_{th}$).

The outstanding workload is the workload increment within the specific period in which the first condition is satisfied. The queue occupancy information alone is not enough to evaluate the load status of a node because the queue length can be high for a short period in a transient situation even though the node is not overloaded. The outstanding workload is the mixed information of the length and the residence time of packets in the interface queue. Thus, using both pieces of information can prevent a wrong decision on the node's load status, which is the reason that the queue length and the outstanding workload are used herein together. When a node is considered overloaded, its threshold value is decremented by the amount of $thresh_{dec}$.

A node reverts its threshold value to raise the possibility of forwarding a RREQ when its overloaded status is considered dissolved. If the queue length of a node has been less than min_{th} for at least $dissolve_{th}$ seconds, the threshold value of the node returns to the initial value.

Each node maintains a small number of states to update its threshold value. The threshold value is updated whenever the node receives a RREQ message. The threshold update algorithm is detailed in Figure 30.

Even though effective suppression of RREQs can be beneficial to load balance and congestion alleviation, excessive suppression can lead to partition in a sparse network. To deal with this problem, a *priority flag* is introduced in a RREQ to make each node differently process RREQs according to the flag. Each node unconditionally forwards a RREQ if the flag in the RREQ is set to one.

For example, the priority flag can be set to zero for the first cycle of route discovery so that each node can operate in a selective RREQ forwarding mode. If the try fails, then the source can attempt again with the flag set to one, which forces unconditional forwarding of the RREQ.

3.3 An Improvement: Considering Link-Layer Information

In a wireless network, congestion is a regional phenomenon since a wireless link is shared among multiple nodes. In some cases, therefore, it is not effective enough to consider only queue occupancy information. For example, when a node shares a wireless link with an overloaded node, the node is likely to experience high delays even though its queue occupancy is low.

Considering link-layer information might be helpful to solve this problem. For example, MAC contention information can be used in an IEEE 802.11 network. In this chapter, *MAC utilization* is used to capture the *busy-ness* of a communication channel. Each node monitors its channel status and logs the MAC utilization. It is considered in an IEEE 802.11

Initialize:
$threshold \leftarrow max_{th}$
$load\_state \leftarrow 0$
$time_{updated} \leftarrow 0$
$workload_{prev} \leftarrow 0$

Update:
$time_{elapsed} \leftarrow now - time_{updated}$
$workload_{current} \leftarrow workload_{prev} + que\_len \times time_{elapsed}$
if $que\_len > (min_{th} + max_{th})/2$ **then**
  if $load\_state \leq 0$ **then**
    $load\_state \leftarrow 1$
    $time_{updated} \leftarrow now$
    $time_{elapsed} \leftarrow 0$
  end if
  if $time_{elapsed} > 0$ **then**
    increase $workload_{out}$ by $(workload_{current} - workload_{prev})$
  else
    $workload_{out} \leftarrow 0$
  end if
  $workload_{prev} \leftarrow workload_{current}$
  if $workload_{out} > workload_{th}$ **then**
    $threshold \leftarrow max(threshold - thresh_{dec}, min_{th})$
    $time_{updated} \leftarrow now$
    $workload_{out} \leftarrow 0$
  end if
else if $que\_len < min_{th}$ **then**
  if $load\_state \geq 0$ **then**
    $load\_state \leftarrow -1$
    $time_{updated} \leftarrow now$
    $time_{elapsed} \leftarrow 0$
  end if
  if $time_{elapsed} \geq dissolve_{th}$ **then**
    $threshold \leftarrow max_{th}$
    $time_{updated} \leftarrow now$
  end if
else
  $load\_state \leftarrow 0$
end if

Figure 30. Threshold update algorithm.

network that a link is not utilized when

- the channel is idle,
- there is no pending packets in the interface queue, and
- the MAC is not in a backoff procedure.

Otherwise, the link is considered utilized. Each node maintains the moving average of its MAC utilization as a secondary factor in conjunction with the primary workload factor for the decision of whether or not to forward a RREQ.

3.4 Performance Evaluation

In this section, the performance of the enhanced versions of protocols with the new scheme is evaluated, and they are compared to the base protocols. The simulation environment is explained, and the simulation results are presented and discussed.

3.4.1 The Simulation Environment

All of the simulations were performed using GTNetS [48]. The distributed coordination function (DCF) of the IEEE 802.11 [46] standard was used as the MAC protocol with a 2 Mbps-bandwidth shared medium in the simulation.

AODV [11] and DSR [10] were chosen as the base routing protocols, and the enhanced versions with the workload-based adaptive load-balancing scheme for each base routing protocol (termed as -WAL) were implemented. Each routing protocol model has a send buffer of 64 data packets with a timeout value of 30 seconds. After the timeout, the packet is expunged from the send buffer. The send buffer holds pending packets while waiting for route replies (RREPs). In addition, each wireless interface of a node has a queue that can hold up to 50 packets. This queue gives higher priority to routing protocol messages than data packets. The parameter values used for the new scheme are specified in Table 1.

For the simulations, 30 different scenarios were constructed. Each scenario is a set of mobility patterns and traffic patterns. The mobility model used was random-waypoint [50],

Table 1. The Parameter Values for WAL

parameter	value
min_{th}	1
max_{th}	5
$thresh_{dec}$	2
$workload_{th}$	0.5
$dissolve_{th}$	30

and the pause time was 100 seconds. The node speed was uniformly distributed between 0 and 20 meters/second (average 10 meters/second).

The simulated network consists of 100 nodes. For traffic, 40 flows were used, and the data rate of each flow was gradually increased from two to six packets/second (or 330 to 1000 Kbps) per experiment. All traffic was created with a constant-bit-rate (CBR) data source, and every packet size was 512 bytes. In the simulation, each mobile node is placed within a rectangle of 2200 meters $\times$ 600 meters. Each simulation executed for 500 simulated seconds.

3.4.2 Performance Results

To reflect various aspects of the routing protocols, the following performance metrics were used:

- Packet latency
- Packet delivery fraction
- Routing overhead

In the simulation results, each data point represents an average of 30 runs with different mobility patterns and traffic patterns. For a fair comparison, however, an identical set of mobility and traffic patterns was applied to the routing protocols in each simulation run.

As shown in Figure 31, the new scheme greatly reduces the end-to-end delay for AODV. The delay is decreased up to 32 % by applying the new scheme to AODV. On average,

AODV-WAL demonstrated a 27 % smaller delay than the base protocol. The delay performance gain gets bigger as the offered load increases. The delay performance of DSR was also improved with the new scheme. DSR-WAL showed about a 14 % smaller delay than DSR on average. DSR, however, turned out not to benefit from the new scheme as much as AODV.

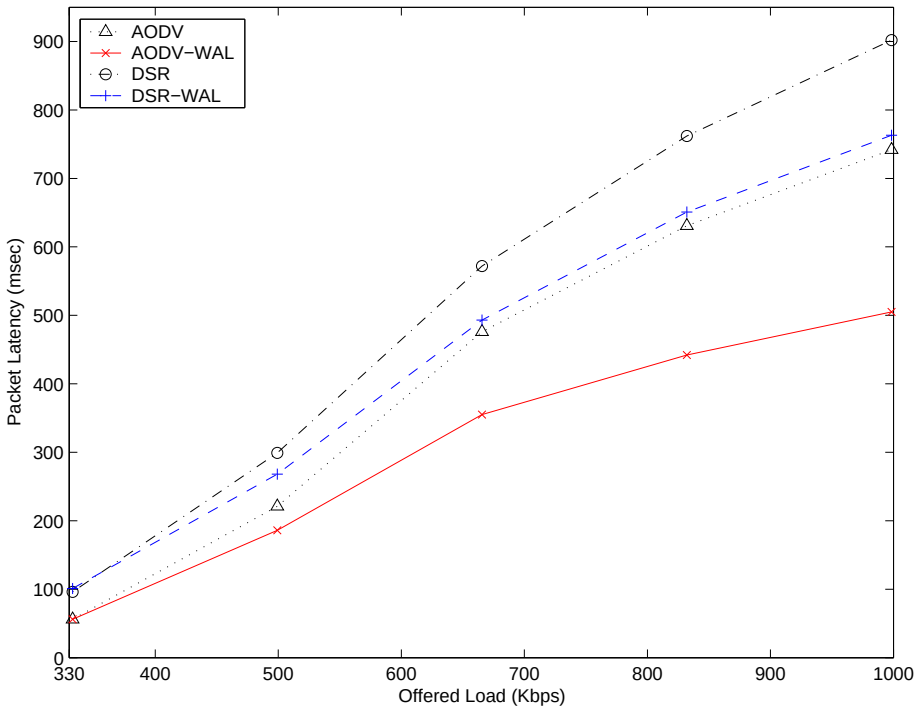


Figure 31. Average packet latency.

This different aspect of the two protocols is mainly due to the different route discovery behavior. DSR adopts a very aggressive route discovery strategy. Thus, a DSR node learns much more route information than a AODV node in a route request cycle, which leads to high hit rate in the route cache. This means most of route replies are generated from intermediate nodes rather than destination nodes in case of DSR. Hence, even though an intermediate node is already overloaded, a route reply can be generated toward the source node without forwarding the RREQ via the overloaded intermediate node in DSR. If the

RREQ had arrived at the overloaded node, it might be dropped at the node, and another path not including the overloaded node could be found, which can happen more frequently in AODV.

Figure 32 shows the average route acquisition time of each protocol. The route acquisition time is a measure of how fast a routing protocol can discover a route to a destination. It can be measured by observing how long a packet waits in the send buffer until it is actually transmitted. The figure reveals that the route acquisition time slightly increases with the new technique.

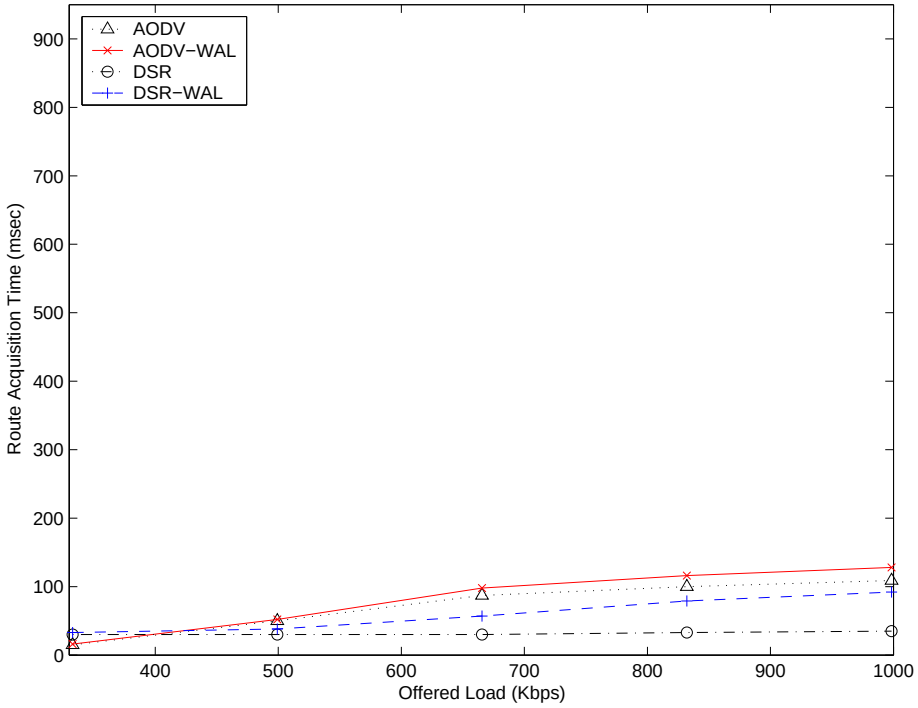


Figure 32. Average route acquisition time.

Figure 33 shows that the new scheme does not adversely affect the network throughput. Rather, the performance was somewhat improved for both protocols in terms of packet delivery fraction. If RREQs are excessively suppressed in a route discovery phase, it can

happen that a route is not found, and fairly lots of packets are dropped, degrading the network throughput. However, this was not the case for the new scheme because it adaptively suppressed the RREQs according to the local load status, and the simulation result supports this fact.

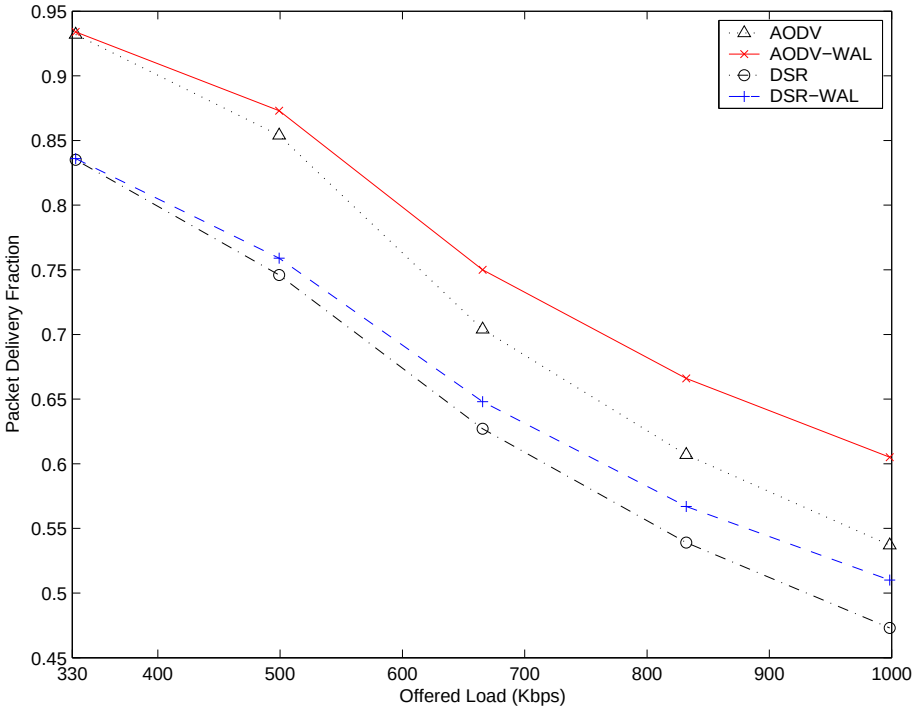


Figure 33. Average packet delivery fraction.

As can be seen in Figure 34, the routing overhead performance was improved a lot for AODV. AODV-WAL demonstrated up to a 32 % less routing overhead than the base protocol. This performance gain was obtained mainly from suppressing RREQs. In AODV, a large portion of routing messages are RREQs [55]. AODV-WAL suppresses these RREQs effectively and thus prevents the unnecessary propagation of RREQs over the network while greatly reducing overall routing overhead. Also, reduced link breakage somewhat contributed to this performance improvement. In general, a path in AODV-WAL is more stable than in the base protocol since it tries to exclude overloaded nodes from the path.

Actually, the number of link breakage events in AODV-WAL was about 16 % less than in AODV. This reduced link breakage directly translates to a reduction of routing messages because a link breakage event triggers a route error message. On average, AODV-WAL showed about a 20 % less routing overhead than AODV.

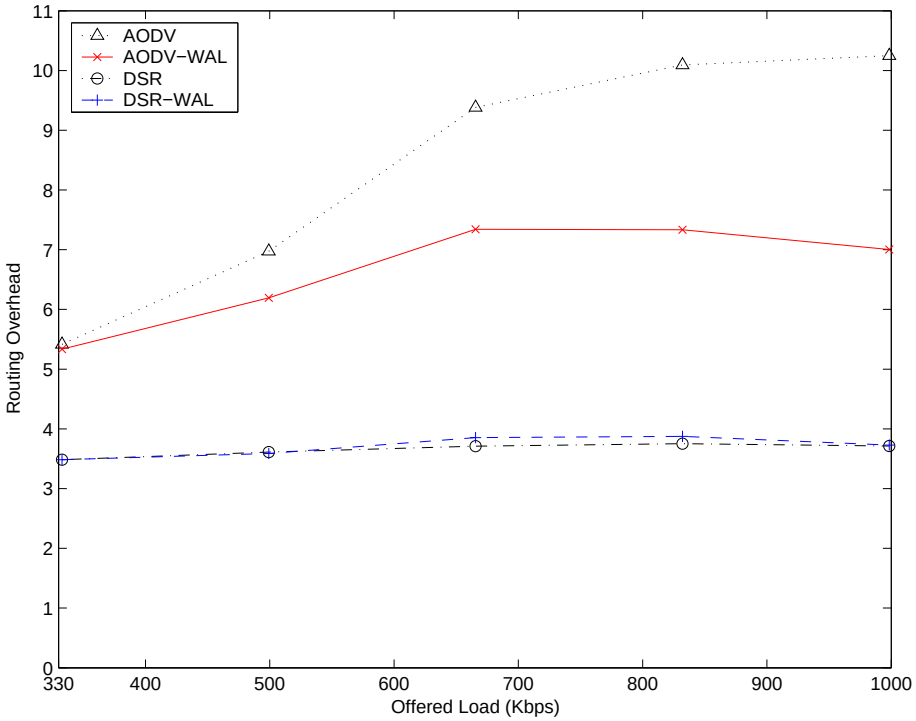


Figure 34. Average routing overhead.

On the other hand, the routing overhead of DSR-WAL is very similar to that of DSR. In DSR, RREPs occupy a large portion in the entire routing messages [55]. Moreover, as explained in the delay performance case, many RREPs are issued from the route caches without having the chance to suppress RREQs in many cases for DSR. These factors explain the similar efficiency of DSR and DSR-WAL.

Figure 35 displays the total workload distribution over the mobile nodes in the simulated network. It is to see how well the new scheme balances load among the network nodes. Each point is the total workload of a node after the simulation and represents its

load status. Many peak spots for AODV nodes can be observed in the figure. They show that those nodes were overloaded severely. Also, large fluctuations are observed, which means load is biased over the network for the base protocol. For AODV-WAL, however, the total workload distribution does not fluctuate much. It is more evenly distributed than the base protocol, which shows that the new scheme successfully balances load among the network nodes.

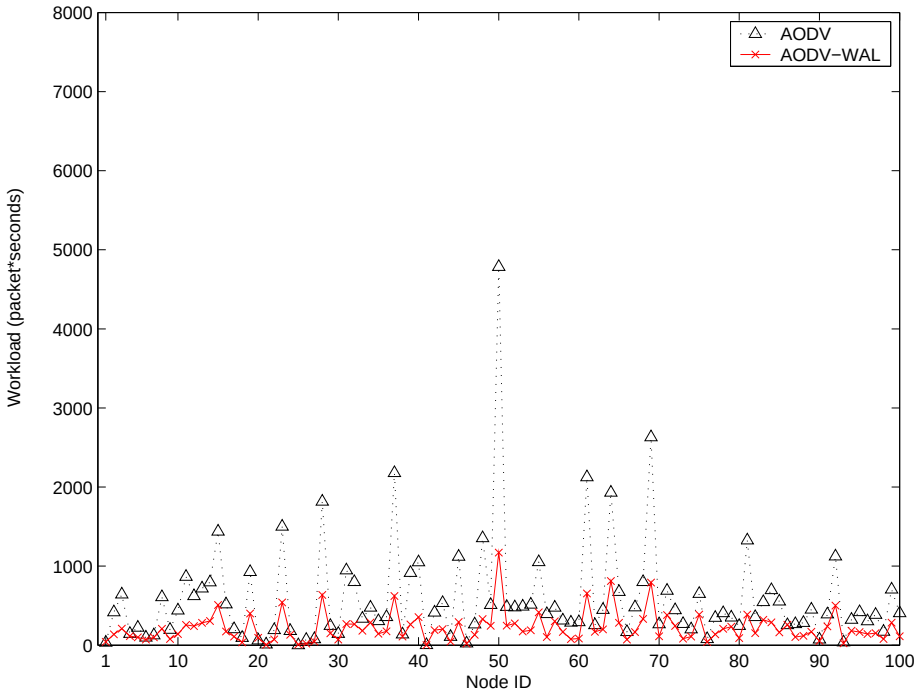


Figure 35. Workload distribution (offered Load = 832 Kbps).

3.4.3 Parameter Setting and Sensitivity

It is not easy to choose optimal values for WAL parameters as they depend on a various range of factors such as mobility, traffic pattern, etc. Roughly speaking, RREQ suppression cannot be done effectively if the threshold is bounded within a range with large values, and the network may be throttled if it is limited to very small values. However, the threshold is adjusted adaptively according to the load status of each node. Thus, even if it starts with a

Table 2. Packet Latency (sec)

Data Rate (Kbps)	330	500	670	830	1000
A	0.056	0.186	0.355	0.442	0.505
B	0.056	0.181	0.340	0.420	0.478

Table 3. Packet Delivery Fraction (%)

Data Rate (Kbps)	330	500	670	830	1000
A	93.4	87.3	75.0	66.6	60.5
B	93.3	87.7	75.8	67.6	61.5

Table 4. Normalized Routing Load

Data Rate (Kbps)	330	500	670	830	1000
A	5.336	6.194	7.342	7.334	7.002
B	5.387	6.096	7.243	7.217	6.956

different value, it will try to settle down at the point where it is considered optimal only if the range is properly set.

To see how different setting of WAL parameters can affect its performance, experiments were carried out for AODV-WAL with two different sets of parameters. The parameter set *A* represents the one used for the experiments in the previous section while the set *B* has different parameters with $max_{th} = 10$ and $thresh_{dec} = 3$.

The results are shown in Table 2 – 4. As can be observed in these tables, AODV-WAL shows very similar performance with two different sets of parameters, which implies that the WAL technique is not very sensitive to its parameter setting.

3.4.4 Enhancements by Using MAC Utilization

Table 5 – 7 show the performance of AODV with WAL only and with WAL + MAC utilization. Both enhanced versions of AODV showed very similar performance in terms of delay, PDF, and routing load. It can be seen that there is no significant performance enhancement in AODV by considering MAC utilization.

Table 8 – 10 show the performance of DSR with WAL only and with WAL + MAC utilization. Both enhanced versions of DSR showed very similar performance in terms of

Table 5. Packet Latency of AODV-WAL and AODV-WAL+MAC (sec)

Data Rate (Kbps)	330	500	670	830	1000
WAL only	0.052	0.170	0.329	0.416	0.475
WAL + MAC	0.052	0.168	0.322	0.403	0.456
Improvement	0.0%	1.0%	2.2%	3.2%	4.0%

Table 6. Packet Delivery Fraction of AODV-WAL and AODV-WAL+MAC (%)

Data Rate (Kbps)	330	500	670	830	1000
WAL only	93.6	88.2	76.4	68.1	61.9
WAL + MAC	93.6	88.3	76.8	68.6	62.5
Improvement	0.0%	0.1%	0.6%	0.7%	1.0%

Table 7. Normalized Routing Load of AODV-WAL and AODV-WAL+MAC

Data Rate (Kbps)	330	500	670	830	1000
WAL only	5.233	5.891	7.113	7.162	6.916
WAL + MAC	5.233	5.790	6.917	6.937	6.671
Improvement	0.0%	1.7%	2.8%	3.2%	3.5%

Table 8. Packet Latency of DSR-WAL and DSR-WAL+MAC (sec)

Data Rate (Kbps)	330	500	670	830	1000
WAL only	0.088	0.258	0.492	0.656	0.774
WAL + MAC	0.086	0.250	0.444	0.559	0.642
Improvement	2.1%	3.2%	9.8%	14.7%	17.1%

Table 9. Packet Delivery Fraction of DSR-WAL and DSR-WAL+MAC (%)

Data Rate (Kbps)	330	500	670	830	1000
WAL only	83.4	76.6	66.6	58.9	52.9
WAL + MAC	83.3	77.5	67.8	60.8	55.0
Improvement	-0.2%	1.2%	1.7%	3.1%	4.0%

Table 10. Normalized Routing Load of DSR-WAL and DSR-WAL+MAC

Data Rate (Kbps)	330	500	670	830	1000
WAL only	3.371	3.263	3.422	3.392	3.316
WAL + MAC	3.346	3.247	3.379	3.307	3.181
Improvement	0.7%	0.5%	1.3%	2.5%	4.1%

PDF and routing load. It can be observed in the table that considering MAC utilization further improves the delay performance of WAL. DSR-WAL+MAC demonstrated up to 17% smaller delays than DSR-WAL. On avergae, the delay performance was improved about 13% by considering link-layer information.

3.5 Conclusions

In this chapter, a novel load-balancing technique for mobile ad hoc networks was presented. The new scheme is simple but very effective in achieving load balance and congestion alleviation. It enables each node to forward RREQ messages selectively according to the load status of the node. Overloaded nodes do not allow additional communications to set up through them so that they can be excluded from the requested paths within a specific period. Each node allows additional traffic flows as long as it is not overloaded.

The new scheme utilizes interface queue occupancy and workload to control RREQ messages adaptively. In the new method, each node maintains a threshold value, which is a criterion for the decision of how to react to a RREQ message. The threshold value of a node dynamically changes according to the load status of the node based on its queue occupancy and its workload within a specific period.

It was shown via simulation that the new scheme significantly reduces packet latency as well as routing overhead especially for *AODV-type* on-demand protocols, where RREQs dominate the entire routing messages. It was also shown that the network throughput is not adversely affected but rather is improved by applying the new scheme to the base protocols. The new scheme successfully balances the network load among nodes, and it can easily be incorporated with existing on-demand routing protocols to work on top of them.

CHAPTER 4

EFFICIENT SIMULATION OF WIRELESS NETWORKS USING LAZY MAC STATE UPDATE

4.1 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* [87, 50], *GloMoSim* and its commercial counterpart *QualNet* [89], *OPNet* [96], and the Georgia Tech Network Simulator (*GTNetS*) [48]. All of these tools have detailed models of the IEEE 802.11 [46] 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 on 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* [89], *SSFNet*

[91], *SWAN* [92], *pdns* [97], and *GTNetS* [48]. For the most part, these parallel simulation tools use a conservative synchronization approach, and rely on *lookahead* [93] 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, 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 [90].

In this chapter, a different approach to improving the performance of wireless network simulations is introduced. The new approach, called *LAMP*, leads to substantial improvements in overall execution time and reduction in the size of the pending event list.

The new 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 this chapter is organized as follows. Section 4.2 gives an overview of the *LAMP* technique, with details given in Section 4.3. In Section 4.4, the performance

[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.

evaluation results of *LAMP* are presented, and the chapter is finalized in Section 4.5 with conclusions.

4.2 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 that they share an IEEE 802.11 [46] wireless medium as shown in Figure 36.

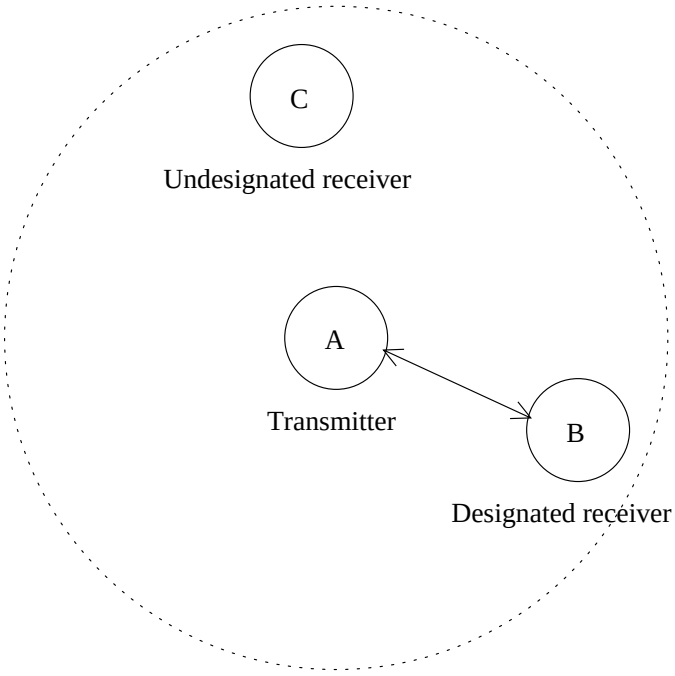


Figure 36. A simple wireless network.

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 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 communication 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

[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.

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. It is shown later that benefits of the reduced event count are significant as compared to the relatively small overhead of the list maintenance.

4.3 The Detailed Structure and Algorithm

4.3.1 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 [46] 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.

4.3.2 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)

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.

Figure 37 shows the timing diagram of the *LAMP* algorithm for the previous example of the simple wireless network described in Figure 36.

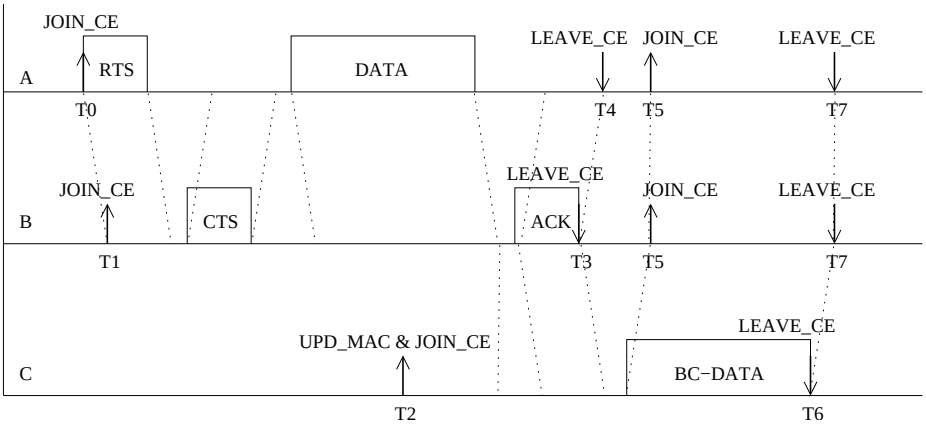


Figure 37. The timing diagram of lazy MAC state update.

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.

4.3.2.1 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.

4.3.2.2 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.

4.3.2.3 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.

4.4 Performance Evaluation

In this section, the performance evaluation results obtained from several different scenarios are presented. 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.

4.4.1 The Simulation Environment

All of the simulation experiments were done using GTNetS [48]. 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 the IEEE 802.11 [46] protocol are used for all experiments with two-ray ground reflection [98] as the propagation model.

Two sets of experiments were done to measure the performance of the *LAMP* technique. First, it was tested 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 packet reception events are scheduled 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 to test the new method for the two extreme cases. For this first set of experiments, all the nodes are put in such a way that every node can sense every other's transmission, i.e., every communication can happen in a single hop to remove effects of routing protocols. In this set of experiments, simulations were executed 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, more realistic networks were constructed with two mobility characteristics. One is with no mobility, and the other with the random waypoint mobility model [50]. 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. The node density was varied from 10 to 100, and the number of traffic flows from 5 to 20. Experiments were also performed varying the network size, 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 and 1000.

For both experiments, all traffic was created with a constant-bit-rate (CBR) data source with a rate of four packets/second, with a fixed packet size of 512 bytes. Ten simulation runs per scenario were executed and they are averaged 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.

The following were used as performance metrics:

- Memory usage
- Number of events processed
- Execution time

For the execution time, it is converted to *speedup* rather than the value is directly specified. The speedup metric captures the performance improvement as compared to traditional methods.

4.4.2 Two Extreme Cases

Figure 38 shows the simulation results from the unicast only experiment, and Figure 39 the broadcast only experiment. As can be seen in Fig 38(a) and Figure 39(a), the memory usage of *LAMP* is nearly identical to that of the traditional method. On 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 the experiments, and can be explained in the same way.

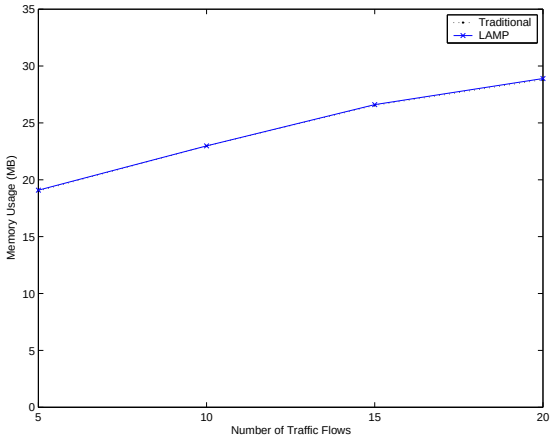
As expected, in Figure 38(b), it can be seen that *LAMP* significantly reduces the number of events processed during the simulation. The event count was reduced by a factor of 15 for 5 traffic flows, a factor of 6 for 20 traffic flows, and a factor of 7 on average. It can be also observed, for both methods, 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 Figure 39(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.

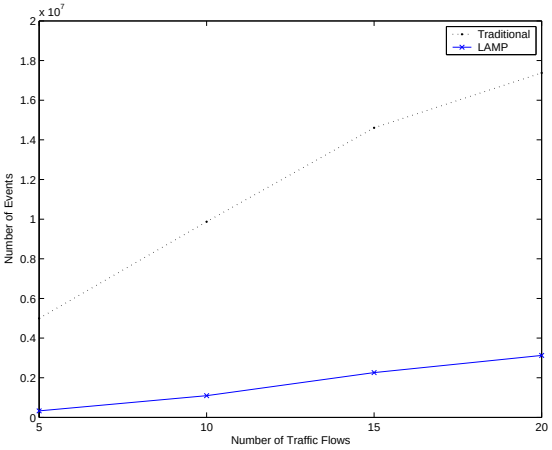
For the traditional method, it can be observed that the processed events in the unicast only experiment were much more than those in the broadcast only experiment. This is due to the increased number of packets for unicast communications using additional frames such as RTS, CTS, and ACK.

Figure 38(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 average, *LAMP* demonstrated speedup of about 3 in these experiments.

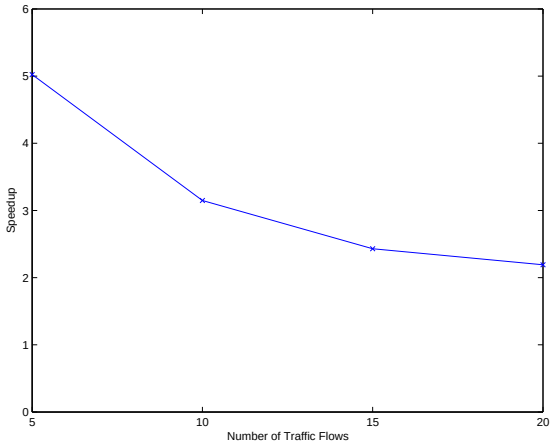
In Figure 39(c), on the other hand, it can be seen that speedup by *LAMP* is less than one, which means that the simulation with *LAMP* was actually slightly slower than the one with the traditional method. Even though the difference is small (about 10 % slower on average), one may wonder what made this 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



(a) Memory usage

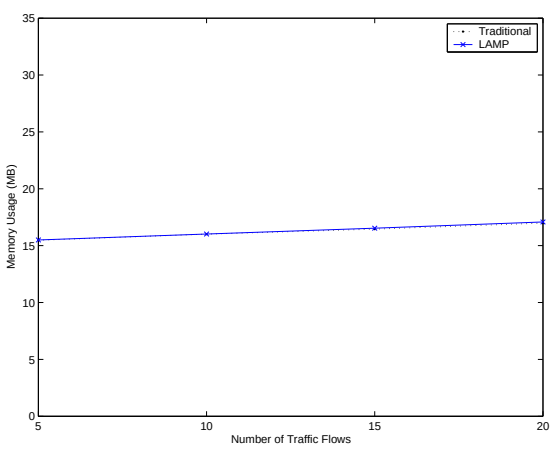


(b) Number of events

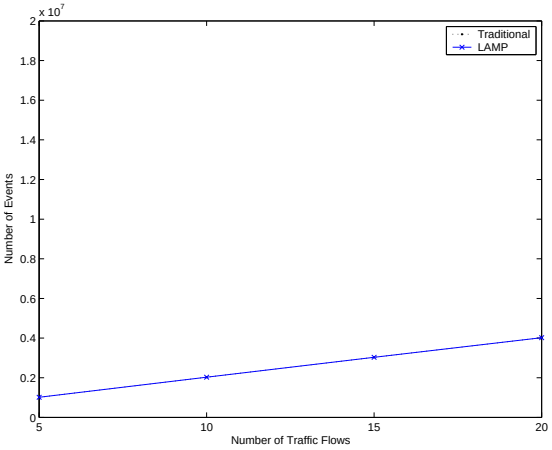


(c) Speedup

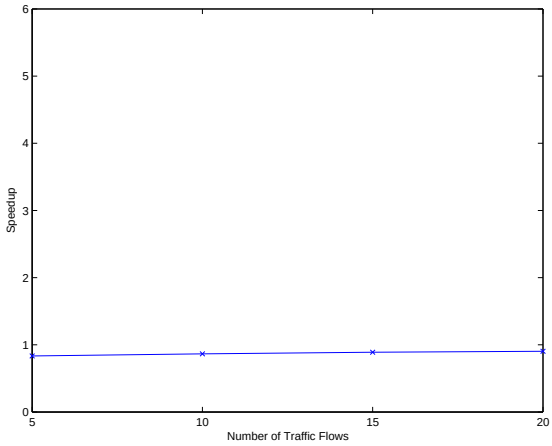
Figure 38. Unicast only (100 nodes with varying traffic flows).



(a) Memory usage



(b) Number of events



(c) Speedup

Figure 39. Broadcast only (100 nodes with varying traffic flows).

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.

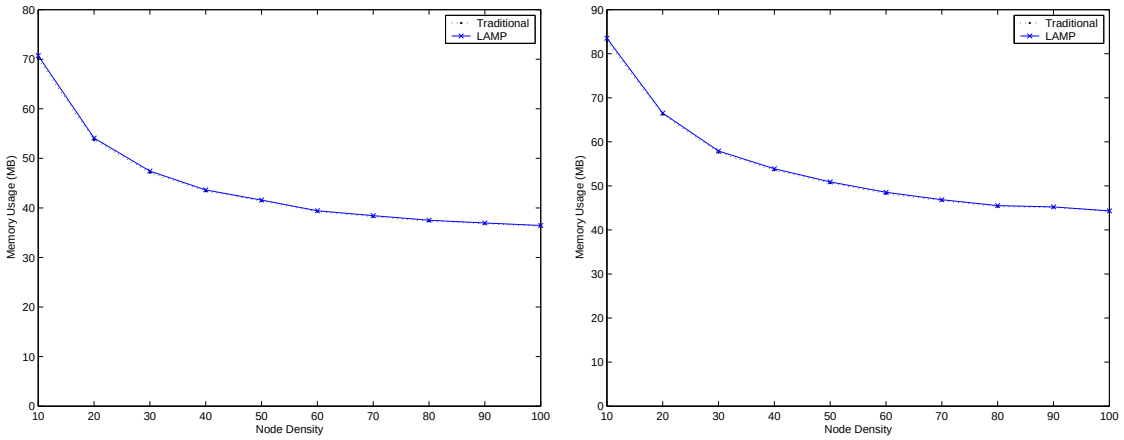
4.4.3 The 500-node and 1000-node Experiments

Figure 40 shows memory usage of the simulation results from the 500 and 1000-node experiments with no mobility. This was obtained using various node densities with a fixed number of traffic flows.

Figure 41 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.

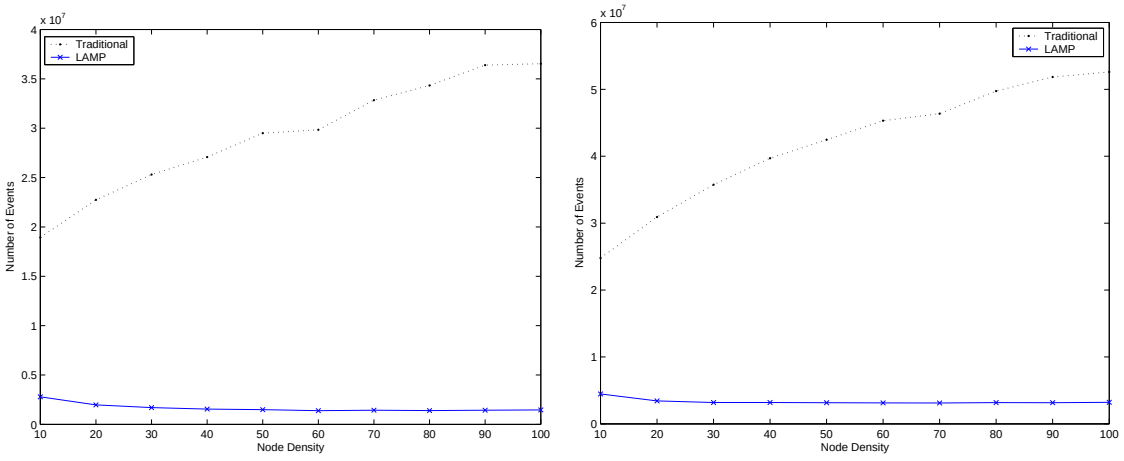
As can be seen in Figure 42, a speedup of about up to eight was achieved by *LAMP*. On average, the simulation with *LAMP* was about six times faster than the traditional method. Speedup increases with node density because time for processing events with the traditional method increases with node density, while time for processing events with *LAMP* is largely unaffected by node density.

Figure 43 shows memory usage of the simulation results from the 500 and 1000-node experiments with no mobility. This was obtained varying the number of traffic flows with fixed node density.



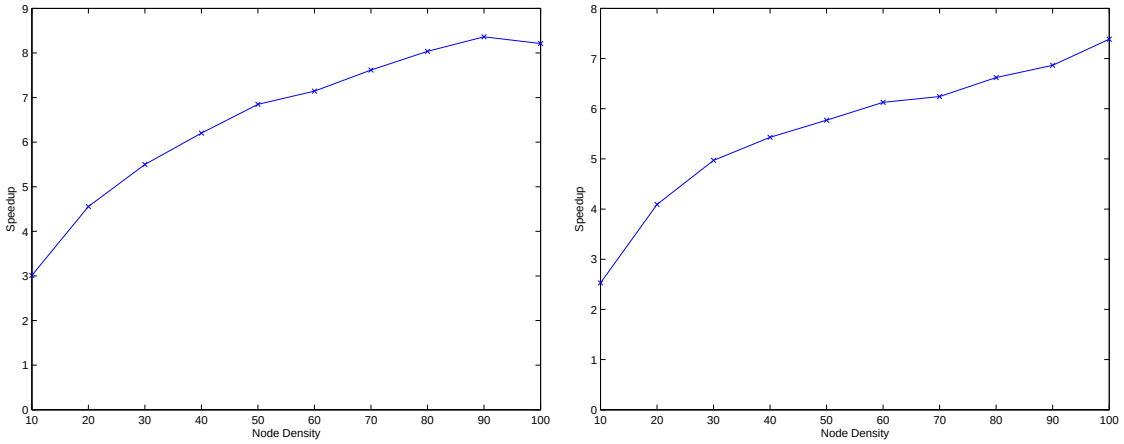
(a) 500 nodes (b) 1000 nodes

Figure 40. Memory usage (with no mobility varying node densities with 10 traffic flows).



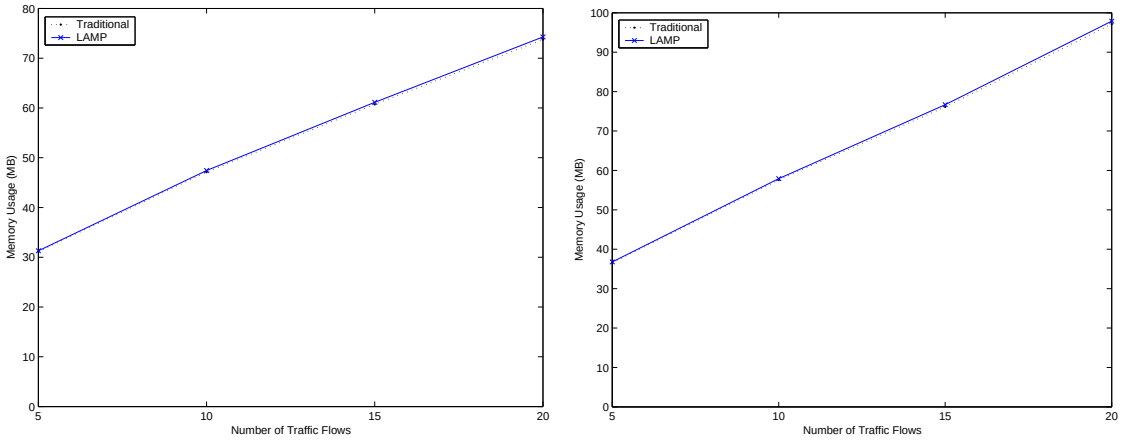
(a) 500 nodes (b) 1000 nodes

Figure 41. Number of events (with no mobility varying node densities with 10 traffic flows).



(a) 500 nodes (b) 1000 nodes

Figure 42. Speedup (with no mobility varying node densities with 10 traffic flows).



(a) 500 nodes (b) 1000 nodes

Figure 43. Memory usage (with no mobility varying traffic flows with node density of 30).

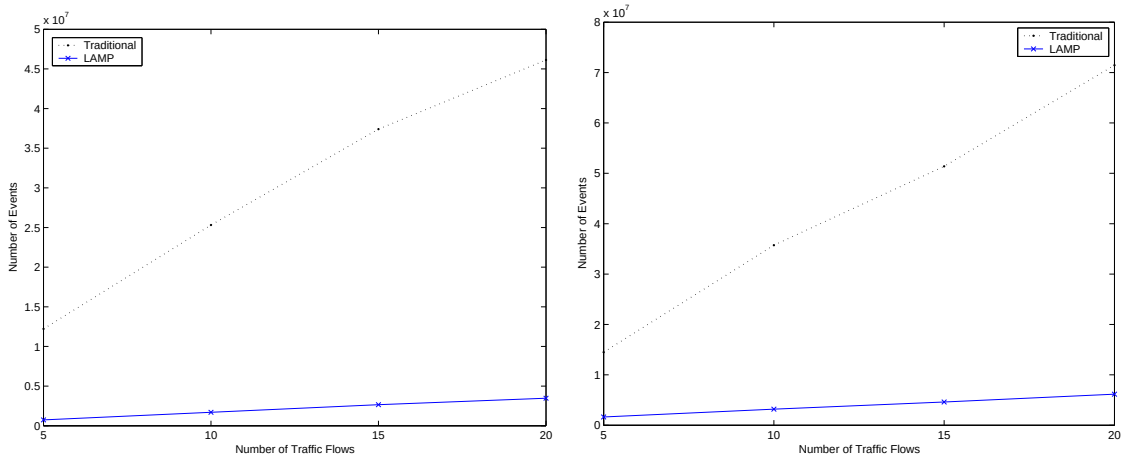
In Figure 44, it can be seen, for both methods, that the number of events increases with the number of traffic flows. But the slope of the line graph for the two method is quite different. The event count for the traditional method quickly increases with traffic flows, which is not the case for *LAMP*.

Figure 45 shows speedup obtained from the 500 and 1000-node experiments with no mobility varying the number of traffic flows. It can be observed 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 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 these experiments, speedup of up to seven (for 500 nodes with five traffic flows) was achieved. On average, speedup of about four was achieved by *LAMP*.

Figure 46 shows memory usage of the simulation results from the 500 and 1000-node experiments with mobility. This was obtained using various node densities with a fixed number of traffic flows.

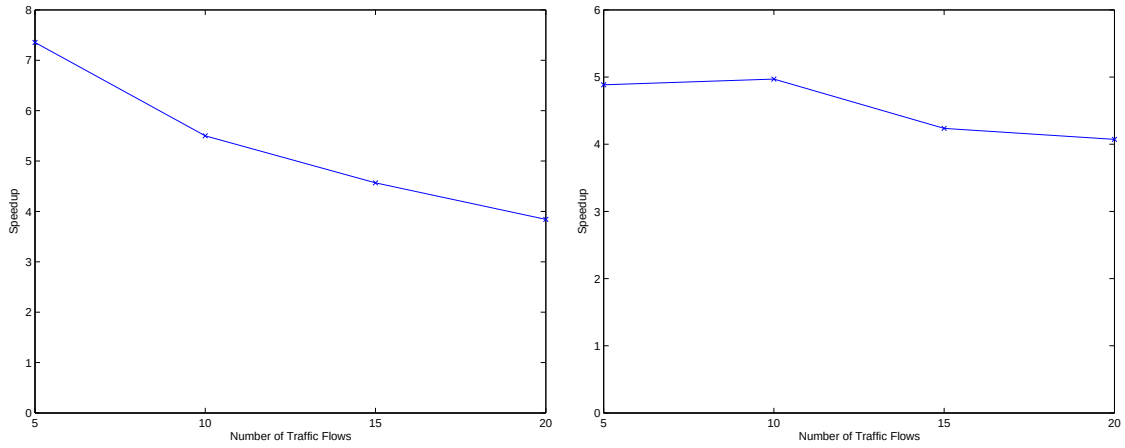
The graphs in Figure 47 look similar to those in the preceding case with no mobility. The number of events processed in the simulation with the traditional method grows with node density while the number for *LAMP* remains relatively stable. This again confirms that *LAMP* is scalable with respect to 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.

Figure 48 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



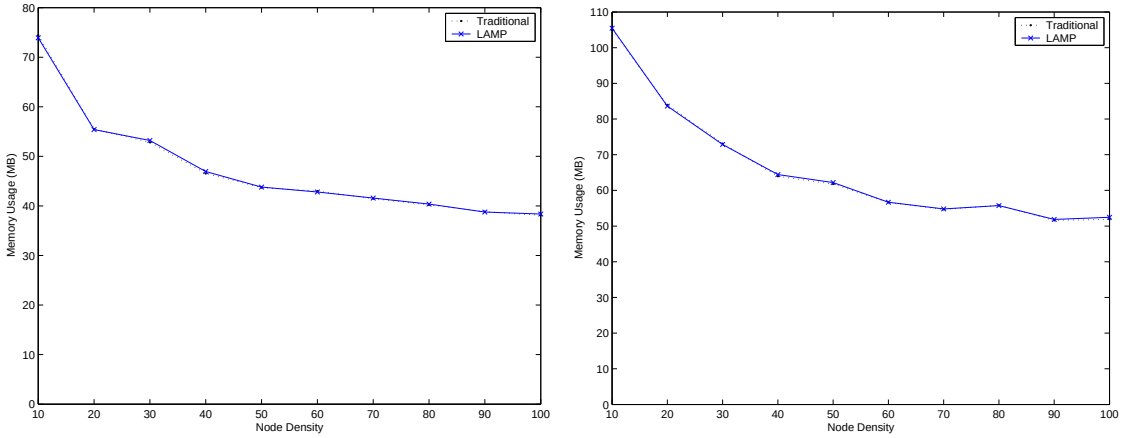
(a) 500 nodes (b) 1000 nodes

Figure 44. Number of events (with no mobility varying traffic flows with node density of 30).



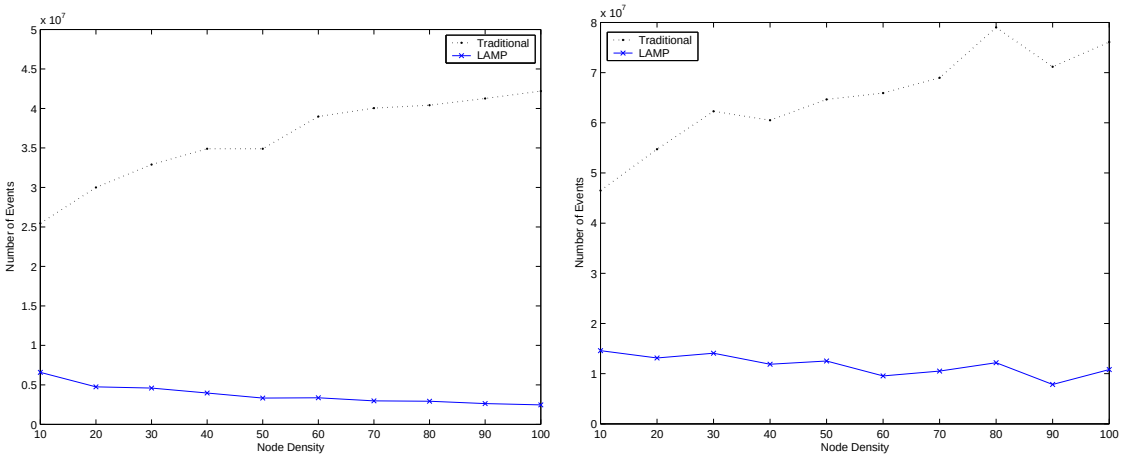
(a) 500 nodes (b) 1000 nodes

Figure 45. Speedup (with no mobility varying traffic flows with node density of 30).



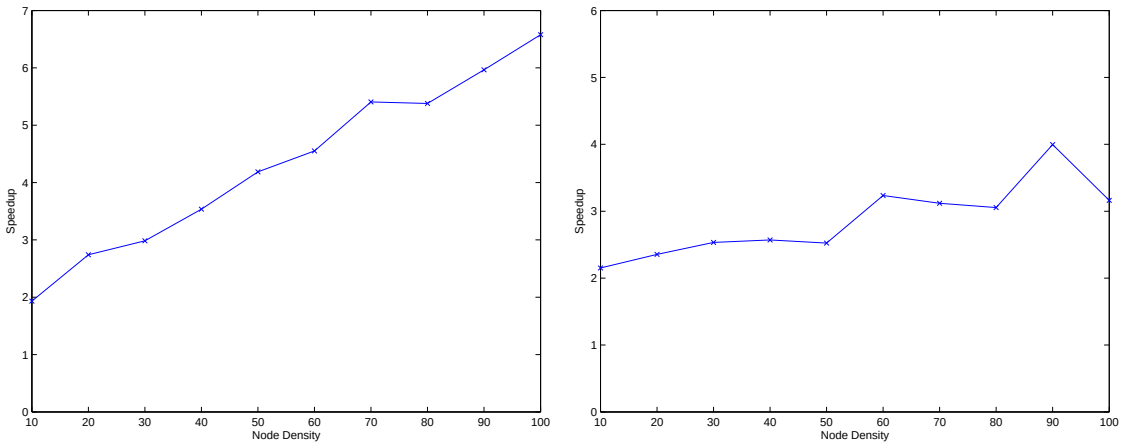
(a) 500 nodes (b) 1000 nodes

Figure 46. Memory usage (with mobility varying node densities with 10 traffic flows).



(a) 500 nodes (b) 1000 nodes

Figure 47. Number of events (with mobility varying node densities with 10 traffic flows).



(a) 500 nodes (b) 1000 nodes

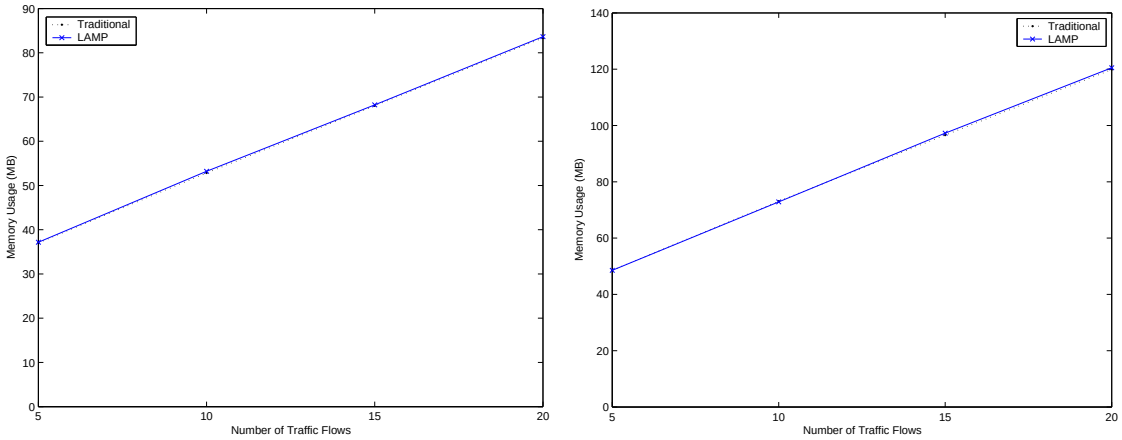
Figure 48. Speedup (with mobility varying node densities with 10 traffic flows).

LAMP was nearly seven times faster for the 500-node experiment and four times faster for the 1000-node experiment than the one with the traditional method, and about three times faster on 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.

Figure 49 shows memory usage of the simulation results from the 500 and 1000-node experiments with mobility. This was obtained varying traffic flows with fixed node density.

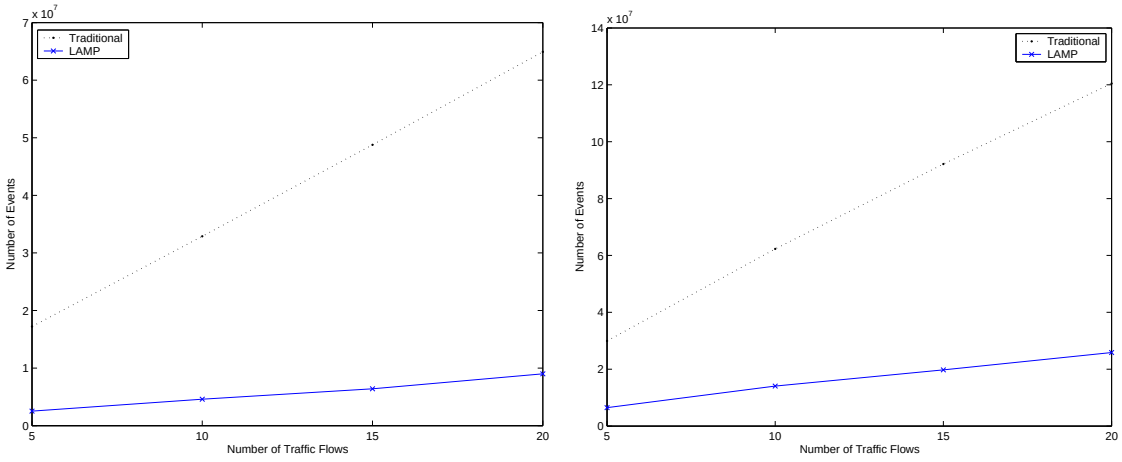
Figure 50 shows the number of events processed during simulation from the 500 and 1000-node experiments with mobility. This was obtained varying traffic flows with fixed node density. The same tendency can be observed in Figure 50. For both methods, the number of events processed during the simulation increases with the number of traffic flows, as discussed previously.

Figure 51 shows speedup obtained from the 500 and 1000-node experiments with mobility varying the number of traffic flows. The maximum speedup was about 3.2, and the average speedup of 2.7 was achieved from these experiments. The reason for decreasing speedup with traffic flows was discussed above.



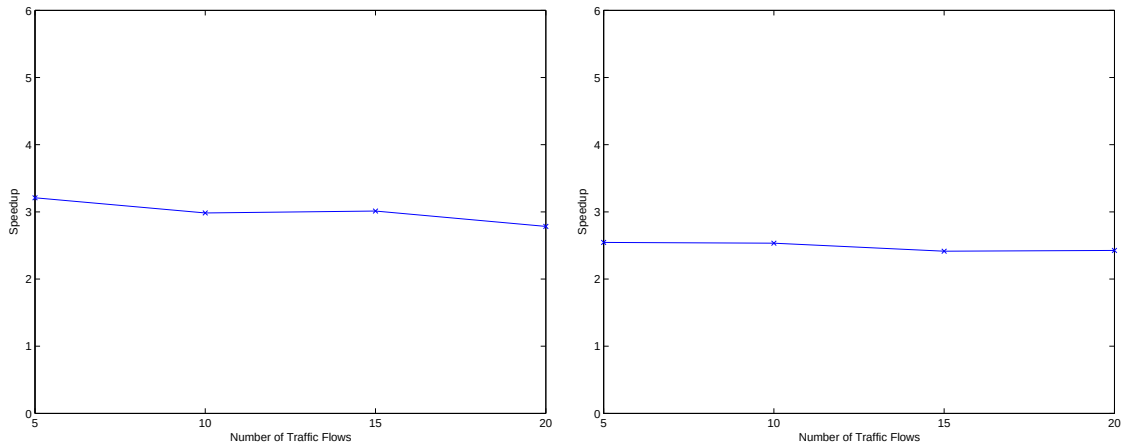
(a) 500 nodes (b) 1000 nodes

Figure 49. Memory usage (with mobility varying traffic flows with node density of 30).



(a) 500 nodes (b) 1000 nodes

Figure 50. Number of events (with mobility varying traffic flows with node density of 30).



(a) 500 nodes (b) 1000 nodes

Figure 51. Speedup (with mobility varying traffic flows with node density of 30).

4.4.4 The Accuracy of LAMP

Tables 11, 12, and 13 show packet delivery fraction, packet latency, and normalized routing load respectively, which were obtained from the 500-node experiment varying traffic flows with no mobility for each simulation method.

As can be seen in these tables, 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.

4.5 Conclusions

A novel simulation technique for mobile wireless networks was presented, which is efficient in terms of the execution time and the event list size, and can be achieved with a sequential simulation. The new 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

Table 11. Packet Delivery Fraction in LAMP (%)

Traffic Flows	5	10	15	20
Traditional	99.997	99.994	99.989	99.985
LAMP	99.997	99.994	99.989	99.985

Table 12. Packet Latency in LAMP (ms)

Traffic Flows	5	10	15	20
Traditional	18.663	21.901	24.342	23.582
LAMP	18.663	21.901	24.342	23.582

Table 13. Normalized Routing Load in LAMP

Traffic Flows	5	10	15	20
Traditional	0.1902	0.2242	0.2318	0.2313
LAMP	0.1902	0.2242	0.2318	0.2313

of MAC protocols, but it was implemented for the IEEE 802.11 [46] MAC sub-layer for the 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.

CHAPTER 5

PARALLEL EXTENSION OF LAMP FOR SCALABLE SIMULATION OF WIRELESS NETWORKS

5.1 Introduction

Even though *LAMP* improves the efficiency and scalability of wireless network simulation, other issues such as a large amount of computing resources due to very large-scale network simulation still need to be addressed as *LAMP* was originally designed for sequential simulation of wireless networks.

The parallel simulation of wireless networks, however, suffers from the small lookahead problem and consequently degraded speed performance. This problem results from the inherent small propagation delay of wireless networks because a lookahead value in a parallel wireless network simulation is mainly obtained from a propagation delay of a signal going through a wireless communication medium.

To cope with this speed performance problem of parallel wireless simulation, the *LAMP* technique is extended to a parallel version for scalable simulation of wireless networks.

A parallel and distributed wireless simulation environment was constructed using GTNetS [48]. This simulation environment is based on the conservative synchronization method for time management. In addition, the enhanced version of GTNetS using *parallel LAMP* was also implemented, and the performance of this enhanced simulator was compared to the base parallel simulator through extensive simulation experiments.

The remainder of this chapter is organized as follows. In Section 5.2, the detailed data structures and algorithms for the parallel extension of *LAMP* are explained. In Section 5.3, the parallel simulation environment and results are presented and discussed. Section 5.4 concludes this chapter.

5.2 Parallel Extension of LAMP

In a parallel and distributed simulation, a message exchanged between logical processes (LPs) basically contains time information and a packet. In a wireless environment, a packet needs to be scheduled at each receiver if the received signal strength is strong enough to be detected even though the packet is not destined for it. Thus, a system needs to compute the path loss whenever it receives a message containing a packet to determine whether or not to schedule a packet receipt event at each receiver within the system. For this purpose, additional information on the transmitter such as location, transmitter power, and radio range is necessary. Hence, this additional information is also included in a message.

The time appearing in a message cannot be greater than the current simulation time of a system at the very moment it has received the message because conservative time management is used. Thus, the packet in a message is scheduled at the time specified in the message by a system as soon as it receives the message.

In scheduling a packet receipt event, there can be computational redundancy regarding calculation of path loss and selection of receivers. For example, let us suppose that a system keeps receiving messages from a node in another system. Then, the system needs repeated computation for path loss and receiver choice even though each computation leads to the same result if the nodes involved are static.

To remove this computational redundancy, a *radio map* is introduced. Each entry in the radio map consists of a *transmitter name* and a *receiver set*. A node id supplied by a system cannot be used as the transmitter name since the name must be globally unique. Thus, the IP address of a node is used as this name. Each entry in the receiver set contains a *receiver id* and *signal strength*. The receiver id identifies the receiver node uniquely in a system. Therefore, a node id can be used as a receiver id. The signal strength field explains how strong the received signal at the receiver is.

When a system receives a message from a node within another system, it first scans its radio map to see if the map has already an entry for the sender node. If so, it has

only to schedule the packet at each receiver in the receiver set of the entry, and the saved signal strength value is used without calculating the corresponding path loss. Otherwise, the system performs path loss calculation for each node in it and examines if the node can receive the packet. A receiver set is built by adding every receiver detecting the packet with the corresponding signal strength value. A new entry is then inserted into the radio map with the corresponding transmitter name and the constructed receiver set.

In network simulations including a wireless one, a sender of a packet needs to resolve the destination's IP address into the corresponding MAC address to transmit a packet at the link layer. In real world networks, this job is done by the ARP [38]. In a sequential simulation (a single system), this can be done in two ways. One is by modeling the ARP and using it, and the other is by utilizing internal information of the system. Normally, a simulation system maintains all the information about nodes, links, interfaces, IP addresses, and MAC addresses within it. Thus, a system can have a map that combines the IP address of each interface with the corresponding MAC address or provide a mechanism that does the same job. This method, however, cannot be used in distributed systems because there is no central administration.

Figure 52 describes this situation. Let us assume that there are two simulation systems *A* and *B* and that two nodes *S* and *D* belong to system *A* and *B* respectively. Let us suppose that *S* has a data packet destined for *D*. For link-layer transmission of the packet, *S* has to know the MAC address of the corresponding interface in *D*. However, system *A* does not have necessary information on system *B*.

One approach to solving this problem is to use the ARP as in the sequential simulation. An ARP query packet is delivered in the form of a message to another system, and the corresponding ARP reply can be transmitted back to the system in which the originator of the ARP query exists. If ARP modeling is wanted in simulations, this method is a good solution. Otherwise, this can incur additional overheads because of increased communication among multiple systems.

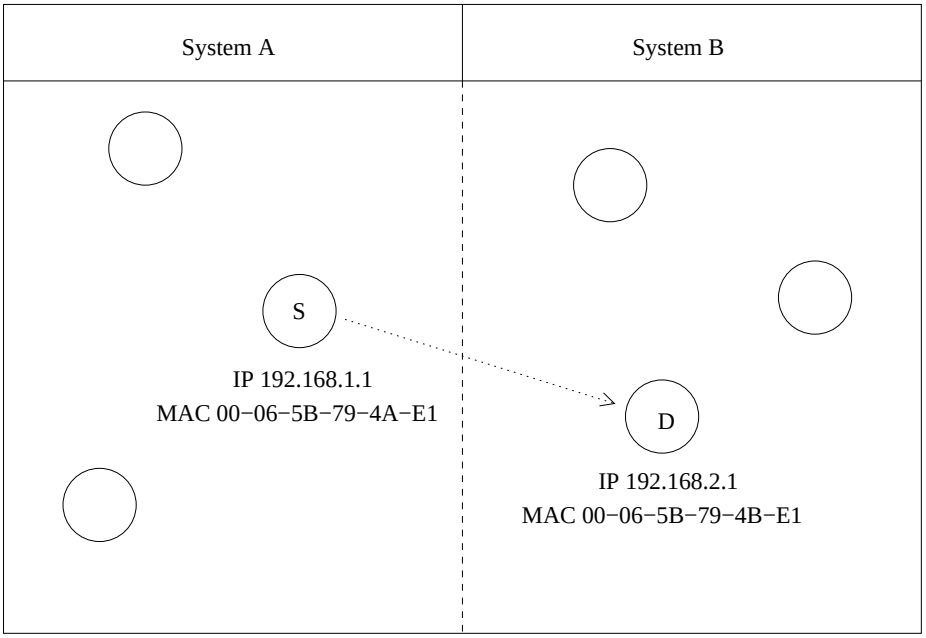


Figure 52. A distributed system

Another approach is to force each interface to use its IP address as its MAC address. Then, a sender can simply use the destination's IP address as the MAC address at the link layer. With this approach, there is no need for address resolution. Therefore, this does not incur any overhead regarding address resolution.

5.3 Performance Evaluation

This section presents the performance evaluation results from extensive parallel simulation experiments. GTNetS with the *parallel LAMP* technique is compared to GTNetS based on the traditional parallel simulation method through simulations. The simulation environment is described in detail, and then the simulation results are presented and discussed.

5.3.1 The Simulation Environment

Two different versions of GTNetS [48] were built for experiments. One is GTNetS with a parallel simulation capability, which is the base simulator for comparison. The other is the enhanced version of GTNetS with *parallel LAMP*.

The IEEE 802.11 [46] MAC protocol was used for all experiments with two-ray ground reflection [98] as the propagation model. For every experiment, a static wireless network was used. The primary variables were node density and traffic volume. The node density was varied from 20 to 100, and the numbers of traffic flows used were 20, 60, and 100. The experiments were also performed varying the network size, traffic patterns, and the number of LPs. The ratio of the radio transmission range to the network radius was properly set to force multi-hop routing for all experiments. The numbers of nodes in the simulated network were 2500 and 5000.

For every experiment, all traffic was created with a constant-bit-rate (CBR) data source with a rate of four packets/second, with a fixed packet size of 512 bytes. Five simulation runs per scenario were executed, and they were averaged to represent each data point in the simulation results. Each simulation executed for 500 simulated seconds.

A rectangular 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 $D = \frac{N}{X \cdot Y} \cdot \pi r^2$, where D is the node density, N the network size, X the terrain width, Y the terrain length, and r the transmission range. For each experiment, the node density was pre-determined, and then the computed terrain width and length were used to generate the node spacing with radio transmission range of 250 meters.

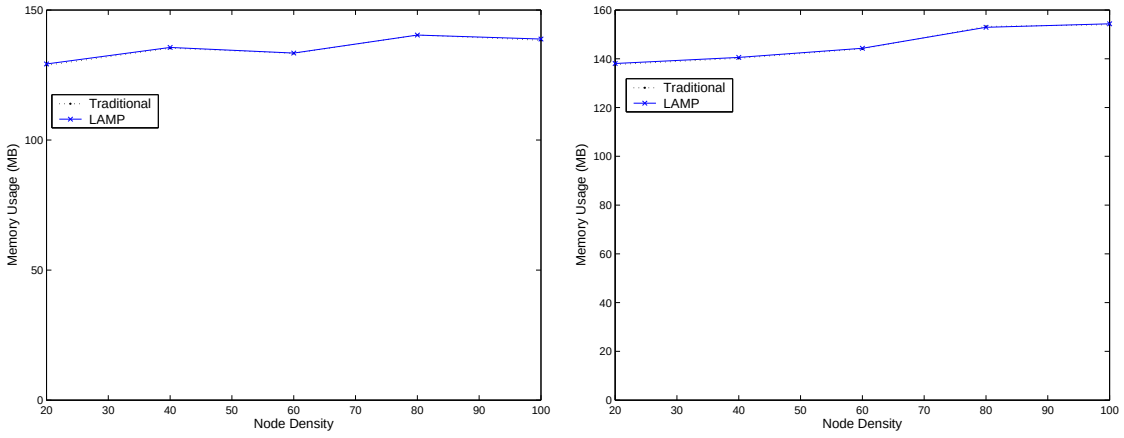
The performance metrics are the following:

- Memory usage
- Number of processed events
- Speedup

The speedup metric captures the speed performance improvement of GTNetS with *parallel LAMP* against the one with the traditional method.

5.3.2 Performance Results

Figure 53 shows memory usage of the simulation results from the 2500 and 5000-node experiments with two LPs varying node densities with a fixed number of traffic flows.



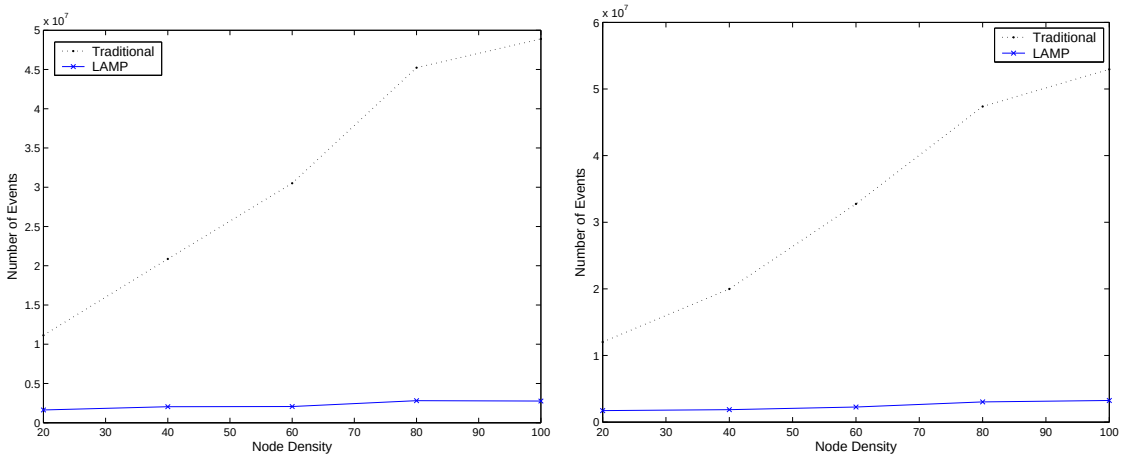
(a) 2500 nodes (b) 5000 nodes

Figure 53. Memory usage (with two LPs varying node densities with 20 traffic flows).

As can be seen in Figure 53, the memory usage of GTNetS with *parallel LAMP* is nearly identical to that of the traditional method. On average, GTNetS with *parallel LAMP* consumed only about 240 KB more than the traditional method, which is less than 0.2 % with respect to the total memory footprint. As explained in the previous chapter, this is because the only memory overhead incurred by GTNetS with *parallel LAMP* is the use of the wireless link event list and the communicating entity list whose sizes are relatively small.

Figure 54 shows that GTNetS with *parallel LAMP* processed substantially reduced number of events during simulation. Please note that the number of events processed by GTNetS with *parallel LAMP* does not change much with varying node densities while the event count for the traditional method almost linearly increases. For the traditional method, this phenomenon is natural 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 *parallel LAMP* is scalable with respect to the node density of a network as in the *LAMP* case, and is responsible for the nearly flat graph with varying node densities.

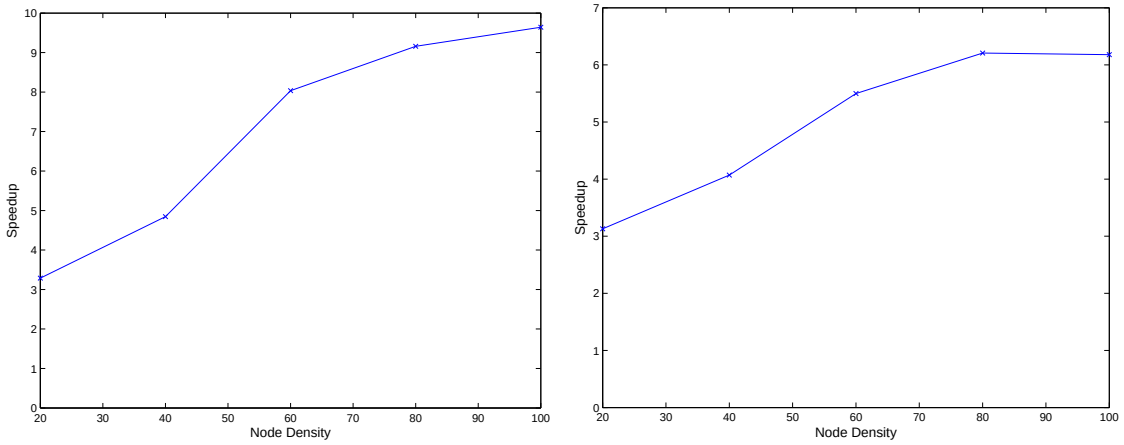


(a) 2500 nodes (b) 5000 nodes

Figure 54. Number of events (with two LPs varying node densities with 20 traffic flows).

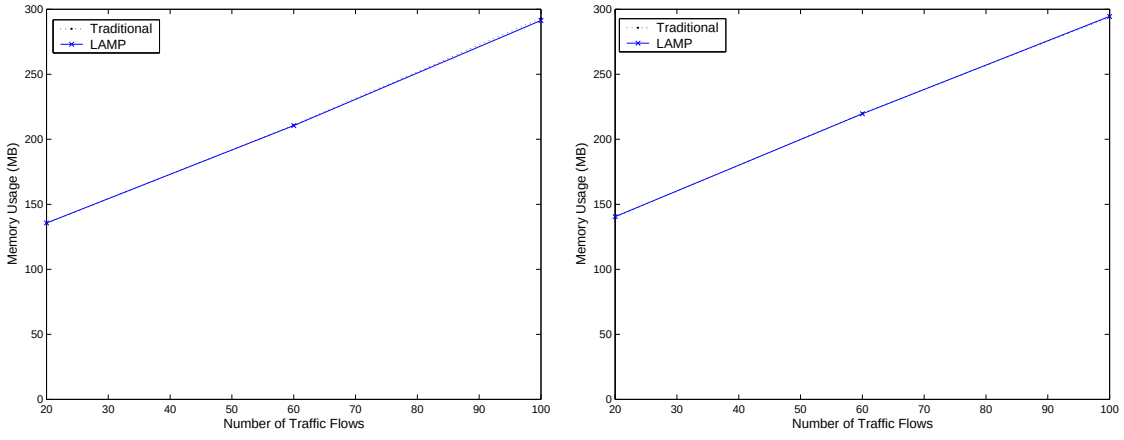
As can be seen in Figure 55, a speedup of about up to 10 was achieved by GTNetS with *parallel LAMP*. On average, the simulation by GTNetS with *parallel LAMP* was about 7 times faster than the traditional method for 2500 nodes, and about 5 times faster for 5000 nodes. Speedup increases with node density because time for processing events with the traditional method increases with node density, while time taken by GTNetS with *parallel LAMP* to process events is largely unaffected by node density.

Figure 56 shows memory usage of the simulation results from the 2500 and 5000-node experiments with two LPs varying the number of traffic flows with fixed node density. As can be seen in the figure, the memory consumption by the simulation increases with the



(a) 2500 nodes (b) 5000 nodes

Figure 55. Speedup (with two LPs varying node densities with 20 traffic flows).



(a) 2500 nodes (b) 5000 nodes

Figure 56. Memory usage (with two LPs varying traffic flows with node density of 40).

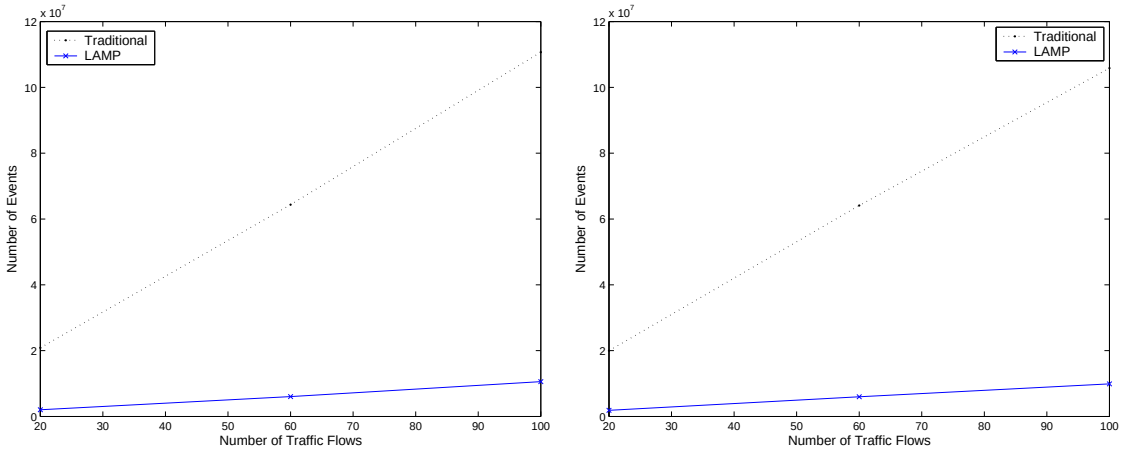
number of traffic flows.

In Figure 57, it can be observed, for both methods, that the events increase with the number of traffic flows almost linearly. 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 GTNetS with *parallel LAMP* that produced much more gentle slopes.

Figure 58 shows speedup obtained from the 2500 and 5000-node experiments respectively varying the number of traffic flows. It can be observed 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 decreased speed performance 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 message traffic. As discussed in the previous chapter, the increased number of broadcast packets can adversely affect the performance of *LAMP* relative to the traditional method. For these experiments, speedup of about up to 5 (for 2500 nodes with 20 traffic flows) was achieved. On average, speedup of about 4 was achieved by GTNetS with *parallel LAMP*.

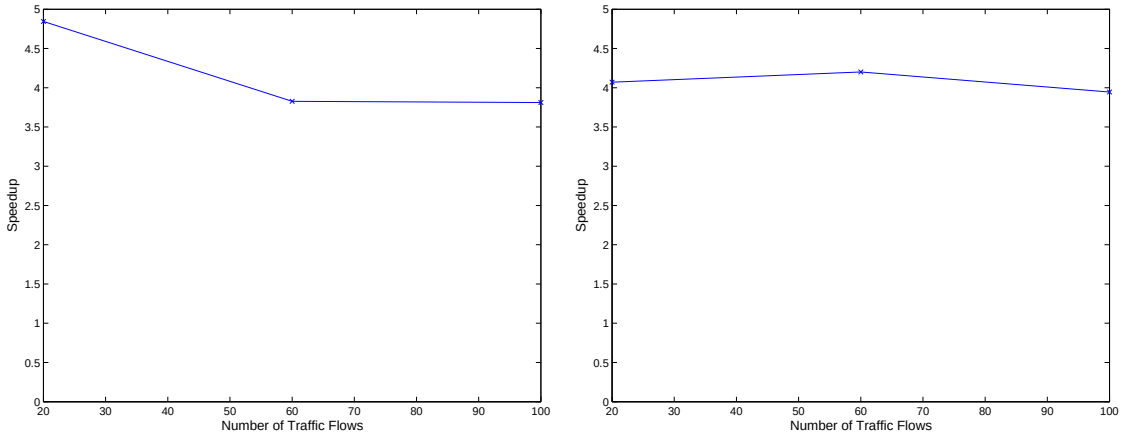
Figure 59 shows memory usage of the simulation results from the 2500 and 5000-node experiments with four LPs varying node densities with a fixed number of traffic flows. As shown in this figure, both simulators consumed almost the same amount of memory. The reason that GTNetS with *parallel LAMP* consumed a little bit more memory than the traditional method was explained previously.

Figure 60 shows the number of events executed during the simulations with four LPs. As observed in the previous experiments with two LPs, GTNetS with *parallel LAMP* produced a greatly reduced number of events compared to the traditional method. The reason that the number of events from GTNetS with *parallel LAMP* stays stable with respect to the node density was also explained in the previous experiments.



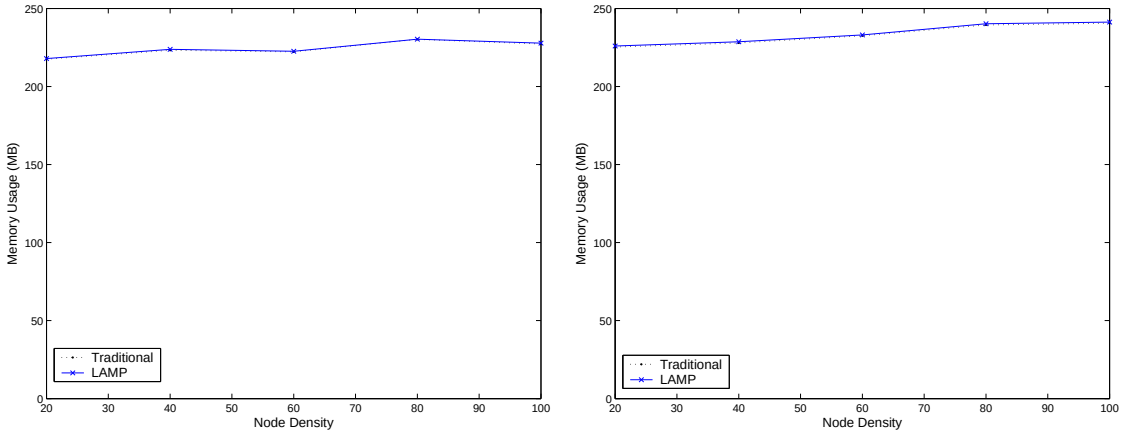
(a) 2500 nodes (b) 5000 nodes

Figure 57. Number of events (with two LPs varying traffic flows with node density of 40).



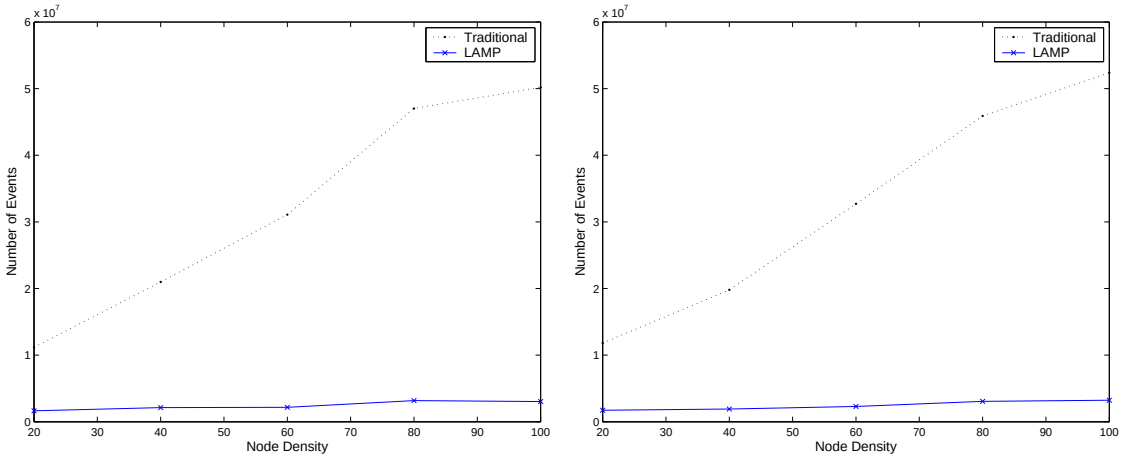
(a) 2500 nodes (b) 5000 nodes

Figure 58. Speedup (with two LPs varying traffic flows with node density of 40).



(a) 2500 nodes (b) 5000 nodes

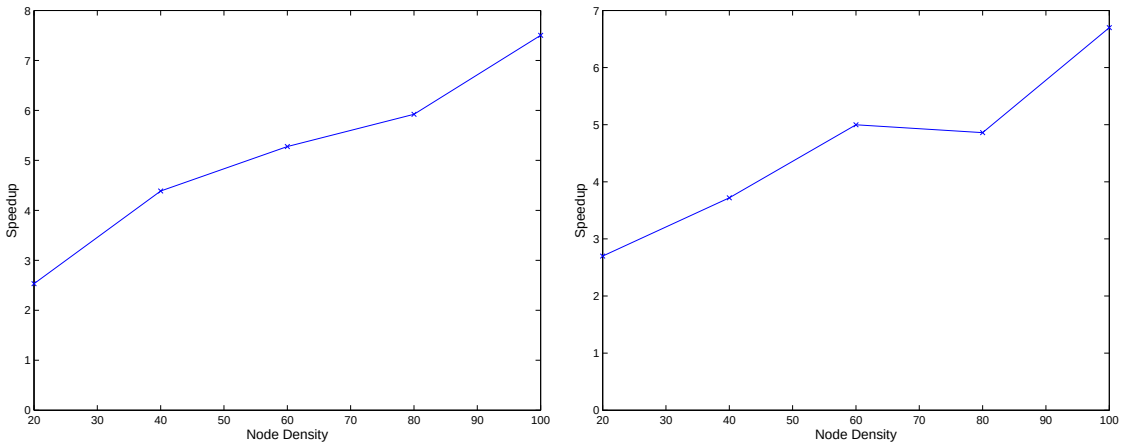
Figure 59. Memory usage (with four LPs varying node densities with 20 traffic flows).



(a) 2500 nodes (b) 5000 nodes

Figure 60. Number of events (with four LPs varying node densities with 20 traffic flows).

Figure 61 shows the speed performance of GTNetS with *parallel LAMP* in terms of speedup. The higher speedup is achieved as the node density increases regardless of the network size. On average, the simulation with *parallel LAMP* was about 5.4 times faster than the traditional method for 2500 nodes and was about 4.8 times faster for 5000 nodes.

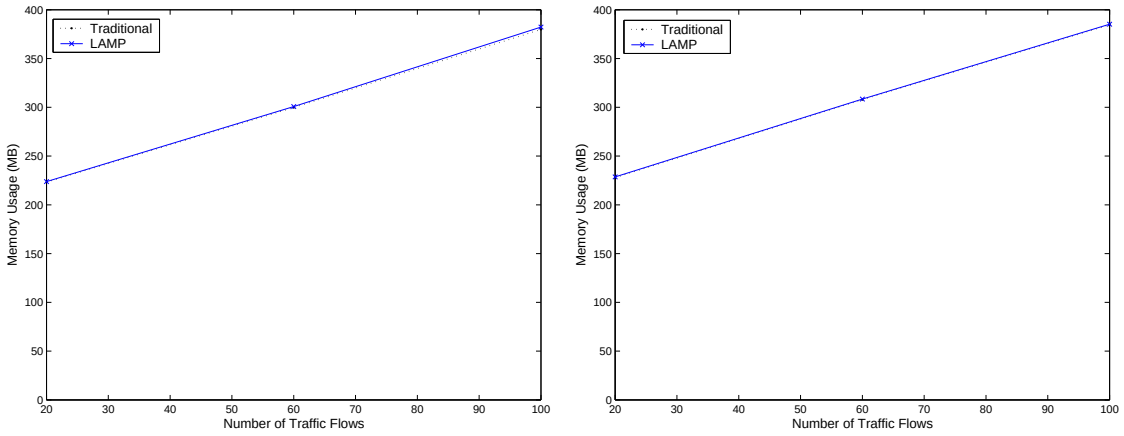


(a) 2500 nodes (b) 5000 nodes

Figure 61. Speedup (with four LPs varying node densities with 20 traffic flows).

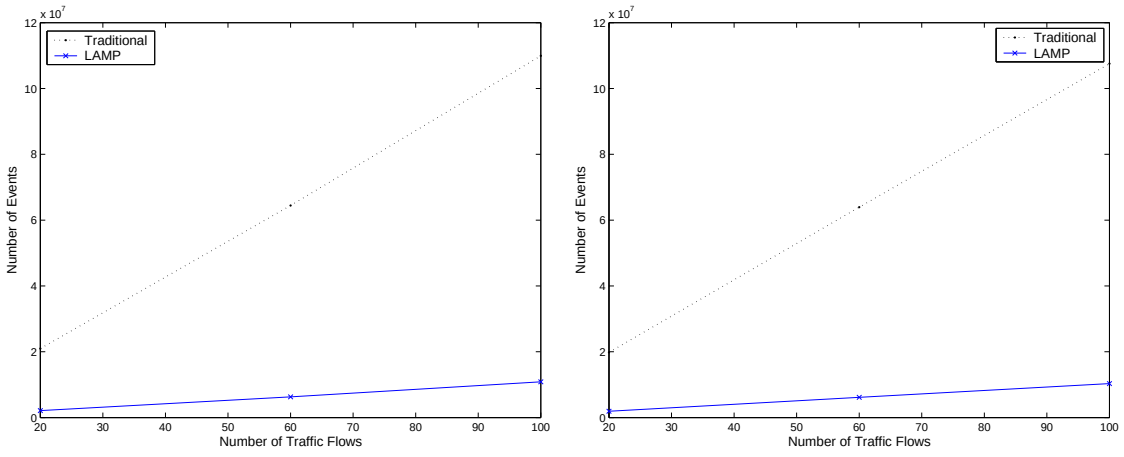
Figure 62 shows memory usage of the simulation results from the 2500 and 5000-node experiments with four LPs varying the number of traffic flows with fixed node density. As can be seen in this figure, the memory consumption by the simulation increases with the number of traffic flows, and GTNetS with *parallel LAMP* consumed slightly more memory than the traditional method.

Figure 63 shows the number of events processed during the simulation with four LPs and a varying number of traffic flows. As experienced in the previous simulations with two LPs, the number of events grows almost linearly with the number of traffic flows for both simulators. But GTNetS with em parallel LAMP produces a much more gentle slope than the traditional method.



(a) 2500 nodes (b) 5000 nodes

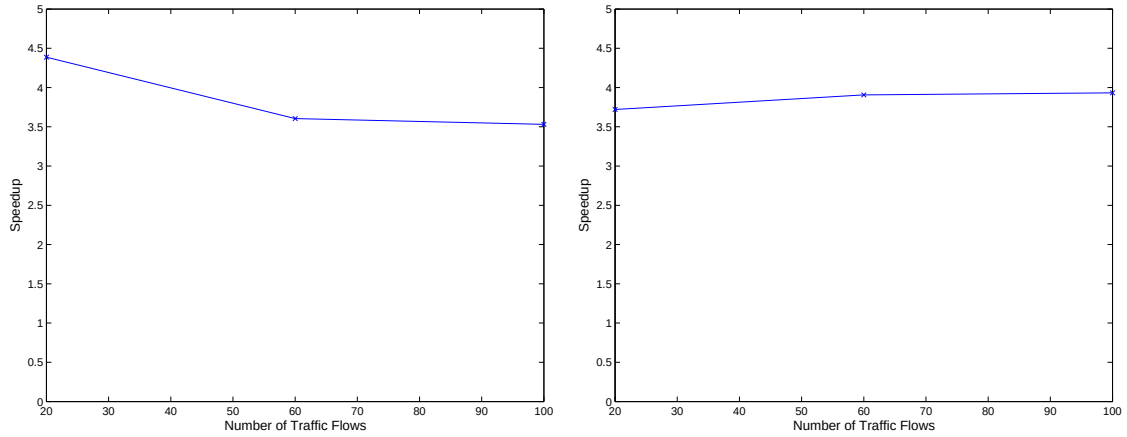
Figure 62. Memory usage (with four LPs varying traffic flows with node density of 40).



(a) 2500 nodes (b) 5000 nodes

Figure 63. Number of events (with four LPs varying traffic flows with node density of 40).

Figure 64 shows speedup obtained from the 2500 and 5000-node experiments respectively with four LPs and a varying number of traffic flows. As can be seen in this figure, the speedup performance of GTNetS with *parallel LAMP* was not affected very much by the number of traffic flows. For 2500 nodes, speedup slightly decreases with the number of traffic flows. For 5000 nodes, speedup does not change much for the different number of traffic flows. On average, speedup of about 3.6 was achieved for 2500 nodes, and about 3.9 for 5000 nodes.



(a) 2500 nodes (b) 5000 nodes

Figure 64. Speedup (with four LPs varying traffic flows with node density of 40).

5.4 Conclusions

The parallel simulation of wireless networks suffers from the small lookahead problem and consequently degraded speed performance due to the inherent small propagation delay of wireless networks. To cope with this speed performance problem of parallel wireless simulation, the *LAMP* technique was extended to a parallel version for scalable simulation of wireless networks.

A parallel and distributed wireless simulation environment was constructed using GTNetS [48]. This simulation environment is based on the conservative synchronization method for time management. The enhanced version of GTNetS using *parallel LAMP* was also implemented.

For assessing the performance of this enhanced simulator against the base parallel simulator, extensive simulation experiments were carried out. The simulation results reveal that the new simulator tremendously reduces the number of simulation events processed during simulation and consequently improves the speed performance significantly.

CHAPTER 6

CONCLUSIONS

In this dissertation, routing protocols, load-balancing protocols, and efficient evaluation techniques for multi-hop mobile wireless networks are explored. The main contributions of the researches presented in the dissertation can be summarized as follows:

- Design of the Dynamic NIx-Vector Routing (DNVR) protocol for MANETs
- Development of a novel distributed load-balancing technique for MANETs
- Development of the Lazy MAC State Update (LAMP) technique for efficient evaluation of mobile wireless networks
- Extension of the LAMP technique to a parallel version for scalable simulation of wireless networks

In Chapter 2, a new protocol for multi-hop routing in MANETs, which is called *DNVR*, is presented. DNVR is based on the *NIx-Vector* concept for efficient routing originally designed for wired networks. DNVR acquires a loop-free route and maintains it on a need basis as do other on-demand routing protocols. However, DNVR has several new features compared to other existing reactive routing protocols, which lead to more stable routes and better scalability. DNVR effectively validates the stored routes as well as efficiently senses the up-to-date network topology during a route discovery phase by sending a unicast probe packet. To accommodate networks with a high degree of mobility, the routing states are invalidated in a timely manner. DNVR adopts a conservative route discovery strategy by suppressing route requests in some cases, and thus only a few routes are maintained per destination. Moreover, it attains bandwidth efficiency by using a *Neighbor Index* and Medium Access Control (MAC) addresses for routing purposes, which obviates the need for address resolution.

It was shown via simulation that DNVR scales well to a large network with varying traffic load under diverse mobility scenarios. DNVR was compared to the well known *Dynamic Source Routing (DSR)* protocol, which is believed to be one of the most efficient on-demand routing protocols. Simulation results reveal that DNVR is as efficient as DSR while at the same time providing higher packet delivery ratios and smaller delays than DSR in most cases.

In Chapter 3, a simple but very effective method to achieve load balance and congestion alleviation is presented. Currently, ad hoc routing protocols lack load-balancing capabilities. Therefore, they often fail to provide good performance, especially in the presence of a large volume of traffic. The new scheme is motivated by the observation that ad hoc on-demand routing protocols flood route request (RREQ) messages to acquire routes, and only nodes that respond to those messages have a potential to serve as intermediate forwarding nodes. If a node ignores RREQ messages within a specific period, it can completely be excluded from the additional communications that might have occurred for that period otherwise. Thus, a node can decide not to serve a traffic flow by dropping the RREQ for that flow. In the new scheme, RREQ messages are forwarded selectively according to the load status of each node so that overloaded nodes can be excluded from the requested paths. Each node begins to allow additional traffic flows again whenever its overloaded status is dissolved. The new scheme utilizes interface queue occupancy and workload to control RREQ messages adaptively.

The enhanced versions of protocols with this scheme are compared to the base protocols. Simulation results reveal that the new scheme greatly reduces packet latency as well as routing overhead without adversely affecting the network throughput, and it successfully balances the network load among nodes.

In Chapter 4, a new method for efficient simulation of wireless networks is proposed. 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 on 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, attention is turned to more traditional sequential simulation methods, and a method is sought to reduce the overhead incurred in the MAC state update propagation between wireless nodes. A novel method called *LAMP* is proposed to substantially reduce this overhead.

Using GTNetS [48], the efficiency of the *LAMP* approach is compared to the more traditional approach, and it is shown that a significant performance improvement can be achieved with no loss of accuracy.

In Chapter 5, extension of *LAMP* to a parallel version for scalable simulation of wireless networks is discussed. A base parallel and distributed simulation environment was constructed, and it was enhanced with *LAMP*. Extensive simulations were carried out for performance evaluation of the new technique, and simulation results reveal that the new simulation environment tremendously reduces the simulation events processed during simulation and consequently shortens the execution time significantly.

REFERENCES

[1] S. Corson and J. Macker, "Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations," RFC 2501, IETF, January 1999.

[2] H.-Y. Hsieh and R. Sivakumar, "On using the ad-hoc network model in cellular packet data networks," *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, June 2002.

[3] Mobile Ad-hoc Networks Working Group, Internet Engineering Task Force, http://www.ietf.org/html.charters/manet-charter.html.

[4] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM'94)*, August 1994.

[5] S. Murthy and J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *ACM/Kluwer Mobile Networks and Applications*, vol. 1, no. 2, pp. 183-197, October 1996.

[6] M. Joa-Ng and I.-T. Lu, "A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1415-1425, August 1999.

[7] J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," *IEEE Int'l Conference on Network Protocols (ICNP'99)*, November 1999.

[8] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, IETF, October 2003.

[9] R. Ogier, F. Templin, and M. Lewis, "Topology dissemination based on reverse-path forwarding (TBRPF)," RFC 3684, IETF, February 2004.

[10] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing, edited by T. Imielinski and H. Korth, Ch. 5, pp. 153-181*, Kluwer Academic Publishers, 1996.

[11] C. Perkins and E. Royer, "Ad hoc on-demand distance vector routing," *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, February 1999.

[12] V. Park and S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," *IEEE Conference on Computer Communications (INFOCOM'97)*, April 1997.

[13] R. Dube, C. Rais, K.-Y. Wang, and S. Tripathi, "Signal stability based adaptive routing (SSA) for ad hoc mobile networks," *IEEE Personal Communications*, vol. 4, no. 1, pp. 36-45, February 1997.

[14] Y.-C. Hu and D. Johnson, "Implicit source routes for on-demand ad hoc network routing," *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, October 2001.

[15] S. Singh, M. Woo, and C. Raghavendra, "Power-aware routing in mobile ad hoc networks," *ACM Int'l Conference on Mobile Computing and Networking (MobiCom'98)*, October 1998.

[16] M. Corson and S. Batsell, "A reservation-based multicast (RBM) routing protocol for mobile networks: overview of initial route construction," *IEEE Conference on Computer Communications (INFOCOM'95)*, April 1995.

[17] J. Garcia-Luna-Aceves and E. Madruga, "A multicast routing protocol for ad-hoc networks," *IEEE Conference on Computer Communications (INFOCOM'99)*, March 1999.

[18] E. Royer and C. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," *ACM Int'l Conference on Mobile Computing and Networking (MobiCom'99)*, August 1999.

[19] T. Ozaki, J. Kim, and T. Suda, "Bandwidth-efficient multicast routing for multihop, ad-hoc wireless networks," *IEEE Conference on Computer Communications (INFOCOM 2001)*, April 2001.

[20] L. Ji and M. Corson, "Differential destination multicast - a MANET multicast routing protocol for small groups," *IEEE Conference on Computer Communications (INFOCOM 2001)*, April 2001.

[21] J. Jetcheva and D. Johnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, October 2001.

[22] S. Lee, W. Su, and M. Gerla, "On-demand multicast routing protocol in multihop wireless mobile networks," *ACM/Kluwer Mobile Networks and Applications*, vol. 7, no. 6, pp. 441-453, 2002.

[23] C. Gui and P. Mohapatra, "Efficient overlay multicast for mobile ad hoc networks," *IEEE Wireless Communications and Networking Conference (WCNC 2003)*, March 2003.

[24] P. Floreen, P. Kaski, J. Kohonen, and P. Orponen, "Lifetime maximization for multicasting in energy-constrained wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 117-126, January 2005.

[25] I. Maric and R. Yates, "Cooperative multicast for maximum network lifetime," *IEEE Journal on Selected Areas in Communications,* vol. 23, no. 1, pp. 127-135, January 2005.

[26] J. Sobrinho and A. Krishnakumar, "Quality-of-service in ad hoc carrier sense multiple access wireless networks," *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 8, pp. 1353-1368, August 1999.

[27] C. Lin and J.-S. Liu, "QoS routing in ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 8, pp. 1426-1438, August 1999.

[28] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," *IEEE Journal on Selected Areas in Communications,* vol. 17, no. 8, pp. 1488-1505, August 1999.

[29] C. Lin, "On-demand QoS routing in multihop mobile networks," *IEEE Conference on Computer Communications (INFOCOM 2001),* April 2001.

[30] S. Chakrabarti and A. Mishra, "QoS issues in ad hoc wireless networks," *IEEE Communications Magazine,* vol. 39, no. 2, pp. 142-148, February 2001.

[31] C. Zhu and M. Corson, "QoS routing for mobile ad hoc networks," *IEEE Conference on Computer Communications (INFOCOM 2002),* June 2002.

[32] Y.-K. Ho and R.-S. Liu, "A novel routing protocol for supporting QoS for ad hoc mobile wireless networks," *Wireless Personal Communications,* vol. 22, no. 3, pp. 359-385, 2002.

[33] Q. Xue and A. Ganz, "Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks," *Journal of Parallel and Distributed Computing,* vol. 63, no. 2, pp. 154-165, 2003.

[34] J. Stine and G. Veciana, "A paradigm for quality-of-service in wireless ad hoc networks using synchronous signaling and node states," *IEEE Journal on Selected Areas in Communications,* vol. 22, no. 7, pp. 1301-1321, September 2004.

[35] L. Chen and W. Heinzelman, "QoS-aware routing based on bandwidth estimation for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications,* vol. 23, no. 3, pp. 561-572, March 2005.

[36] K.-C. Wang and P. Ramanathan, "QoS assurances through class selection and proportional differentiation in wireless networks," *IEEE Journal on Selected Areas in Communications,* vol. 23, no. 3, pp. 573-584, March 2005.

[37] G. Riley, M. Ammar, and E. Zegura, "Efficient routing using NIx-Vectors," *IEEE Workshop on High Performance Switching and Routing (HPSR 2001),* May 2001.

[38] D. Plummer, "An ethernet address resolution protocol: or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware," RFC 826, IETF, November 1982.

[39] D. Johnson, D. Maltz, and Y.-C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)," Internet Draft, IETF, July 2004, work in progress.

[40] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Routing protocols for mobile ad-hoc networks - a comparative performance analysis," *ACM Int'l Conference on Mobile Computing and Networking (MobiCom'99),* August 1999.

[41] C. Carter, S. Yi, and R. Kravets, "ARP considered harmful: manycast transactions in ad hoc networks," *IEEE Wireless Communications and Networking Conference (WCNC 2003),* March 2003.

[42] L. Perrone, Y. Yuan, and D. Nicol, "Modeling and simulation best practices for wireless ad hoc networks," *Winter Simulation Conference (WSC 2003),* December 2003.

[43] C. Santivanez, B. McDonald, I. Stavrakakis, and R. Ramanathan, "On the scalability of ad hoc routing protocols," *IEEE Conference on Computer Communications (INFOCOM 2002),* June 2002.

[44] S. Lee, E. Royer, and C. Perkins, "Scalability study of the ad hoc on-demand distance vector routing protocol," *ACM/Wiley Int'l Journal of Network Management,* vol. 13, no. 2, pp. 97-114, March 2003.

[45] X. Hong, K. Xu, and M. Gerla, "Scalable routing protocols for mobile ad hoc networks," *IEEE Network Magazine,* vol. 16, no. 4, pp. 11-21, July - August 2002.

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

[47] IEEE Standards Committee 754, "IEEE standard for binary floating-point arithmetic," ANSI/IEEE standard 754-1985, ANSI/IEEE, New York, 1985, reprinted in SIGPLAN notices, 22(2):9-25, 1987.

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

[49] http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/.

[50] 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.

[51] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *ACM Int'l Conference on Mobile Computing and Networking (MobiCom'99),* August 1999.

[52] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002),* June 2002.

[53] R. Sivakumar, P. Sinha, and V. Bharghavan, "Braving the broadcast storm: infrastructural support for ad hoc routing," *Int'l Journal of Computer and Telecommunications Networking,* vol. 41, no. 6, pp. 687-706, 2003.

[54] S. Wang, "Reducing the energy consumption caused by flooding messages in mobile ad hoc networks," *Int'l Journal of Computer and Telecommunications Networking,* vol. 42, no. 1, pp. 101-118, 2003.

[55] S. Das, C. Perkins, and E. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Conference on Computer Communications (INFOCOM 2000),* March 2000.

[56] S. Lee and M. Gerla, "Dynamic load-aware routing in ad hoc networks," *IEEE Int'l Conference on Communications (ICC 2001),* June 2001.

[57] M. Pearlman, Z. Haas, P. Scholander, and S. Tabrizi, "Alternate path routing in mobile ad hoc networks," *IEEE Military Communications Conference (MILCOM 2000),* October 2000.

[58] P. Pham and S. Perreau, "Multi-path routing protocol with load balancing policy in mobile ad hoc network," *IFIP Int'l Conference on Mobile and Wireless Communications Networks (MWCN 2002),* September 2002.

[59] P. Pham and S. Perreau, "Performance analysis of reactive shortest path and multipath routing mechanism with load balance," *IEEE Conference on Computer Communications (INFOCOM 2003),* March 2003.

[60] M. Pearlman, Z. Haas, P. Scholander, and S. Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc networks," *ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2000),* August 2000.

[61] A. Nasipuri, R. Castaneda, and S. Das, "Performance of multipath routing for on-demand protocols in ad hoc networks," *ACM/Kluwer Mobile Networks and Applications,* vol. 6, no. 4, pp. 339-349, August 2001.

[62] Z. Ye, S. Krishnamurthy, and S. Tripathi, "A framework for reliable routing in mobile ad hoc networks," *IEEE Conference on Computer Communications (INFOCOM 2003),* March 2003.

[63] Y. Ganjali and A. Keshavarzian, "Load balancing in ad hoc networks: single-path routing vs. multi-path routing," *IEEE Conference on Computer Communications (INFOCOM 2004),* March 2004.

[64] O. Kelly, J. Lai, N. Mandayam, A. Ogielski, J. Panchal, and R. Yates, "Scalable parallel simulations of wireless networks with WiPPET: modeling of radio propagation, mobility and protocols," *ACM/Kluwer Mobile Networks and Applications,* vol. 5, no. 3, pp. 199-208, 2000.

[65] D. Nicol, "Scalability of network simulators revisited," *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2003)*, February 2003.

[66] D. Nicol, J. Liu, M. Liljenstam, and G. Yan, "Simulation of large-scale networks using SSF," *Winter Simulation Conference (WSC 2003)*, December 2003.

[67] R. Fujimoto, "Parallel and distributed simulation systems," *Winter Simulation Conference (WSC 2001)*, December 2001.

[68] G. Riley, "Large-scale network simulations with GTNetS," *Winter Simulation Conference (WSC 2003)*, December 2003.

[69] D. Nicol and R. Fujimoto, "Parallel simulation today," *Annals of Operations Research*, vol. 53, pp. 249-286, December 1994.

[70] R. Bryant, "Simulation of packet communication architecture computer systems," MIT-LCS-TR-188, Massachusetts Institute of Technology, 1977.

[71] K. Chandy and J. Misra, "Distributed simulation: a case study in design and verification of distributed programs," *IEEE Trans. on Software Engineering*, vol. 5, no. 5, pp. 440-452, September 1979.

[72] W. Su and C. Seitz, "Variants of the Chandy-Misra-Bryant distributed discrete-event simulation algorithm," *SCS Multiconference on Distributed Simulation*, March 1989.

[73] W. Cai and S. Turner, "An algorithm for distributed discrete-event simulation - the carrier null message approach," *SCS Multiconference on Distributed Simulation*, January 1990.

[74] D. Nicol, "Principles of conservative parallel simulation," *Winter Simulation Conference (WSC'96)*, December 1996.

[75] D. Nicol and J. Liu, "Composite synchronization in parallel discrete-event simulation," *IEEE Trans. on Parallel and Distributed Systems*, vol. 13, no. 5, pp. 433-446, May 2002.

[76] D. Jefferson, "Virtual time," *ACM Trans. on Programming Languages and Systems*, vol. 7, no. 3, pp. 404-425, July 1985.

[77] D. Jefferson, "Virtual time II: storage management in conservative and optimistic systems," *ACM Symposium on Principles of Distributed Computing (PODC'90)*, August 1990.

[78] B. Lubachevsky, A. Shwartz, and A. Weiss, "Rollback sometimes works...if filtered," *Winter Simulation Conference (WSC'89)*, December 1989.

[79] S. Turner and M. Xu, "Performance evaluation of the bounded time warp algorithm," *Workshop on Parallel and Distributed Simulation (PADS'92)*, January 1992.

[80] S. Das, R. Fujimoto, K. Panesar, D. Allison, and M. Hybinette, "GTW: a time warp system for shared memory multiprocessors," *Winter Simulation Conference (WSC'94)*, December 1994.

[81] P. Reynolds, "A spectrum of options for parallel simulation," *Winter Simulation Conference (WSC'88)*, December 1988.

[82] L. Sokol, B. Stucky, and V. Hwang, "MTW: a control mechanism for parallel discrete simulation," *Int'l Conference on Parallel Processing*, August 1989.

[83] P. Dickens and P. Reynolds, "SRADS with local rollback," *SCS Multiconference on Distributed Simulation*, 1990.

[84] J. Steinman, "Breathing time warp," *Workshop on Parallel and Distributed Simulation (PADS'93)*, May 1993.

[85] P. Martini, M. Rumekasten, and J. Tolle, "Tolerant synchronization for distributed simulations of interconnected computer networks," *Workshop on Parallel and Distributed Simulation (PADS'97)*, June 1997.

[86] A. Boukerche, S. Das, and A. Fabbri, "SWiMNet: a scalable parallel simulation testbed for wireless and mobile networks," *ACM/Kluwer Wireless Networks*, vol. 7, no. 5, pp. 467-486, 2001.

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

[88] Wireless and mobility extensions to ns, available at http://www.monarch.cs.cmu.edu/.

[89] 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.

[90] 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.

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

[92] J. Liu, L. Perrone, D. Nicol, M. Liljenstam, C. Elliott, and D. Pearson, "Simulation modeling of large-scale ad-hoc sensor networks," *European Simulation Interoperability Workshop (SIW 2001)*, June 2001.

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

[94] 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.

[95] J. Liu and D. Nicol, "Lookahead revisited in wireless network simulations," *Workshop on Parallel and Distributed Simulation (PADS 2002)*, May 2002.

[96] http://www.opnet.com/.

[97] 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.

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

[99] 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.

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

[101] 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.