Scalable Location Management for Geographic Routing in Mobile Ad hoc Networks

by

Sumesh J. Philip

(July 22, 2005)

Major Professor: Chunming Qiao, Ph.D.

A dissertation submitted to the

Faculty of the Graduate School of

The State University of New York at Buffalo

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

Department of Computer Science and Engineering

To Julie, without whose loving support, encouragement and patience, this would have been a never ending journey.

ACKNOWLEDGMENT

I would like to thank my advisor, Prof. Chunming Qiao, for his guidance and encouragement, without which this work would have been impossible. His professional advice and insights have always led me in the right directions in research and career. I would like to thank my committee members, Prof. Shambhu Upadhyaya and Prof. Hung Ngo, for their valuable suggestions and comments. Thanks also goes to Dr. Moncef Elaoud at Telcorida Technologies, for being my thesis outside reader and offering insightful comments.

Words cannot express my deep sense of gratitude to my family for their constant support and encouragement. Special thanks to my son, Aiden, for those welcoming breaks while writing this thesis.

I would especially like to thank my friend and colleague, Dr. Vishal Anand, for his encouraging words and glimpses of what lies ahead at the end of the tunnel.

Thanks to my fellow colleagues: Joy Ghosh, Sami Sheeshia, Dahai Xu, Dr. Hongyi Wu, Dr. Jikai Li, Dr. Xiaojun Cao, Vivek Kumar, Mohit Virendra, Dr. Ramkumar Chinchani, Suranjan Pramanik, Peng Lin, and Sunu Mathews for their companionship and innumerable discussions that made my research life a memorable and enjoyable one.

Last but not least, my sincere thanks to all the teachers I have had in my entire student life who have played a vital role in getting me to where I am today. They will always be remembered for their efforts that provided a strong foundation over which I have built my career.

Contents

List of Figures

List of Tables

ABSTRACT

While many solutions have been proposed for routing in mobile ad hoc networks, only a few have considered the issue of scalability of these protocols in networks having node membership in the order of thousands spread over a large area. Geographic routing using source–destination locations has been widely suggested as a scalable alternative to conventional routing approaches in mobile ad hoc networks. However, efficient location management algorithms are required to discover a destination's location before data transfer can be attempted using geographic routing. To be deemed scalable with respect to network size, mobility and traffic, the signalling overhead due to location management must be kept low so that the performance of geographic routing is minimally affected.

In this research, we introduce a novel location management protocol known as Scalable Location Management (SLALoM), which outlines a scheme for partitioning a given terrain into ordered regions for location management. Our detailed analysis shows that under random node mobility and communication requirements, SLALoM improves upon the asymptotic location management cost compared to existing location management schemes. As an optimization, we use the concept of local forwarding to introduce a scheme called Efficient Location Forwarding (ELF) that mitigates the location update cost of SLALoM. We show that, while the asymptotic overhead cost by such an improvisation matches that of SLALoM, ELF outperforms SLALoM in practise.

Noting that a two-level hierarchy leads to an overall reduction in the location management cost, we investigate the use of multilevel hierarchy to further minimize the sig-

nalling cost and make efficient use of the limited bandwidth of the wireless channel. We propose a novel grid ordering scheme known as Hierarchical Grid Location Management (HGRID) that yields only a logarithmic increase in the location update cost with respect to the number of nodes in a uniformly and randomly distributed ad hoc network. We also show that, under a specific framework, all the proposed protocols are scalable with respect to mobility and network size. We carry out extensive simulations to quantitatively compare the performance of the protocols under practical considerations that could not be incorporated into the analysis, and to study how location management can affect geographic routing.

While the proposed protocols perform well in dense networks, they may suffer from protocol incorrectness caused by low node density. In order to tackle this problem, we introduce proxy based location management, a novel enhancement that can be used in conjunction with Face Routing in planar graphs to operate efficiently in sparse or irregular network conditions. We show that Face Routing on the planar graph constructed from the unit disk graph may fail to discover proxy server regions. Accordingly, we define an overlay graph as one in which a graph edge is defined between two unit regions if a radio link exists between any two nodes located in these regions. The Connected Overlay Planar Graph Construction problem is to construct a connected planar graph from the overlay graph. We present a polynomial time, centralized algorithm that solves the problem, and propose a novel routing protocol based on distributed version of this algorithm known as Grid Traversal Algorithm (GTA) which performs well in a wireless network due to its localized nature and low control overhead.

Finally, we note that an efficient unicast routing scheme must be amenable to support resource efficient multicasting. While the Steiner Tree problem is known to be NP-Hard in literature, we investigate efficient heuristics that incorporate node locations for approximating a minimum cost delivery tree for group communications in large ad hoc networks. We propose Location Guided Core (LGC), a novel protocol that integrates location management and multicast tree construction into a single framework. The protocol is purely distributed and localized, and achieves a sub-linear reduction for the average control overhead over flooding, which is predominantly used to disseminate node locations for location based multicasting in existing protocols described in literature.

Chapter 1

Introduction

Since their emergence more than twenty years back, wireless networks have gained popularity world wide. This is particularly true with the introduction of mobility to wireless nodes. In the last decade alone, wireless subscribers and traffic has seen an unprecedented growth compared to the traditional telephone switching system. Architecture wise, wireless networks can be classified into two categories - those that require an infrastructure set up to aid their operation [1], [2], [3] and those that operate without the aid of any centralized administration or support services [4], [5], [6]. Wireless networks that belong to the first category typically make use of a fixed architecture consisting of *Base Stations*, *Base Station Controllers* and *Mobile Switching Centers* for inter node communication, and wireless communication is usually over a single hop i.e. a full duplex channel between the mobile node and the Base station. Examples of such networks are Cellular networks for mobile communication over a wide geographic area, and WiFi/WLAN [7], [8] hotspots for short range high speed data communication. On

the other hand, multi-hop ad hoc networks are truly dynamic and operate on the fly. There is no centralized control nor fixed routers, and hence each mobile node acts as a router to forward packets intended for destination nodes. Feasibility of such networks can be quite useful for dynamic applications such as conferences, search and rescue missions, disaster relief, automated battlefields, and sensor networks, to name a few [9], [10].

Routing in ad hoc networks has posed an interesting challenge in the research community, in which finding and maintaining a route between a source–destination pair in a communication session imposes a major hurdle in designing an efficient routing protocol for such networks [11]. Due to mobility, the network topology varies frequently, and end-to-end sessions are subject to link failures constantly. Many protocols have been suggested for solving this problem, and these can be broadly classified into two: *proactive* and *reactive* protocols. Proactive protocols are table driven [12] [13], and continuously evaluate routes to all other nodes in the network, while reactive protocols compute routes only on-demand [14], [15]. A third class of protocols, known as *hybrid* protocols, takes advantage of the best of both proactive and reactive protocols [16], [17]. While many solutions have been proposed for routing in mobile ad hoc networks, few have considered the issue of scalability of such protocols in networks having node membership in the order of thousands, and that are spread over a large geographic area. A unique characteristic of ad hoc networks is that the limited bandwidth of the wireless channel is shared by signalling traffic as well as data, and the former is given a higher priority than data. This works fairly well for routing protocols in small networks with

low node mobility, since the volume of signalling traffic is low enough to carry out the route discovery and maintenance phases without disturbing the data traffic. However, increased node mobility and node membership can lead to excessively high signalling traffic, leading to congestion and poor network performance [18], [19]. Intuitively, any routing protocol that tries to maintain state (e.g. a pre-computed source route, network topology) for routing purposes, appears non-scalable for ad hoc networks, since maintenance of the state requires additional signalling over the entire network.

Recent advances in positioning techniques such as GPS [20] and other ad hoc localization [21] has motivated researchers to pursue better routing schemes that take into account the physical location of the node (location is usually represented as a tuple consisting of latitude, longitude and altitude or as a point in space using cartesian coordinates). Many schemes have been proposed for position based routing in static ad hoc networks that guarantee loop free routes from a source to a destination [22]. One of the key observation from all of the position based routing algorithms is that the routing decision at an intermediate node is solely based on its position, its locality information (position of neighbors), and the position of the destination. Nodes can periodically broadcast short packets containing their identities as well as locations so that each node is aware of its neighborhood. Another motivation for using geographic routing is that link breakages do not necessarily result in routes getting broken, since packets can be readily forwarded via alternate links, and are guaranteed to be delivered as long as a path exists in the network. Clearly, only localized algorithms provide scalable solutions, and geographic routing is indeed a potential candidate for scalable routing in a

critically power/bandwidth constrained network.

Although geographic routing may be the key in providing a scalable solution for routing in ad hoc networks, an important requirement for any position based routing algorithm is the need for an accurate position information of the destination. Thus, a fundamental problem in geographic routing is maintaining the locations of nodes in a distributed manner in an ad hoc network such that the position of a required destination can be determined with minimal effort. Flooding based schemes can be used for determining the position of the destination before the actual routing, but these are not scalable with respect to geographic routing. Location management has been exhaustively studied in conjunction with Cellular networks, but the dynamic nature of the network as well as the scarcity of bandwidth makes this problem more interesting and challenging in ad hoc networks. A few location management schemes can be found in literature [23], [24], but all have a cost complexity that grows as $\sqrt{N}$, where N is the number of nodes in the network. We are motivated to find efficient schemes that have a lower cost complexity than existing location management algorithms, and hence, better suited for location management in geographic routing.

In this proposal, we introduce three *novel* techniques – *SLALOM*, *ELF* and *HGRID* – that can be used to reduce the location management overhead in geographic routing networks. The main idea in our algorithms is to divide the terrain into unit regions, and combine these regions into groups in a specific way such that the location management primitives such as location update, maintenance and querying within these groups result in minimal control overhead. In *SLALoM* (Scalable Location Management), we

define a *two–level hierarchy* and *near/far* home regions such that near home regions are updated more frequently than far home regions. This optimization results in a better location management cost for *SLALoM*, and we show that the average location management cost increases only as $N^{\frac{1}{3}}$, an asymptotic improvement over $\sqrt{N}$. Currently, this is the best known upper bound for asymptotic average location management cost found in literature. While the asymptotic location management cost is of theoretic interest, simulations show that the location update scheme in *SLALoM* causes localized congestion in the network. We use the concept of *location forwarding* to create a new protocol called *ELF* (Efficient Location Forwarding) to improve the practical performance of *SLALoM*. We find that, while the asymptotic overhead cost due to this improvisation matches that of SLALoM, *ELF* outperforms *SLALoM* in average case scenarios.

While the overhead in the previous two protocols have been derived under the assumption that any node can randomly initiate a network connection to any other node in the network, and move arbitrarily any where in the network, this may not be true in practise for certain applications envisioned for ad hoc networks. Particularly, node movement may be limited to a specific region in the network. There may only be few occasions when a node is required to traverse the entire network diameter frequently. Another observation is that there is a good chance that communication needs may frequently arise between nodes that are geographically located close to each other than ones that are located far away. Under these considerations, one wonders about the effect of employing a multi–level hierarchy for location information management, in which the clarity of location information is higher across lower order leaders, while higher order leaders

have a better overall view of the network. Thus, we define a new grid ordering scheme for Hierarchical Grid Location Management, where the hierarchy is defined by the position of the node in the specific locale of the terrain. We show that, the asymptotic location update cost – which forms the majority of the location management overhead – increases only as $\log N$. Simulation results show that this decreased overhead results in a much improved performance for HGRID over existing location management protocols, and is a promising contender for location management in a wireless ad hoc network architecture.

One of key requirements of the proposed grid based location management protocols is that the network density must be sufficient to contain nodes in server regions such that the protocol packets always reach their intended destinations. While the probability of protocol incorrectness is low for densely deployed ad hoc networks, this may not be the case when the server regions become empty due to lack of sufficient node density or high node mobility. Such a condition, in which there are no nodes in a server region is called the *empty region* problem, and we investigate the effect of node density and mobility on the empty region problem. We outline a proxy based location management scheme to combat this problem wherein an adjacent non–empty region is delegated the responsibility of taking over the server duty in the event that a server region becomes empty. The concept of proxies can thus be used in conjunction with grid based location management schemes to overcome the problem caused by low network density. However, the proxy scheme requires the support of an efficient geographic routing protocol to seek out adjacent non–empty regions, and we show that current geographic routing

protocols described in literature may cause the proxy scheme to operate inefficiently. We define on *Overlay Graph* as one in which the unit regions become the vertices in the graph, and a bidirectional edge exists between two vertices in the overlay graph if there are two mobile nodes within these regions that are in radio range of each other. The connected overlay planar graph construction problem is then to construct a connected planar subgraph using the overlay graph. We present a polynomial time, centralized algorithm using a specific property of such graphs to solve the problem. A planar graph face routing protocol called Grid Traversal Algorithm (GTA) is then proposed which is effective in aiding the proxy location management scheme.

One of the key requirements of an efficient network protocol is that it must be able to support multicast operation due to group communication between the nodes. While many protocols have been suggested for constructing multicast delivery trees in ad hoc networks [25] [26], [27], [28], most of these rely on source based trees or core trees based on the network topology. Due to change in the topology caused by mobility, all the protocols mentioned resort to either flooding the entire network to reconstruct the multicast tree or periodic heartbeat messages from all nodes in the multicast core to update the state of the tree. Clearly, such schemes are not scalable due to the excessive control overhead that grows with an increase in the number of multicast nodes per group or with an increase in number of multicast groups themselves. While the Steiner tree problem is known to be NP-Hard [29], we note that any protocol that tries to build a minimum cost delivery tree can only construct an approximate tree as the network size grows, and even so with complete information of the network topology. In other words,

if one were to build an approximate tree, then the control overhead needed to keep the topology information up-to-date across all the node would be enormous. We take an alternate stance: instead of trying to minimize the data delivery cost by creating the best possible multicast tree, create an approximate tree that may not be as efficient but construct such a tree with a much lesser control overhead and without giving in on the network throughput.

We investigate the use of node locations in constructing a scalable multicast protocol. Our work improves upon the Location Guided Steiner protocol [30] that approximates a minimum cost multicast tree by flooding node locations and distributively constructing such a tree using the node locations. We use the concept of a multicast core similar to that in [31] to build a shared core tree among the multicast senders and receivers. While core based protocols are known to be non–optimal, they have the distinct advantage that the amount of state the routers need to maintain is less than their source tree based counterparts. Our protocol takes the core idea to a new level in the sense that the core construction is location based, and involves only a subset of the nodes to construct/maintain the core. The use of locations further reduces the overhead required to operate the protocol and we show that the average control overhead achieves a sub-linear reduction in the number of nodes compared to that of flooding as the network size grows.

The rest of the thesis is organized as follows. We introduce the challenges in unicast routing and the issue of scalability in ad hoc routing in chapter 2. Chapter 3 introduces three novel location management protocols, namely SLALoM, ELF and HGRID. Chap-

ter 4 presents our research results on the scalability properties of the proposed protocols and describes the performance studies carried out via simulations to analyze how the protocols stack up against each other with respect to various network parameters such as network size, mobility and traffic. We outline the empty region problem caused by low network density and our proxy based solution in Chapter 5. A multicast extension to grid based location management is described in Chapter 6 and we conclude this work with some future research directions in Chapter 7.

Chapter 2

Background and Related Work

This chapter introduces the background and motivation for our research, and outlines the some of the challenges and solutions described in the ad hoc network literature.

2.1 Routing in Mobile Ad hoc networks

Since its inception, the Internet has existed as a network with a fundamentally *quasi–static* topology. While the Internet was designed in a distributed fashion with an eye for adapting to topology changes (due to link outages and router failures), its routing technology was not designed for node mobility or drastic topological changes. Due to mobility, the network topology in an ad hoc network varies frequently, and end-to-end sessions are subject to link failures constantly. The limited bandwidth which is shared between control and data makes it almost impossible for nodes to maintain an up–to–date view of the network topology. Hence, the design of an efficient routing protocol

for mobile ad hoc networks must have the following considerations:

- Distributed operation: Since nodes may arbitrarily leave and join the network, and the network is subject to unpredictable node failure due to power drainage or sleeping, link failures due to channel conditions and network partitions, the routing task has to distributed and not be borne by just a few nodes in the network.

- Loop freedom: Due to unstable states in the network, a small fraction of packets may spin around in the network for arbitrary time periods. Ad hoc solutions such as TTL values can bound the problem, but a more structured and well-formed approach is generally desirable as it usually leads to better overall performance.

- On demand operation: Instead of assuming an uniform traffic distribution within the network (and maintaining routing between all nodes at all times), let the routing algorithm adapt to the traffic pattern on a demand or need basis. If this is done intelligently, it can utilize network energy and bandwidth resources more efficiently, albeit at the cost of increased route discovery delay.

- Proactive operation: In certain contexts, the additional latency demand-based operation incurs may be unacceptable. If bandwidth and energy resources permit, proactive operation is desirable in these contexts.

- ”Sleep” period operation: As a result of energy conservation, or some other need to be inactive, nodes of a MANET may stop transmitting and/or receiving (even receiving requires power) for arbitrary time periods. A routing protocol should be able to accommodate such sleep periods without overly adverse consequences.

- Scalability: Increased node mobility and network size can lead to excessively high signalling traffic and delays, leading to congestion and poor network performance. In this work, we devote to this issue, which concerns applications involving thousands of nodes spread over a large geographic area.

In summary, the networking opportunities for MANETs are intriguing and the engineering tradeoffs are many and challenging. A diverse set of performance issues requires new protocols for network control. Literature describes many protocols to route in MANETs [32], [11]. In general, these can be classified into two: *proactive* or *reactive*. Proactive protocols are table driven, and continuously evaluate routes to all nodes in the network, while reactive protocols compute routes only on-demand. We describe a few popular protocols described in literature.

2.1.1 Proactive Protocols

Proactive protocols a.k.a table driven protocols (the name deriving from their use of tables in trying to maintain an up-to-date view of the network by exchanging these tables) are usually modifications of well known routing protocols for their wired counterparts; namely - distance vector or link state protocols [33]. These are also *flat* routing protocols in the sense that they try to compute the shortest path between any two pairs of nodes given the topology information. Since the topology is known well before, communication sessions have minimal delay during initialization since the next intermediate node for packet forwarding is already known. In Global State Routing [12], the key motivation is to maintain topology information by exchanging link states of

known destinations between neighbors periodically. On the other hand, the Wireless Routing Protocol [13] tries to avoid the well known *count–to–infinity* problem due to link failures by using the predecessor to destination information in routing exchanges. In the Destination Sequenced Distance Vector (DSDV) protocol [34], a methodology similar to the Distributed Bellman Ford algorithm is adopted, in which route entries are tagged with a *sequence number* to indicate the freshness of the entry. More recently, an optimized version of the link state protocol using multi–point relaying was suggested in [35]. However, in a highly dynamic environment, some of these protocols prove quite ineffective in trying to maintain a unified view of the network topology across all nodes due to the large size of routing messages that consume the already scarce bandwidth in wireless networks [36].

2.1.2 Reactive Protocols

Reactive protocols take a different approach to routing by computing routes only when necessary. When the need for a communication session arises, the source node broadcasts a *route query* packet which is flooded across the network in an attempt to find a route to the destination. If a route is found (either by the destination or another node that previously knew a route to the destination), a *route reply* is sent back to the source. The source can potentially choose from multiple such replies and chooses the best path it obtains via the route discovery process. However, routes are error prone due to node mobility, in which case, the intermediate node that discovers the broken route sends a *route error* packet back to the source. The source then proceeds to restart the discovery

process (if it does not already have an alternate path in its cache) to find a new route to the destination. Data packets carry the entire source route as in Dynamic Source Routing (DSR) [14] or follow the forward (backward) path set up by the route discovery process in Ad hoc On Demand Distance Vector (AODV) [15] routing. The route selection process may be nodal degree based as in Associativity Based Routing (ABR) [37] or signal strength based as in SSR [38]. The Temporally Ordered Routing Algorithm (TORA) [39] is yet another on demand routing protocol which creates a destination oriented acyclic graph for multiple routes to the destination.

2.1.3 Issue of Scalability in Current Solutions

Although most of the protocols described in the previous sections try to adapt to the dynamic nature of the network and the constraints imposed by mobility, none of them have been shown to scale with either network size or increased node mobility. Table driven protocols usually try to optimize on the shortest path between the source destination pair, but add considerable control overhead that increases as $O(n^2)$ to maintain a snapshot of the network across all nodes. [36], [40] has shown that table driven protocols perform poorly with respect to on demand protocols in terms of network control overhead, achievable throughput as well as per packet delay in many scenarios, including networks that have node memberships in the order of tens of nodes. Additionally, caching techniques and *implicit source routes* were proposed in [41] and [42] to reduce the overhead in on demand routing. Ironically, even though reactive protocols only incur overhead for route construction and maintenance when needed (which gives them

the upper hand over proactive protocols), they too suffer from limited scaling properties due to the overhead incurred in carrying out route discovery and maintenance in large networks [43], [44] due to network wide flooding. Route discovery, being a broadcast mechanism, can flood the network and the impact of this behavior in large networks will be significant. Also, since path lengths are longer on the average when the network diameter increases, communication sessions are prone to multiple link breakages. This results in multiple nodes initiating the route recovery mechanism using error packets. These error packets may not reach the source due to upstream link failures along the reverse path, and even if they do, the source node will be unable to make a repair before another link in the route breaks. While hybrid solutions that combine the best of both reactive and proactive protocols have been proposed in literature [16] [17], these suffer due to the inherent drawbacks in their respective reactive/proactive components in large networks.

Clustering and hierarchical network organization has been a well known technique to achieve scalability in wired networks [45], [46], and there has been considerable work in this area to adapt this concept for organizing multi–hop wireless networks into hierarchical clusters for routing scalability [47], [48], [49], [50], [51], [52], [53]. However, the original motivation behind hierarchical routing is to reduce the memory requirements for storing routing table in internetworks of thousands of networks consisting of millions of hosts, and not to withstand the unpredictable and dynamic nature of mobile ad hoc networks. Even it were possible to form unique clusters and organize them into a hierarchical network (which in itself is a daunting task), maintenance of this hierarchy

with mobility would be extremely difficult, given the dynamic topology changes and the limited bandwidth by which the reconfiguration information must be propagated across the nodes. Cluster head changes can cause a *rippling effect*, where the role change of a leader in the higher level of hierarchy results in subsequent changes in lower leaders all the way to the bottom of the hierarchy. Another difficult task in maintaining a hierarchical network is the assignment and binding of node addresses to hierarchical addresses, which is primarily used for routing in the hierarchy. Periodic or triggered registration/binding of addresses to hierarchical addresses, and the query–response for obtaining the hierarchical address before the start of a communication session add to the control overhead, and hence, congests the network. In addition to all these, the path used for routing may be sub–optimal than the shortest path – an inherent tradeoff in hierarchical routing – which may lead to higher bandwidth consumption.

2.2 Geographic Routing

Recent advances in positioning Several schemes have been proposed in literature to locate wireless nodes in the context of indoor and outdoor applications. These solutions are GPS based [54], [20], infrastructure based [55] or cellular network based [56], [57], [58] for mobile location tracking in an outdoor environment. For tracking nodes in an indoor environment, the RADAR system [59] builds a signal fingerprint of the entire region, and matches a node's received signal strength to obtain its location. In the CRICKET location support system [60], nodes receive periodic radio and ultrasound

signals from installed base stations, and associate themselves with the closest base station. More recently, [61], [62], [63], [21], [64], [65] describe distributed algorithms to obtain node positions in ad hoc networks where location unaware nodes identify their locations with the aid of more capable nodes that already know their locations. Recently, a new family of protocols have been proposed that take advantage of these localization algorithms to use the position of nodes to aid in routing in wireless multi-hop networks with an eye on achieving routing scalability. A survey of position based routing algorithms can be found in [66] and [67]. We describe representative schemes from the above in the following section.

2.2.1 Greedy Routing Algorithms

Finn's *Cartesian routing* [68] is the earliest known position based routing mechanism found in literature for datagram routing in wired networks. Greedy forwarding has been an efficient contender for position based routing in wireless packet networks. In [69], Takagi et. al. suggested *Most Forward within Radius with backward progression* (MFR), in which all nodes are aware of the positions of their active neighbors within their transmission radii and packets to a destination are forwarded to the node that makes most forward to the final destination location. If no nodes are in the forward direction, the intermediate nodes transmits to the least backward terminal, if any. Thus, MFR tries to minimize the number of hops a packet has to traverse in order to reach the destination. Under a fixed transmission range for all nodes and spatial distribution of nodes according to a two dimensional Poisson process, [69] proved that the

optimal number of nodes covered by a single transmission should be eight such that the forward progress made during each transmission is maximized. In figure 2.1, source *S* forwards a packet to *C* using MFR for a packet destined for *D*. Although MFR per-

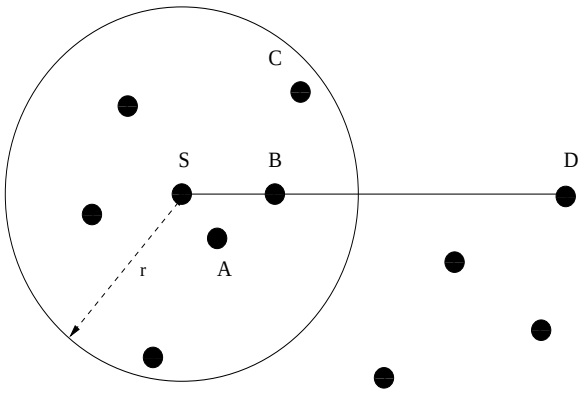


Figure 2.1: Greedy Forwarding

forms in scenarios where the transmitter is able to control the transmit power according to the distance between the sender–receiver pair, yet another strategy known as *Nearest Forward Progress* (NFP) [70] performs better than MFR where the sender is able to control the packet transmit power. As per this approach, the packet is transmitted to the sender's nearest neighbor that makes forward progress to the destination. Thus, in figure 2.1, *S* forwards the packet to *A* using the NFP scheme. Intuitively, when a node transmits at full power, although the packet is able to find a receiver that makes maximal forward progress, the increased interference range due to this strategy results in additional collisions, resulting in unsuccessful transmissions. Thus, the average forward progress, denoted by $p \cdot f(a,b)$, where p is the probability of a successful transmission and $f(a,b)$ is the progress made for a transmission from node *a* to node *b*, is higher for NFP than MFR.

2.2.2 The Local Maxima Problem and Recovery via Face Routing

A main disadvantage of greedy forwarding is the occurrence of a local maxima from which it may not recover. For example, in figure 2.2, node x is an intermediate node for a packet destined for node D. However, x does not have any active neighbors that can make forward progress towards D. Thus, a *void* (or a *geographic hole*) is said to have occurred at x, since the greedy forwarding scheme led to the occurrence of a local maxima at x. Although two paths $[x \rightarrow w \rightarrow v \rightarrow D]$ and $[x \rightarrow y \rightarrow z \rightarrow D]$ exist to D, x will not choose to forward to either w or y using greedy forwarding. To recover from

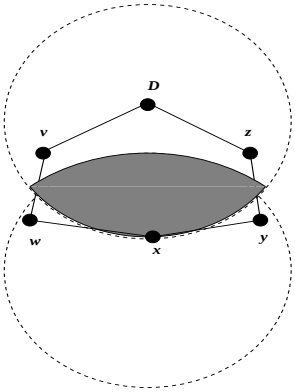


Figure 2.2: The local maxima problem

geographic holes, Karp et. al. [71] suggested the *Greedy Perimeter Stateless Routing* (GPSR) in which a packet follows the path according to the well known *right-hand* in planar graphs rule to traverse around the void. Thus, in figure 2.2, it is possible to traverse around the perimeter $[x \rightarrow w \rightarrow v \rightarrow D \rightarrow z \rightarrow y \rightarrow x]$ by using this cycle traversing property. The concept of creating planar graphs from arbitrary graphs using *Relative Neighborhood Graphs* (RNG) and *Gabriel Graphs* (GG) is well known in literature. Thus, only by knowing the local neighborhood information alone, it is possible

for nodes in an arbitrary graph to remove edges locally to form RNG or GG planar graphs. In [71], the authors adapted the routing protocol to obtain the RNG graph, since they felt that the RNG being a subset of the GG, offered the possibility of an enhanced MAC protocol performance due to reduced interference from a smaller subset of edges. Having formed the planar graph locally, the routing is done as follows: carry out greedy forwarding whenever possible. On encountering a geographic hole, switch to perimeter mode. Figure 2.3 shows how perimeter forwarding occurs in GPSR. A connected pla-

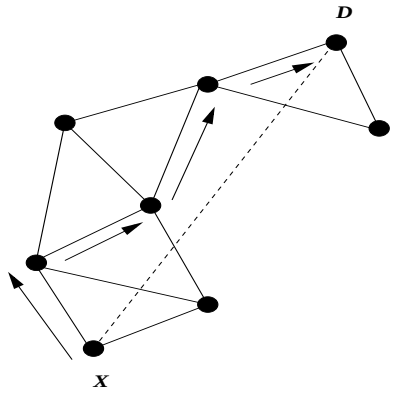


Figure 2.3: Example of Perimeter Routing in GPSR

nar graph partitions the plane into *faces* that are bounded by the polygons made up of the edges of the graph. The concept is to forward the packet on the faces of the graph that are progressively closer to the destination. On each face, the packet is forwarded along the interior of the face by using the right hand rule: forward the packet on the next edge counter clockwise from the edge on which it arrived. Whenever the line between the source and the destination (which indicates the shortest Euclidean path and the best direction to reach *D*) intersects the edge along which the packet is to be forwarded, check if the intersection is closer to any other intersection previously encountered. If

this is true, switch to the new face bordering the on the edge which the packet was about to traverse, and continue to forward the packet according to the right hand rule. This algorithm guarantees the successful delivery of a packet to D if a path does indeed exist, or the packet gets dropped at an intermediate node which notices the repetition of forwarding the packet at the start of the cycle.

Independently, Bose et. al. suggested the Face-II [22] algorithm using a similar concept to GPSR to route messages using the right hand rule in planar graphs. They assume a *unit graph* model in which a edge (u,v) exists between nodes u and v, if $dist(u,v) \leq R$, where R is the transmit range. Given an arbitrary graph, nodes can compute the Gabriel Graph locally by removing non-planar edges with the aid of their neighborhood position information. Since the Minimum Spanning Tree (MST) is a subset of both the unit graph as well as GG, the intersection of both must give rise to a connected planar graph. The Face-II algorithm is similar to the graph traversal in GPSR, and routes packets towards the destination by sequentially touring the faces of the planar graph by the right hand rule. Finally, they combine greedy forwarding with Face-II to create a new algorithm called GFG (Greedy-Face-Greedy) that switches to greedy routing from face routing when a suitable node that located closer to the point where face routing started is found.

2.2.3 Scalability of Geographic Routing

One of the key observation from all of the position based routing algorithms is that the routing decision at an intermediate node is solely based on its position, its locality

information (position of neighbors), and the position of the destination. In any distributed routing protocol (except flooding), these are minimal requirements to perform routing. With positioning devices such as GPS becoming cheaper and availability of alternate localization schemes, determining one's own position is no longer an obstacle. Nodes can periodically broadcast short packets containing their identities as well as locations so that each node is aware of its neighborhood. Link breakages will not result in loops if intelligent routing decisions are made. Thus, in terms of required resources and loop freedom, geographic routing gains an upper hand over proactive protocols. Additionally, routing requires minimal state information, and packets need to carry only the destination location information for intermediate nodes to perform routing. Finally, link breakages do not result in routes getting broken, since packets can be readily forwarded via existing links, and are guaranteed to be delivered as long as a path exists in the network. Clearly, only localized algorithms provide scalable solutions, and geographic routing is indeed a potential candidate for scalable routing in a critically power/bandwidth constrained network.

While geographic routing is localized and resource efficient, the average paths can be relatively longer in sparse networks due to the local maxima problem and traversing large faces in the resulting planar graph. The spanning ratio (defined as ratio of the length of the shortest path between nodes u and v measured by Euclidean distance to that of the Euclidean distance between u and v) due to either GG or RNG can be $\theta(\sqrt{N})$ in the worst case, where N is the number of nodes [73]. However, improvements to face routing were proposed in [74], [75] and [76] using constrained face traversals and

shortcuts using internal nodes to cut down on the redundant transmissions.

2.3 The Location Management Problem

Although geographic routing may be the key in providing a scalable solution for routing in ad hoc networks, a key requirement for any position based routing algorithm is the need for an accurate position information of the destination. Most of the literature in position based routing assume the presence of a location service through which the position of the destination can be obtained, but neither considers the details of such a service nor provides an insight into the scalability of geographic routing imposed by this service. One can compare the location management problem to that of ad hoc routing; namely - if it were possible to maintain a snapshot of the network topology across all nodes, any shortest path algorithm can be used to discover routes to destinations. Similarly, if all nodes knew the exact locations of each other, a geographic routing algorithm can be used to deliver packets to the destination. However, the problem at hand in either case is to maintain route/topology information or the current location information with node mobility. Although there has been tons of work on ad hoc routing in literature, there has not been considerable effort into designing efficient location management schemes.

An obvious choice for a simple location service is using a brute force algorithm, i.e. flooding - a *location query* packet can be broadcast across the network, and either the destination or a node that knows the destination's accurate location can then reply to the

query with the current position of the destination. However, flooding has an $O(N)$ overhead compared to the $O(\sqrt{N})$ overhead of geographic routing, and suddenly, position based routing is not as attractive as before for scalable routing. Clearly, an efficient location management protocol is required for position based routing to deemed scalable. Thus, location management is defined as the problem of maintaining the locations of nodes in a distributed manner in an ad hoc network such that the position of a required destination can be found with minimal effort. The efficiency of such a protocol can then evaluated solely based on the signalling overhead it creates and its effect on the scalability on geographic routing.

Although the problem has been studied exclusively in conjunction with cellular networks (see [77] for a survey), the dynamic/distributed nature of ad hoc networks makes the problem especially interesting and difficult. First of all, the VLR/HLR (Visitor Location Register/Home Location Register) architecture for location management in cellular networks is a static infrastructure, and can be assumed to be operational while the network is operational. The location management signalling can easily be built into the current communication protocols between the mobile unit and the base station using an additional channel. Another difference is that any signalling between the mobile unit and the database is over a single wireless hop followed by fixed wired infrastructure, and hence, signalling bandwidth is not as much a concern as in ad hoc networks. Nodes can be powered down, sleeping or even be partitioned in ad hoc networks, hence choosing a set of nodes as location servers can be non–trivial in ad hoc networks. Since the bandwidth is a scarce commodity in ad hoc networks, the protocol should be careful to

avoid large volume of signalling over many hops.

2.3.1 Related Work

Routing protocols such as Location Aided Routing (LAR) [78] and Distance Routing Effect Algorithm for Mobility (DREAM) [79] are the earliest known schemes for location based routing in literature. Although these were not directly associated with the location management problem, they can be easily modified to fit the bill. DREAM proactively disseminates node position information so that future data sessions may flood data packets in the direction of the previously known position of the destination. On the other hand, LAR tries to find a source route to the destination and controls the request flood region by restricting it to an an area that includes the previous known location of the destination. However, both these protocols are not scalable, due to their flooding nature of control packets. [80] is the earliest known location management protocol described in literature for ad hoc networks, even though they do not specifically deal with locations in their work. The idea here is to select a subset (quorum) of nodes for storing pertinent information (node locations, for example). Updated information is written to a subset of these nodes. When the intersection of these subsets is non–empty, any of the subset can be read for data written to the subset during a previous write cycle. Since there can be multiple responses to a read request, time stamps can be used to separate the most recent information. *Home region* based location management was independently suggested by [23] and [81]. Location management in [23] is similar to that of Mobile IP, where each node selects a unit region in the terrain as its *home region*

by using a mapping function (a hash function, for e.g.) and the unique address of the node. A node updates its home region when its position changes significantly. Home regions can be later discovered by using the mapping function and queried by source nodes in an on demand fashion for locating destination nodes. Similar to the region based protocols, [82] and [24] suggested alternate grid ordering schemes for achieving scalability in location management.

However, due to the manner in which the location servers are chosen, the average path length in these schemes for updating home regions or location servers as well as querying increase as $O(\sqrt{N})$, where N is the number of nodes in the ad hoc network. Thus, the location management overhead also increases proportionately for these protocols. While such overheads are comparable that of geographic routing, we would like to see if new protocols can be designed which can achieve a better upper bound on the location management cost and thereby assisting efficient geographic routing.

2.4 Summary

In this chapter, we introduced various position based routing strategies that could be potentially employed for routing in large ad hoc networks spread over a wide geographic area. The main attraction of these protocols is that the routing is localized and that packet delivery can be guaranteed solely based on location information at intermediate nodes. We also introduce the problem of location management, in which a source node requires the destination node's location information before it can carry out rout-

ing. Location management can the bottleneck in achieving scalability for geographic routing, and efficient location management protocols need to be developed for this purpose. We also presented a survey on existing location management protocols proposed in literature.

Chapter 3

Scalable Location Management Protocols

Our preliminary research in scalable location management resulted in three distinct grid based schemes. Having a priori knowledge of the network geography, we divide the playground into *G* logical unit regions (also known as *Order-1* regions). The basic idea behind our location management schemes is to denote some of the unit regions in the network as location management entities, where nodes physically located in those regions carry out tasks to manage locations in a distributed manner. The unit regions form the basic building for location management, and hence their design (size) must be done carefully such that the cost of location management is optimized. Also, since we require that any information which needs to be broadcast to all the nodes in the region be done via a constant number of radio broadcasts, we decide to keep the length of the unit region constant. Figure 3.1 shows an example of an unit region whose side is $\frac{r}{\sqrt{2}}$,

where r is the radio range of each mobile node. Note that the division of the terrain

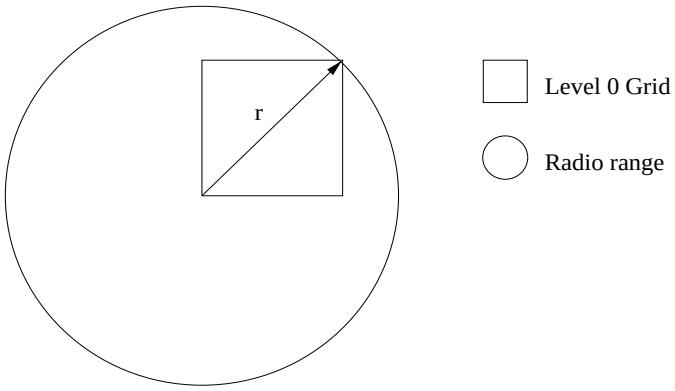


Figure 3.1: An unit region

into unit regions is solely logical and can be done by nodes individually, having *a priori* knowledge of the terrain dimensions as well as the size of a unit region. Once the terrain division is done, nodes can uniquely select specific unit regions to act as location server regions (also known as home regions in the remainder of this chapter). The role of a home region is to keep track of the location of nodes that selected this region as their home region. Instead of choosing individual nodes as servers, our notion of a server region delegates the responsibility of location management to all nodes present in that particular region. This way, even if nodes leave or die in the network, the location management protocol will not suffer.

In general, the location management protocols proposed in this chapter have a similar outline as follows:

- Divide the terrain into well ordered unit regions. Regions may be flat, or aggregated for the purpose of routing scalability. Each node selects one region as its *server* region. The mapping between a node and its server region is unique

and protocol dependant. The mapping is done such that other nodes who wish to know a node's server region can easily do so.

- Each node carries out the *update* (or registration) phase, in which update packets, containing the current location of the node, are geographically routed to server regions. The position of the server region is indicated by a unique point inside the region (such as the lower left corner or the middle). The update phase can be triggered by a timer (periodic) or by node mobility (crossing grid regions). When a node that resides in the server region receives the update packet for the first time, it carries out a region wide *geocast* to update all other nodes that are resident in that region of the updating node's current location. Thus, mobile nodes that are currently resident in a server region form the location *servers* for all the nodes registered to that region.

- When a node moves into a new unit region, it also carries out a *maintenance* phase, in which it requests nodes already present in that region to forward new location information that it must store as part of its server duty.

- Finally, when a source node needs to find the location of a destination node, it queries the destination node's server region using a *query* packet. The query phase is terminated by the *response* phase, in which the first node to receive the query will respond with the latest known location of the destination node. Data is then routed to the destination node by the source node using this location and a position based routing protocol.

In the following subsections, we outline the specific location management protocols in detail. The first protocol, SLALoM, makes no assumption regarding either the mobility of nodes or traffic pattern. On the other hand, the multilevel hierarchical protocol performs best in scenarios where both mobility and traffic are localized.

3.1 Scalable Location Management (SLALoM)

Grid Ordering: Given a square region of area A, SLALoM [83] divides the topography into G logical unit regions (referred to as *Order-1* regions), where each node is aware of the size of the topography as well as the size of an Order-1 region. It then combines K^2 Order-1 regions to form Order-2 regions, where K is a variable and will determine the optimum location management control overhead. Each node selects a *home region* in each Order-2 region via a function F that maps roughly the same number of nodes to each Order-1 region in an Order-2 region. As mentioned before, a *home region* is simply an Order-1 region whose member nodes are responsible for keeping track of the node locations which selected that Order-1 region as their home regions. Thus, due to this specific division of the terrain, every node has $O(\frac{A}{K^2})$ home regions in A (note that since the original square cannot be perfectly tiled with Order-2 regions, it is possible that some nodes may not have home regions in the Order-2 regions adjacent to the boundary of A). Also, if a node u is present in an Order-1 region R_i, which lies in an Order-2 region Q_i, then all home regions of u that lie in or adjacent to Q_i are considered *near* home regions, while the rest are considered *far* home regions.

Remark 3.1.1. Let u and v be nodes in the network. Regardless of where u is located, there is a home region of v that is within $\sqrt{2}K$ of u.

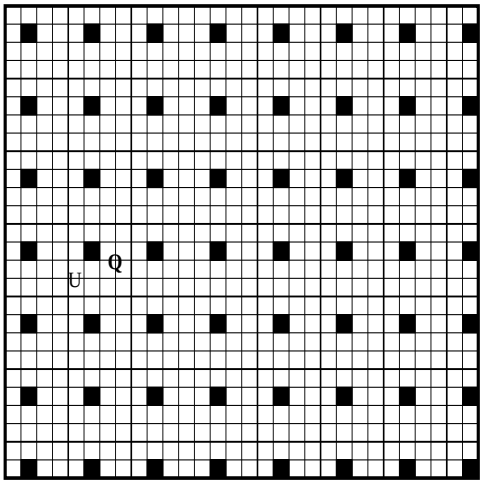


Figure 3.2: Grid ordering in SLALoM

Example 3.1.2. Figure 3.2 shows a sample square topography divided into Order-1 and Order-2 regions, where an Order-2 region consists of 16 Order-1 regions (i.e. $K = 4$ in this example). The shaded regions indicate node u's home regions in each Order-2 region. The shaded region in region **Q** and the eight shaded regions around **Q** represent u's near home regions while the remaining are far home regions.

Location Management in SLALoM: All nodes present in a home region of u act as location servers for u, and keep an entry for the location of u in their location database. When u moves across two Order-1 regions R_i and $R_{i'}$, it does the following:

- If R_i and $R_{i'}$ are in the same Order-2 region Q_i, u informs all it's near home regions of the movement, by a partial location update.

- If R_i is in Q_i, and $R_{i'}$ is in a different Order-2 region $Q_{i'}$, u updates all home regions of the movement by a full location update.

- u also requests nodes in $R_{i'}$ about the location information it has to keep for nodes that have selected $R_{i'}$ as a home region.

A home region is updated by sending a location update packet to the region, and the first location server to obtain the packet will carry out a broadcast in the region to update all location servers in that region about the movement of u. Multiple home regions are informed by location updates that traverse a multicast tree such that each update traverses a distance K between two home regions. The length of such a tree is then $O(\frac{A}{K})$. Thus, it is relatively easy to understand that all home regions know that u is in $Q_{i'}$. In addition, all near home regions know that u is in $R_{i'}$.

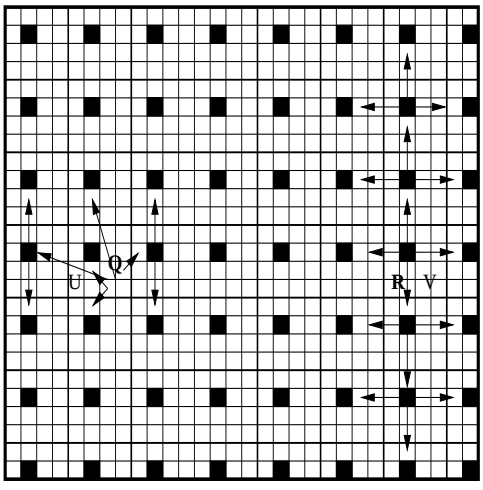


Figure 3.3: Location Update in SLALoM

Example 3.1.3. Figure 3.3 shows a typical location update process in SLALoM. Node

u updates its near home regions around region **Q** due to a local grid crossing, while node *v* carries out a full location update of its home regions across the network, having crossed an Order-2 region into region **R**.

Discovering a node's location: A node *v* wishing to communicate with another node *u* uses the mapping function to identify the closest home region of *u* and sends a query packet to it. If the home region is a near home region, a response is generated by the location server node *z* that receives the query with *u*'s exact location. If the home region is a far home region, *z* forwards the message to the closest near home region of *u*, and the location server in that region which receives this message forwards it to the exact location of *u*.

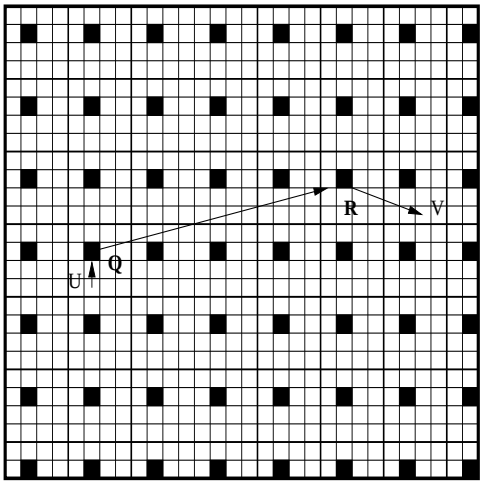


Figure 3.4: Location Discovery in SLALoM

Example 3.1.4. Figure 3.4 shows how node *u* discovers the location of node *v* for forwarding data. Node *u* sends a query packet to *v*'s home region nearest to its Order-2 region, **Q**. The first location server on receiving the query forwards it to **R**, since it

realizes that it is in a far home region due to *v*'s last update. Since **R** contains a near home region, a server that gets the query forwards it to the exact location of *v*.

As stated before, SLALoM makes no assumption regarding either the mobility or traffic pattern in the network. Nodes may move across the entire terrain and initiate communications sessions with other nodes located randomly throughout the network. Essentially, this requires multiple home regions in the terrain such that there is some representative server for each node at a finite distance from any other node and that can be queried to obtain the locations of nodes. This is precisely why a node selects a home region in each Order-2 region. However, updating all home regions when a node has moved significantly is costly, and we try to alleviate this cost using near and far home regions. In Chapter 4, we will show that the proper choice of *K*, i.e., the number of Order-1 regions that make up an Order-2 region, has an impact on the total location management cost of the proposed protocol.

3.2 Efficient Location Forwarding (ELF)

From the previous section, we note that the location update phase in SLALoM can be costly in terms of number of transmission required. Namely, each boundary crossing will result in *at least* 9 *anycasts* to near home regions. Additionally, if the boundary crossing is across Order-2 regions, the entire set of that node's home regions have to be updated via multicast. In a bandwidth restricted environment, such updates may cause channel congestion, especially with increasing node mobility. Thus, we need explore

mechanisms to control this overhead without losing the distinct advantages of having multiple home regions. As an optimization, we can delay the location update phase by setting a forwarding pointer from the previous home region to the new home region when a node moves between Order-2 regions. Subsequent such operations can lead to a chain of forwarding pointers from the last known home region of the node to its current home region. Thus, we can save on the number of multicast updates the node would have initialized. The tradeoff for such savings is the longer distance that a query will now have to traverse to reach the current home region of the node. In this section, we outline the proposed protocol in detail.

Grid Ordering: The grid-based topology construction in ELF [84] is similar to that of SLALoM, except that we do not have the concept of near or far home regions. Instead, we have *forwarding* home regions and a *terminal* home region. Thus, similar to SLALoM, there are $O(\frac{A}{K^2})$ home regions, of which only one is a terminal home region, and the rest are forwarding home regions for each node. The terminal home region corresponds to the home region selected by a node in its current Order-2 region. Each node also keeps a counter c_{cross} which keeps track of the number of Order-2 boundary crossings it has made since its last update of A.

Location Management in ELF: When u moves across two Order-1 regions R_i and $R_{i'}$, it does the following:

- If R_i and $R_{i'}$ are in the same Order-2 region Q_i, u informs its terminal home region of the movement.

- If R_i is in Q_i, and $R_{i'}$ is in a different Order-2 region $Q_{i'}$, and if c_{cross} is less than a threshold αK^{β}, u sets up a forwarding pointer between its previous terminal home region (which now becomes a forwarding home region) and its terminal home region

- If c_{cross} is greater than the threshold, u updates all home regions using the multi-cast tree mentioned previously.

- u also requests nodes in $R_{i'}$ about the location information it has to keep for nodes that had selected $R_{i'}$ as a home region.

Discovering a node's location: Similar to SLALoM, a node v wishing to communicate with another node u sends a query packet to the closest home region of u by using the mapping function. There can be two cases:

- If the home region is a terminal home region, a location response is returned by the location server to v with the exact location of u.

- Otherwise, a location response is returned to v with the location of the home region of the last known Order-2 region of u that this home region knows. From here onward, the message from v follows the chain of forwarding home regions set up by u before the message is finally handed to u.

Example 3.2.1. In figure 3.5, Order-2 region **Q** represents the region when node u last updated all the home regions, and Order-2 region **R** contains the terminal home region. The intermediate home regions indicated by the arrows represent the forwarding home

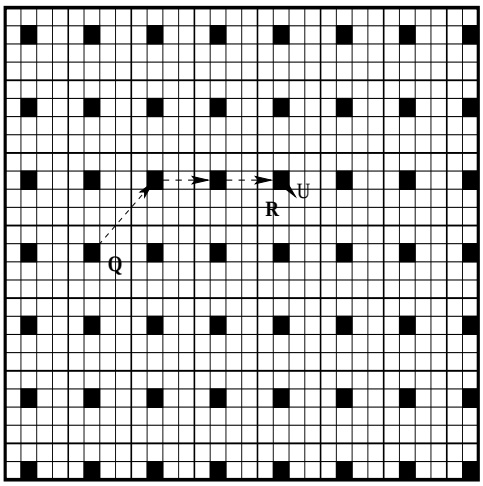


Figure 3.5: The topography is divided into Order-2 regions, each consisting of 16 Order-1 regions. The shaded grids indicate where node U's home regions may be located. Region Q is where U last updated the entire square, and the arrows indicate the forwarding chain set up till the current Order-2 region of U

regions set up by *u*. A query for *u* follows the forwarding chain until it reaches the terminal home region in **R** which knows the accurate location of node *u*.

3.3 Hierarchical Grid Location Management (HGRID)

In the last two sections, we described efficient location management schemes for arbitrary ad hoc networks. i.e, we do not make any assumptions regarding either node mobility or communication requirements between node pairs. Any node can randomly initiate a network connection to any other node in the network, and nodes move arbitrarily any where in the network. However, this may not be true for all type of applications envisioned for ad hoc networks. Particularly, node movement may be limited to a specific region in the network. There may only be few occasions when a node is required to traverse the entire network diameter frequently. Under such circumstances, any lo-

cation management scheme that tries to update location information to nodes located at far ends of the network consumes unnecessary network bandwidth. This would also imply that location update cost should be proportional to node mobility. For instance, if a node is totally stationary upon initialization, its ideal update cost must be zero. A rapidly moving node may incur additional overhead by issuing frequent location registrations which is proportional to its speed. Another observation is that there is a good chance that communication needs may frequently arise between nodes that are geographically located close to each other than ones that are located far away. Hence, location queries for nearby nodes must stay local, while those for nodes that are located far away may be penalized more.

Under these considerations, one wonders about the effect of deploying a multi–level hierarchy for location information management, in which the clarity of location information is higher across lower order leaders, while higher order leaders have a better overall view of the network. In fact, the above considerations are best suited for a hierarchical setup for location servers. However, we note from section 2.1.3 that trying to build a hierarchy in mobile wireless networks is usually a daunting task due to cluster management and hierarchical address maintenance. But, if we are able to design a hierarchy based on unit regions, where the position of a node automatically assigns it a specific role in the hierarchy, we can get rid of complex message exchanges required to initialize and maintain a hierarchy across the network. This observation is precisely the motivation behind the HGRID protocol [85], [86].

For the purpose of clarity, we will refer to Order-1 regions as Level-zero (L_0) re-

gions, which form the lowest set of regions in the hierarchy. By combining L_0 regions in groups of 4 and selecting exactly one of those regions in each group, we obtain the next set of leader regions, namely Level-One (L_1). A similar procedure of grouping and dividing is done until the hierarchy of k levels can be established. A formal definition of the procedure that creates a multilevel hierarchy from a set of L_0 regions follows:

Grid Ordering: The grid hierarchy is defined by a recursive process as follows: at each level $i(1 \leq i \leq k-1)$, we select the top rightmost L_{i-1} leader to be the i^{th} hierarchical leader of the bottom left L_i grid, top leftmost L_{i-1} leader to be the hierarchical leader of the bottom right L_i grid, bottom rightmost L_{i-1} leader to be the hierarchical leader of the top left L_i grid and bottom leftmost L_{i-1} leader to be the hierarchical leader of the top right L_i grid. The top of the hierarchy, (L_k), is defined by the four L_{k-1} grids.

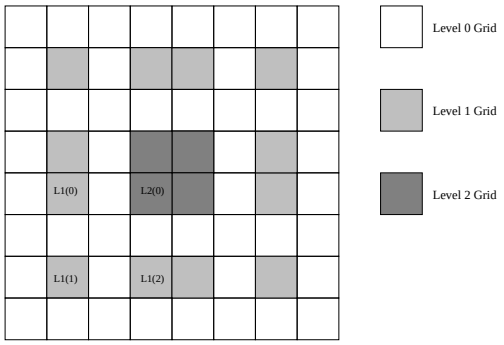


Figure 3.6: A three level hierarchy in HGRID. A level 1 grid leader knows the exact location of all nodes located in the four level 0 grids under it. Level 2 leaders are constituted from level 1 leaders. A local broadcast protocol ensures that all the leaders in level 2 grids are aware of all the nodes in the network.

Example 3.3.1. Figure 3.6 shows a three level hierarchy constructed from a terrain consisting of 64 Order-1 or L_0 regions. By combining $4L_0$ regions at a time, and selecting

one region from each set, we get $16L_1$ leaders, represented by the lightly shaded regions. Note that this selection is not unique across all groups of L_0 regions as per the hierarchy construction algorithm. For e.g., for the lower leftmost L_0 regions, the top rightmost region labeled $L1(1)$ is selected as the L_1 leader for those four regions. On the other hand, the bottom rightmost L_0 region is selected as the L_1 leader for the four L_0 regions located at the top leftmost corner of the terrain. During the next iteration, four L_1 leaders are grouped at a time, and exactly one region is selected from each group to be an L_2 leader, represented by the dark shaded regions. Thus, the unit region labeled $L2(0)$ is a Level-two leader region selected amongst from four L_1 regions, namely $L1(0)$, $L1(1)$, $L1(2)$ and itself. In a nutshell, the hierarchy construction algorithm builds up a hierarchy in the form of a logical tetrahedral pyramid, starting with the unit regions.

We also note that this may not be the only way to form a hierarchy using unit grid (L_0) regions. Other configurations (such as combining 9 unit regions at a time, with the region located in the center delegated as the next level leader) can also be used to build the hierarchy. However, the general idea behind location management and the average location management cost (to be described in Chapter 4) applies for all practical purposes. Additionally, the terrain need not be a perfect square for the protocol to operate correctly. The logical hierarchy construction may be applied by assuming a perfect square topology, but ignore all the unit regions that fall outside the actual terrain dimensions.

Location Management in HGRID: Each time a node *u* crosses an L_0 grid boundary, it broadcasts its entry into the new region, and unicasts two *LOC_UPDATE* (location

update) packets – one to the L_1 grid of its previous L_0 grid (if required) indicating its departure from the region, and another packet to the L_1 grid of its current L_0 grid to indicate its arrival. Each packet contains information regarding the node's current and previous location, as well as the action to be taken (insertion/deletion from the location database) at each leader to make the location servers consistent in their view of the network. The *LOC_UPDATE* packets are processed at each level of the hierarchy in the following manner: node *v* which is present in the hierarchical leader grid to receive the *LOC_UPDATE* first, updates its own location database, and broadcasts the packet in the current grid. Every location server that receives the broadcast simply updates its location database, if it is co-located in the same grid.

Node *v* also checks the *LOC_UPDATE* to see if the boundary crossing requires its hierarchical leader to be alerted, and if so, unicasts the packet to its next destination by geographic forwarding. If the movement specified in the *LOC_UPDATE* is within the area covered by the current hierarchical leader, *v* decides to stop the update process. Thus the location update process continues until the *LOC_UPDATE* reaches either any one of the four L_{k-1} leaders, or a hierarchical leader grid which covers the grid boundary whose crossing started the registration process. When a *LOC_UPDATE* reaches a L_{k-1} leader, the node receiving the packet first carries out a local broadcast protocol to make all the L_{k-1} leader databases consistent. Additionally, node *u* carries out a location maintenance process similar to the previous approaches to update its location database to be consistent with others servers in its new unit region.

Lemma 3.3.2. *A LOC_UPDATE which was produced by an L_i^{th} boundary crossing by a node visits at most i+1 hierarchical leaders for ($0 \leq i \leq k-1$) and i leaders for i = k in any L_{k-1} grid.*

Proof. Since by construction, L_{i+1} leaders are constituted from L_i leaders, any boundary crossing in an L_{i+1} grid requires only the leaders from the lowermost level to the $i+1^{th}$ leader to be notified. Since there are at most k hierarchical leaders to be visited from the lowest level to the highest, an L_k boundary crossing visits at most k leaders. □

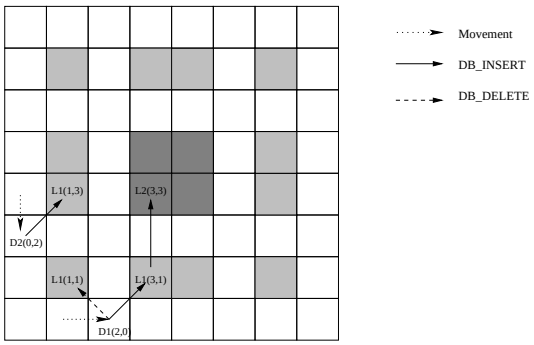


Figure 3.7: Location Update in HGRID

Example 3.3.3. Figure 3.7 shows a typical location update process in the hierarchical location management scheme. Since D_1 crosses a *level-1* boundary, it has to unicast two *LOC_UPDATE* packets, one to L_1(1,1), the L_1 leader of the previous L_0 grid it had visited, and the other to L_1(3,1), the L_1 leader of it's current grid. The update to L_1(1,1) instructs location servers within the grid to delete the entry for D_1 from their location databases (*DB_DELETE*), and the update to L_1(3,1) instructs the servers in L_1(3,1) to make a new entry for D_1 (*DB_INSERT*). The update to L_1(1,1) stops at L_1(1,1), while the update to L_1(3,1) continues to be unicast to L_2(3,3), since a *level-1* boundary crossing

requires the next hierarchical leader L_2(3,3) to be updated. Notice that the movement of another node D_2 terminates at L_1(1,3), since the movement is local to the L_1 hierarchy managed by L_1(1,3). Note that each step shown in the figure may consist of multiple physical transmissions of the update packet to reach the specified grid.

Location Discovery: If destination D is in the same unit region as source S, D would be in S's neighbor table because of the local broadcast protocol. Otherwise, a *LOC_QUERY* (location query) packet for D is sent to S's L_1 leader. As part of location server set up, it is trivial to realize that if S and D are in the same L_i^{th} grid, the query has to be forwarded until it reaches an L_i^{th} server (in the worst case), before a location reply can sent back. Since the location databases in the upper levels of the hierarchy carry the approximate location information of nodes, location replies from these servers return the address of the server who has more accurate information of the destination.

Lemma 3.3.4. *A LOC_QUERY visits at most k-1 hierarchical leaders.*

Proof. Since the query has to be forwarded to a L_{k-1} leader in the worst case, and since the serves in the top level have complete knowledge about the network, the lemma follows. □

Example 3.3.5. In figure 3.8, the location query from source S is forwarded by L_1(1,3) to L_2(3,3), since it has no knowledge of destination D. The query terminates at L_2(3,3), and a location reply is returned to S with the destination location specified as L_1(3,1) in the reply packet.

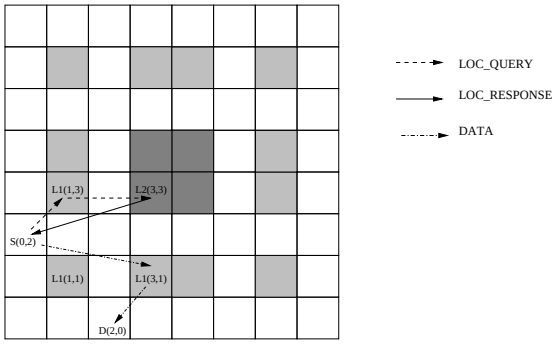


Figure 3.8: Location Discovery and Data Transfer

Data Transfer: If S and D are co–located in the same L_1 grid, the location of D, as indicated by the location reply, is accurate and S can forward the data directly to D's location. Otherwise, the location is approximate, and S forwards the data packet to the location server specified in the location reply. When the data packet reaches the specified grid, a server v that receives the packet first checks its neighbor table to see if D is co-located in the same grid. If so, the packet is successfully forwarded to the destination. Otherwise, v searches for the D in its location database. By construction, v must have an entry for D in its location database. If v has accurate information about D, it further forwards the packet to D, otherwise, the packet is forwarded to the next location server which is a level lower than v in hierarchy, but has more accurate information about D's position. This process continues until the packet reaches D, or it reaches an L_0 grid, and the node that receives the packet drops it since it has no information about D. This can happen because D would have left this grid, and D's location update to its new hierarchical leader failed to reach the leader before the data packet was forwarded by the leader to D's previously visited grid. Since packet transmission time is much

smaller than node mobility, such drops should not be frequent under low network load conditions.

Example 3.3.6. In figure 3.8, *S* forwards the data packet to L_1(3,1), as indicated by the location response from the location server. When the packet reaches L_1(3,1), the node that receives the packet searches its location database, and realizes that *D* is located in (2, 0). It then forwards the packet to (2, 0) which is successfully received by *D*.

Thus, under the HGRID protocol, routing may sub-optimal due to the fact that (i) the location servers have partial information regarding the nodes in the network, and (ii) location replies may carry approximate node locations. The first data packet to arrive at the destination may suffer by taking a much longer route than the shortest path to the destination, had the precise location of the destination been known. To alleviate this problem, when *D* receives a packet from *S*, it can send a notification to *S* with its current location so that the new packets to *D* do not suffer the same fate as their predecessor. For TCP connections, this can easily be built into the acknowledgement packet that *D* proceeds to send to *S*.

3.4 Summary

In this Chapter, we have described three novel location management protocols based on the division of the terrain into unit regions and combining them to in a manner unique to each algorithm to form location server regions. The motivation to do so is derived from the observation that node aggregation based on the terrain and delegation of location

server responsibility to unit regions is more stable than one based on individual nodes and their locations. Once the server regions are defined, the location management protocol consists of updates, maintenance, querying and data delivery. The server region selection in SLALoM is under the assumption that nodes move randomly in the terrain and that communication requirements arise between source-destination pairs randomly located in the terrain. Due to the potential volume of signalling updates due to Order-2 region crossing in SLALoM, an optimization scheme for controlling the number of location updates is proposed, and is known as ELF. Finally, a multilevel hierarchical partitioning of the terrain is suggested in HGRID to account for more realistic scenarios, where node mobility and communication needs are localized. We will show that the design choice for partitioning the terrain has an impact on the cost of location management, defined as the control overhead required to distributively manage and discover the locations of nodes, on all the proposed protocols in Chapter 4.

Chapter 4

Scalability Analysis

In this section, we analyze the scalability of the three protocols described in Chapter 3 with respect to increasing network size. We use the notion of scalability from [87], where the scalability of a protocol is defined as the ability of a protocol to support the continuous increase of its network parameters without degrading the the network performance. For clarity, we briefly describe the main contribution of [87]:

- The *minimum traffic load* of a network is the minimum amount of bandwidth required to forward packets over the shortest paths available, assuming all the nodes have a priori full topology information

- Let $Tr(\lambda_1, \lambda_2, ...)$ be the minimum traffic load experienced by a network under parameters λ_1, λ_2 etc., then the network scalability factor Ψ_{λ_i} with respect to parameter λ_i is defined to be

$$\Psi_{\lambda_i} = \lim_{\lambda_i \to \infty} \frac{\log Tr(\lambda_1, \lambda_2, ...)}{\log \lambda_i}$$

- Let $X_{ov}(\lambda_1, \lambda_2, ...)$ be the total overhead due to protocol X under parameters λ_1, λ_2 etc., then the protocol scalability factor $\rho^X_{\lambda_i}$ of protocol X with respect to parameter λ_i is defined to be

$$\rho^X_{\lambda_i} = \lim_{\lambda_i \to \infty} \frac{\log X_{ov}(\lambda_1, \lambda_2, ...)}{\log \lambda_i}$$

- A protocol X is said to be scalable with respect to parameter λ_i if and only if, as λ_i increases, the total overhead induced by such protocol does not increase faster than the network's minimum traffic load. i.e,

$$\rho^X_{\lambda_i} \leq \Psi_{\lambda_i}$$

Since the per node degree and the node density per unit area remains constant with network size for the class of networks that we consider, we can readily compute the minimum traffic load of the network. If there are N nodes in the network, and each node generates λ_t bits per second and the average path length increases as $\sqrt{N}$ hops, then $Tr(\lambda_t, N) = \theta(\lambda_t N^{1.5})$. This expression assumes that there is an ideal location management protocol from which the source obtains the destination node's location, as well as a perfect geographic routing algorithm that routes the packets in the best possible manner, given the locations of intermediate nodes. By using the definition of network scalability factor with respect to network size N from above, we get $\Psi_N = 1.5$.

4.1 Framework for Location Management

Cost Analysis

The overhead cost of a location management protocol can be divided into three parts:

• *Location update cost (LU)*: This cost covers all the messages nodes send to their home regions whenever they move to a new location.

• *Location maintenance cost (LM)*: This cost covers all the messages nodes (a) send to their previous Order-1 squares to inform them of their departure, (b) send to their current Order-1 squares to inform them of their arrival and (c) collect location information as they become location servers for the nodes currently registered in the new Order-1 square.

• *Location discovery cost(LD)*: This cost covers all the messages sent for locating a mobile.

Hence, ρ_N^Xshould be ≤ 1.5 for each overhead induced by each routing protocol in order to be deemed scalable with network size as per the above discussion.

Mobility Model: In our mobility model, we assume that nodes move randomly and that the movement is independent of each other. Each node selects a direction to move, chosen uniformly between $[0, 2\pi]$. Each node also selects its speed, chosen uniformly between $[v-c, v+c]$ for some time t, where t is distributed exponentially with mean τ. After a mobile has traveled for time t, it selects another direction, speed, and time to travel. As a consequence of this model, the average degree of a node will be proportional to $\pi r_t^2 N/A$ where πr_t^2 is the area within a node's transmission range. To keep this

fraction constant, A must grow linearly with N.

For the grid-based topologies considered, the original area is partitioned into unit regions. Based on the above mobility model, the size of the unit region is chosen so that its average node density is γ, a constant. Hence, there are $A = N/\gamma$ unit regions, each with area a.

Cost Analysis: [23] were the first to analyze the scalability of a location management protocol in a theoretical framework. Under this framework, the cost of a location management protocol is analyzed by using a specific mobility model and a specific geographic routing algorithm, as the *average number of packets* sent within the network in order to maintain the locations of the nodes. The main observations from [23] are the following:

1. *The cost of broadcasting in an Order-*1 *square by a node is proportional to the number of transmissions needed to cover the said square. The latter is in turn proportional to the area of the Order-*1 *square divided by the area covered by a single transmission. Thus,* $b = O(a/r_t^2)$ *packets per Order-*1 *square.*

2. *The distance a node has to cover to cross an Order-*1 *square is proportional to the side of an Order-*1 *square. Thus, the number of Order-*1 *squares a node crosses per second,* ρ_1*, is proportional to* $v/\sqrt{a}$*.*

3. *Given a source-destination distance d, the number of transmissions that need to be carried out to send a packet from the source to destination is given by* d/z*, where z is the average forward progress made in the course of one transmission.*

Note that from [23], it is known that z can be computed from r_t and the average degree of a node in the network.

4.2 Location Management Cost in SLALoM

Location Update Cost: Recall that each time a node moves into a new order-1 square, it has to inform its 9 nearby home regions of its current exact location. This entails 9 broadcasts in a unit region. Furthermore, if such a move also causes the node to move into a new order-2 square, then it has to inform all its far home regions of its current approximate location. This requires $O(A/K^2)$ broadcasts in a unit region. We have

$$LU = O(\rho_1(9b+m_n)+\rho_2(A/K^2 b+m_f)) \text{ packets/sec/node} \quad (4.1)$$

where m_n and m_f are the costs of sending a message to nodes in the near and far home regions respectively. We can estimate m_n (and m_f) by d_n/z (d_f/z) where d_n is the total distance between the source node and its nearby (distant) servers and z is the *average forward progress* made towards a destination node in the course of one transmission. To determine d_n and d_f, we need to consider the total length of the edges of the tree used to span the home regions of a node. We mentioned that this tree will have total length $O(A/K)$. Furthermore, the tree behaves like a BFS-tree in that nearby home regions are close to the root while very far home regions are at the bottom of the tree. In particular, the 9 nearby home regions of the node are spanned by a subtree of

length $9K$. We now have

$$\begin{aligned} c_u &= O(\rho_1(9b + 9K/z) + \rho_2(Ab/K^2 + A/(Kz))) \\ &= O(vK + vN/K^2) \text{ packets/sec/node.} \end{aligned} \tag{4.2}$$

Location maintenance Cost: When a node moves to a new order-1 square, *LM* covers the cost of two broadcasts: the first to inform the node's previous order-1 square of its departure and the second to inform its new order-1 square of its arrival. It also includes the cost of collecting server information, which will be proportional to the number of nodes registered in an order-1 square. Since each node's location is recorded in $O(A/K^2)$ home regions, there are $O(AN/K^2)$ location information to store. But there are $G = O(N)$ order-1 squares so each order-1 square has $O(A/K^2)$ nodes registered at its site. If we assume that one data packet can contain β bits then,

$$\begin{aligned} c_m &= O(\rho_1(2b + A/(K^2\beta))) \\ &= O(v + vN/K^2) \text{ packets/sec/node.} \end{aligned} \tag{4.3}$$

Location Discovery Cost: Let c_l denote the cost of locating a node per second per node. If u wishes to find the location of a v, it sends a unicast to a home region of v closest to it. By construction, such a home region is at most $O(K)$ away. If this home region is near v then u obtains the exact location of v. Thus, $c_l = O(K/z)$ packets per second per node. On the other hand, if the home region is far from v then u obtains an approximate location of v. Node u then routes its message to a home region near v, R_k. The node that receives the message at R_k then sends it to the exact location of v. In this

case,

$$\begin{aligned} c_l &= O(K/z) + O((d(u,R_k) + d(R_k,v) - d(u,v))/z) \\ &= O(K) + O(d(R_k,v)/z) \\ &= O(K) + O(K) \text{ packets/sec/node.} \end{aligned} \tag{4.4}$$

The second term of the first equation arises because we added the extra steps the routing takes before reaching v. The second equation follows from the fact that $d(u,R_k) \leq d(u,v)$. And since the distance of v to any of its nearby home region is at most $2K$, the third equation follows. In this analysis, we shall make the assumption that packets arrive at each node at a rate of λ packets/sec according to a Poisson process.

Total Overhead Cost: Combining the results above, we have the total overhead cost for the entire network.

Theorem 4.2.1. *The average total overhead cost for location management in SLALoM is $O(vKN + vN^2/K^2)$ packets per second, which is minimized when $K = \theta(N^{1/3})$. That is, when K is chosen appropriately, the average total overhead cost of our protocol is $O(vN^{4/3})$ packets per second.*

Proof. The proof follows from minimizing the total overhead cost with respect to K. □

Theorem 4.2.2. *SLALoM is asymptotically scalable with respect to network size.*

Proof. From the previous analysis, $\rho_N^{LU-SLALoM} = 1.33$, $\rho_N^{LM-SLALoM} = 1$ and $\rho_N^{LD-SLALoM} = 1.33$, and the proof follows. □

4.3 Location Management Cost in ELF

Location update cost: By using an argument similar to the one by [23], ρ_2, the number of Order-2 squares a node crosses per second, can also be estimated by $O(\rho_1/K) = O(v/K\sqrt{a})$ Order-1 squares per second, and ρ_3, the rate at which a node updates all home regions is $O(\rho_1/\alpha K^{1+\beta}) = O(v/\alpha K^{1+\beta}\sqrt{a})$ Order-1 squares per second. Recall that each time a node moves into a new Order-1 square, it has to report to its terminal home region its current location. This entails one broadcast in a unit region. Furthermore, if such a move also causes the node to move into a new Order-2 square, and the threshold is less than αK^{β}, then it has to instruct its previous terminal home region to set up a forwarding pointer to the current terminal home region. This again requires another broadcast in a unit region. However, if the threshold is greater than αK^{β}, then all home regions need to be notified about its current approximate location. This requires $O(A/K^2)$ broadcasts in a unit region. Recall that the cost of updating all home regions is proportional to the sum of the length of all edges in the multicast tree spanning all home regions, which is proportional to $O(A/K)$.

$$\begin{aligned} LU &= O(\rho_1(b+K/z)+\rho_2(b+K/z)+ \\ &\qquad \rho_3(A/K^2 b+A/Kz)) \text{ packets/sec/node} \end{aligned}$$

Substituting ρ_1, ρ_2 and ρ_3 we have

$$LU = O(vK+vN/K^{\beta+2}) \text{ packets/sec/node} \tag{4.5}$$

Location maintenance cost: The location maintenance phase consists of two prim-

itives: (a) broadcast on arrival into a new L_0 grid and, (b) receive neighbor/location information to keep as criteria for being a member of the new grid. Hence,

$$\begin{aligned} LM &= \rho_1 (b+\delta) \qquad (1 \leq \delta \leq \gamma) \\ &= O(v) \text{ packets/sec/node} \end{aligned} \tag{4.6}$$

Location discovery cost: If v wishes to discover node u, it sends a location query to a home region of u closest to it. By construction, such a home region is at most $O(K)$ away. If this is a terminal home region, then v obtains the exact location of u. Thus, $c_l = O(K/z)$ packets per second per node. On the other hand, if the home region is forwarding, then v obtains an approximate location of u. Node v then routes its message to the home region of the last known Order-2 region of u, **R** (see Fig. 4.1). The message then follows the forwarding chain set up by u before it is handed to u. In this case, the additional distance the message has to traverse

$$\begin{aligned} d &= x+y-z \\ &\leq 2y \\ &= O(y) \\ &= O(K^{\beta+1}) \end{aligned}$$

Thus,

$$\begin{aligned} LD &= O(K/z) + O(K^{\beta+1}/z) \\ &= O(K^{\beta+1}) \text{ packets/sec/node.} \end{aligned} \tag{4.7}$$

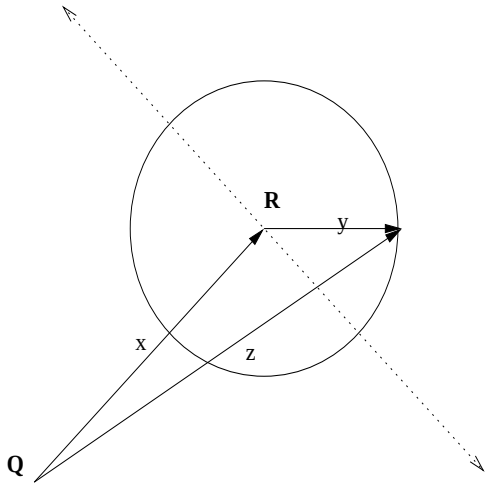


Figure 4.1: Location discovery cost in ELF

In this analysis, we shall make the assumption that packets arrive at each node at a rate of v packets/sec according to a Poisson process.

Total Overhead Cost: Combining the results above, we have the total overhead cost for the entire network. Thus

$$\begin{aligned} c_t &= N(c_u + c_m + c_l) \\ &= O(vNK + vN^2/K^{\beta+2} + vN + vNK^{\beta+1}) \\ &= O(vN^2/K^{\beta+2} + vNK^{\beta+1}) \end{aligned} \tag{4.8}$$

Theorem 4.3.1. *The total overhead cost of ELF protocol is $O(vN^{\theta})$ $(4/3 \leq \theta \leq 3/2)$ packets per second.*

Proof. Minimizing c_t with respect to K, we have

$$K = (\frac{\beta+2}{\beta+1}N)^{\frac{1}{2\beta+3}}$$

and

$$c_t = O(vN^{\frac{3\beta+4}{2\beta+3}}) \tag{4.9}$$

Let θ be $\frac{3\beta+4}{2\beta+3}$. The lower bound for θ results when β is zero and the upper bound results when β is a large number. □

Theorem 4.3.2. *ELF is asymptotically scalable with respect to network size.*

Proof. From the previous analysis, $\rho_N^{LU-ELF} = 1.33 \leq \theta \leq 1.5$, $\rho_N^{LM-ELF} = 1$ and $\rho_N^{LD-ELF} = 1.33 \leq \theta \leq 1.5$, and the proof follows. □

4.4 Location Management Cost in HGRID

We assume the following notations in the analysis:

N - number of nodes

γ - average number of nodes per L_0 grid

k - number of hierarchies = $\frac{log_2 N}{2\gamma}$

d - side of a L_0 grid

P_i - probability of a L_i^{th} server updation

$\bar{d_i}$ - average distance traversed by a *LOC_UPDATE* on an L_i^{th} boundary crossing

z - *average forward progress* made in one transmission

b - cost of broadcasting

ρ - rate at which nodes cross L_0 regions.

Location update cost: Let D be the average distance traversed by *LOC_UPDATE*s

for any boundary crossing and let $\hat{d_i}$ be the total average distance traversed by both *LOC_UPDATE*s on an a L_i^{th} boundary crossing. Since $\hat{d_i}$ can be at most twice the maximum average distance traversed by any one of the *LOC_UPDATE*s on an L_i^{th} boundary crossing, we have:

$$\hat{d_i} \quad \leq 2 \times \bar{d_i} \tag{4.10}$$

Here we take maximum average distance to be the distance traversed by the *LOC_UPDATE* from the current L_0 grid to the i^{th} hierarchy.

For the average distance traversed as a result of boundary crossing in each level of the hierarchy, we have:

$$\begin{aligned} \bar{d_1} &\leq d(\frac{2+\sqrt{(}2)}{4}) \\ \bar{d_2} &\leq \bar{d_1} + 2 \times d(\frac{2+\sqrt{(}2)}{4}) \\ &\leq 3 \times d(\frac{2+\sqrt{(}2)}{4}) \end{aligned}$$

In general, the average distance traversed in the i^{th} hierarchical level is

$$\bar{d_i} \quad \leq 2^{i-1} \times d(\frac{2+\sqrt{(}2)}{4}) \qquad (1 \leq i \leq k-1)$$

$\bar{d_k}$ is a special case, since in the best case, we have to broadcast only once (the L_k^{th} boundary crossing into a topmost leader), and in the worst case, we have to visit *k-1* leaders starting from L_0. Therefore

$$\begin{aligned} \bar{d_k} &\leq d(\frac{1+2+4+...+2^{k-2}}{2}) \\ &\leq d(\frac{2^{k-1}-1}{2}) \\ &\leq 2^{k-1} \times d(\frac{2+\sqrt{(}2)}{4}) \end{aligned}$$

Generalizing, the average distance traversed by a *LOC_UPDATE* in the i^{th} hierarchical level is

$$\bar{d_i} \leq 2^{i-1} \times d(\frac{2+\sqrt{(2)}}{4}) \qquad (1 \leq i \leq k) \tag{4.11}$$

In addition, P_i, the probability that the L_i^{th} server is updated, is:

$$\begin{aligned} P_1 &= \frac{4^k}{2 \times 4^k \times [1-2^{-k}]} \\ &= \frac{1}{2 \times [1-2^{-k}]} \\ P_2 &= \frac{\frac{4^k}{2}}{2 \times 4^k \times [1-2^{-k}]} \\ &= \frac{P_1}{2} \end{aligned}$$

In general, the probability values are

$$P_i = \frac{P_1}{2^{i-1}} \qquad (1 \leq i \leq k) \tag{4.12}$$

Average distance traversed by a *LOC_UPDATE*

$$\begin{aligned} D &= \sum_{i=1}^{k} P_i \hat{d_i} \\ &= 2 \times \sum_{i=1}^{k} P_i \bar{d_i} \\ &\leq 2P_1 \bar{d_1} \times \sum_{i=1}^{k} \frac{2^i - 1}{2^{i-1}} \\ &\leq 2d(\frac{2+\sqrt{(2)}}{4})(\frac{k}{1-2^{-k}} - 1) \\ &= O(k) \ for\ large\ k. \end{aligned} \tag{4.13}$$

Location update cost consists of $\frac{\bar{d_i}}{z}$ unicasts and i broadcasts (by *lemma 3.1*) for a L_{i-1}^{th} boundary crossing. The average number of broadcasts

$$\begin{aligned} \bar{b} &= \sum_{i=1}^{k} iP_i \\ &= 2 - \frac{k}{2^k - 1} \\ &= O(1) \ for \ large \ k \end{aligned} \tag{4.14}$$

Hence, average location update cost per node for the hierarchical grid location management is

$$\begin{aligned} c_u &= \rho \ (\frac{D}{z} + \bar{b}) \\ &= \rho \ (\frac{D}{z} + O(1)) \\ &= \rho \ O(\frac{k}{z}) \\ &= O(\rho \ log_2 N) \\ &= O(v \ log_2 N) \text{ packets/sec/node} \end{aligned} \tag{4.15}$$

Location maintenance cost:The location maintenance phase consists of two primitives: (i) broadcast on arrival into a new L_0 grid and, (ii) receive neighbor/location information to keep as criteria for being a member of the new grid. Hence, average location maintenance cost per node for hierarchical grid location management is

$$\begin{aligned} c_m &= \rho \ (b + \delta) \qquad (1 \leq \delta \leq \gamma) \\ &= O(v) \text{ packets/sec/node} \end{aligned} \tag{4.16}$$

Location discovery cost:By lemma 3.3.4, a *LOC_QUERY* visits at most k hierarchical leaders in any L_{k-1} grid. Since the network consists of four similar L_{k-1} grids, it is enough to find the cost of location discovery in any of the L_{k-1} grids. Also, notice that *LOC_QUERY* packets originating from grids closer to the hierarchical leaders have to traverse lesser distance than the others in the same L_i grid. Denote by $P_{i,j}^{A}$ the probability that node A is in the j^{th} quadrant of the L_i^{th} grid, and by $\bar{d_i}$ the average distance traversed by a *LOC_QUERY* in the L_i^{th} grid. For any source S and destination D, we can get a recursive equation to $\bar{d_k}$, and the average distance traversed by a *LOC_QUERY* in the

L_k^{th} grid can be derived as follows:

$$
\begin{aligned}
\bar{d_k} \;=\; & (P_{k,1}^S \cap P_{k,1}^D) \times \bar{d_{k-1}} + \\
& (P_{k,1}^S \cap \bar{P_{k,1}^D}) \times (\bar{d_{k-1}} + 2^{k-1} d) + \\
& (P_{k,2}^S \cap P_{k,2}^D) \times \bar{d_{k-1}} + \\
& (P_{k,2}^S \cap \bar{P_{k,2}^D}) \times (\bar{d_{k-1}} + 2^{k-1} \sqrt{(}2) d) + \\
& (P_{k,3}^S \cap P_{k,3}^D) \times \bar{d_{k-1}} + \\
& (P_{k,3}^S \cap \bar{P_{k,3}^D}) \times (\bar{d_{k-1}} + 2^{k-1} d) + \\
& \sum_{i=0}^{k-2} ((P_{i+1,1}^S \cap P_{i+1,1}^D) \times \bar{d_i} + \\
& (P_{i+1,1}^S \cap \bar{P_{i+1,1}^D}) \times (\bar{d_i} + 2^i d) + \\
& (P_{i+1,2}^S \cap P_{i+1,2}^D) \times \bar{d_i} + \\
& (P_{i+1,2}^S \cap \bar{P_{i+1,2}^D}) \times (\bar{d_i} + 2^i \sqrt{(}2) d) + \\
& (P_{i+1,3}^S \cap P_{i+1,3}^D) \times \bar{d_i} + \\
& (P_{i+1,3}^S \cap \bar{P_{i+1,3}^D}) \times (\bar{d_i} + 2^i d)) \\
& \text{where} \\
& \bar{d_0} = 0 \\
& P_{i,j}^A = \frac{4^{i-1}}{4^k} \\
& \bar{P_{i,j}^A} = 1 - P_{i,j}^A \& \\
& P_{k+1,j}^A = 1 \qquad (1 \le j \le 4)
\end{aligned}
\tag{4.17}
$$

Solving for $\bar{d_k}$ we get

$$\begin{aligned} \bar{d_k} &= \frac{3}{4^k}(4\bar{d_1} + 4^2\bar{d_2} + ... + 4^{k-1}\bar{d_{k-1}}) + \\ &\quad \frac{24 \times 32^k - 31 \times 4^k + 7}{4^{2k}} \end{aligned} \tag{4.18}$$

The first term of the expression consists of terms which indicate the average distance traversed in any L_i grid, while the second term indicates the extra distance traversed due to the fact that the query could not be contained in an L_i grid, and therefore had to be referred to the next hierarchical leader. It is evident from the expression that the weight of the second term tends to $O(2^k)$ for large k. Hence, the average distance traversed by the *LOC_QUERY* depends mainly on the average distance moved in the lower levels of the hierarchy.

In the worst case, if we approximate $\bar{d_i}$ to be the worst case distance in the L_i^{th} grid, i.e. $2^i\sqrt{(2)}d$, the average distance becomes

$$\begin{aligned} \bar{d_k} &= \frac{3\sqrt{(2)}d}{7}(2^k - \frac{1}{4^k}) + O(2^k) \\ &= O(2^k) \quad for\ large\ k. \end{aligned} \tag{4.19}$$

Hence the average location discovery cost per node is

$$\begin{aligned} c_d &= \rho\ \frac{\bar{d_k}}{z} \\ &= \rho\ O(\frac{2^k}{z}) \\ &= O(v\ \sqrt{(N)})\ packets/sec \end{aligned} \tag{4.20}$$

Theorem 4.4.1. *HGRID is asymptotically scalable with respect to network size.*

Proof. From the previous analysis, $\rho_N^{LU-HGRID} = 1$, $\rho_N^{LM-HGRID} = 1$ and $\rho_N^{LD-HGRID} = 1.5$, and the proof follows. □

4.5 Numerical Results and Discussion

4.5.1 Simulation Environment

We implemented all the the protocols, including SLURP [23] in Glomosim [88] as separate location management layers that operate in conjunction with IP. SLURP is a flat location management protocol described in literature and serves as a standard for comparison against the proposed protocols. We assume constant bit rate traffic, and data from transport is queued in a separate buffer if the location of the destination is unknown. Packet lifetime in the buffer is 4 seconds, and is subsequently dropped if a location query sent out for the packet's destination fails to return the location of the destination within this lifetime. Apart from having a location database, all nodes are also equipped with a "live connections" table, which is updated when a node receives a data packet or a location change notification. Location change notifications are sent out by end points of a connection when their locations change significantly from their previously advertised locations. This table reduces the number of query packets transmitted during a session, since later data packets in a session can use the destination's location entry in the table until the entry times out. The timeout is determined by the average time it takes a node to move out of a unit grid with an average velocity specified by its mobility. A periodic broadcast protocol enables each node to realize its local connec-

tivity, and records it in a neighbor table to assist in geographic routing. MFR [70] was implemented as the geographic routing algorithm. Specific parameters for our simulations will be listed along with the particular study in the following sections.

4.5.2 Performance Results of SLALoM Vs. ELF

In this scenario, we studied the performance of SLALoM and ELF for scalability with increasing network size. The simulation parameters are shown in table 4.1.

Table 4.1: Simulation parameters for SLALoM vs. ELF

Simulation Time	120 sec	Mobility Model	Random Waypoint
Simulation Area	Varying	Maximum Speed	10 m/sec
Unit region size	250m	Minimum Speed	0 m/sec
Number of Nodes	80 - 720	Pause Time	0 sec
Transmission Range	350m	Traffic Type	Random CBR
Transmission Speed	2 Mbps	Number of Connections	1000
MAC Protocol	IEEE 802.11	Data Payload	1024 bytes
Beacon Interval	1 sec	Buffer Size	1000 packets

We fixed the number of Order-1 grids per Order-2 regions to be four, and the length of the forwarding chain to be two($K = 2, \beta = 0, \alpha = 2$) for the simulations. Note that fixing K to 2 brings out the worst case performance for each of the protocols. For each scenario, 1000 Constant Bit Rate (CBR) connections were randomly generated, with each session sending a 1024 bit data payload. A session terminates successfully if the

location discovery phase returns the correct location so that the data sent to the said location successfully reaches the destination before the simulation ends. We used MFR routing for geographic forwarding, where packets are greedily forwarded by intermediate nodes to reach the intended recipient. Each scenario was run for a period of 120 simulation seconds and each data point presented in the plots represents an average of five simulation runs.

Figure 4.2 shows the *average signalling cost* for both ELF and SLALoM. It is clearly visible from the figure that location update cost dominates other costs (discovery and data transfer) in both schemes, but that ELF performs much better than SLALoM. Since the number of full updates are fewer due to the forwarding chain set up, ELF is able to save on location update cost.

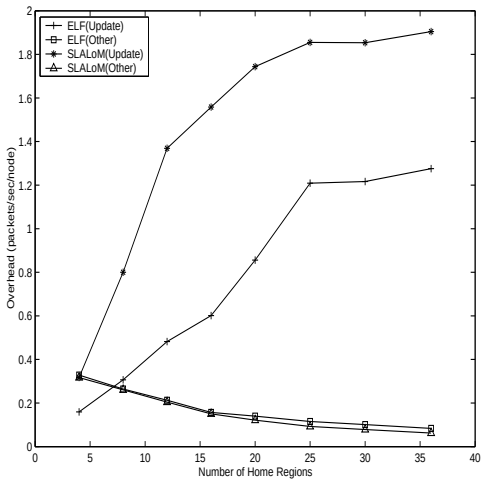


Figure 4.2: Average Signalling Overhead

Figure 4.3 shows CBR *session error probability*. Session error probability is defined as the fraction of successfully received data packets to that of the generated data packets.

Due to the heavy signalling overhead, location discovery packets and data packets suffer heavy queueing under SLALoM, and hence fail to reach the destination. It is quite remarkable that under heavy signalling, the delay experienced by packets is in the order of seconds (see fig. 4.5), indicating that there is a good chance that a location response from the location discovery phase will get dropped, since the originator of the query would have moved out from the grid where it had initiated the query phase. Overall, the number of successfully terminated sessions is much greater in ELF than in SLALoM.

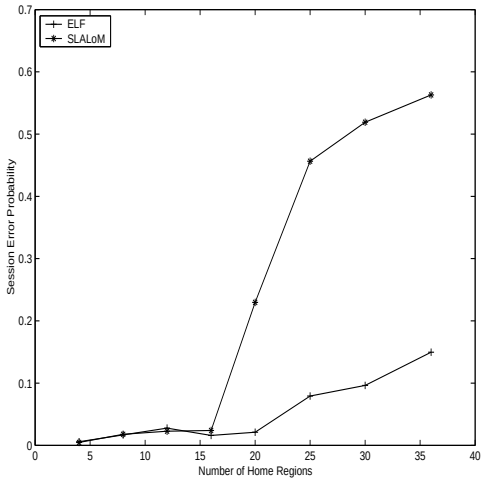


Figure 4.3: Session Error Probability

Figure 4.4 shows the *average path length* (in hops) for data packets. Clearly, the average path length increases for both protocols with increase in terrain size. However, data has to be transmitted over additional hops due to the forwarding chain set up in ELF. On the average, this results in at most one hop more than SLALoM for the scenarios considered, and hence may not be a serious issue in practice.

However, data packets benefit indirectly from location forwarding, as can be seen from

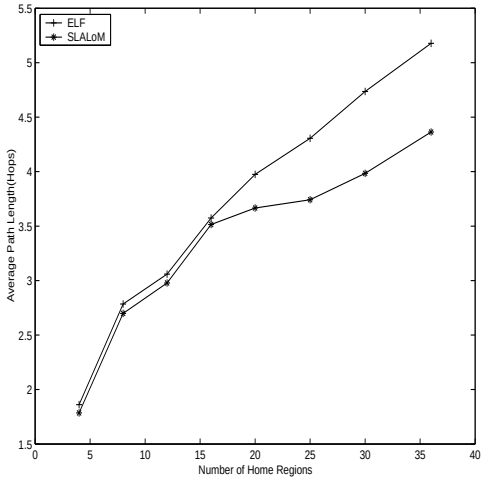


Figure 4.4: Average Path Length for Data

figure 4.5. As the number of nodes increases, so does the volume of update traffic, thereby increasing the network queueing delay at each node for session initiation and successful delivery of data for both protocols. The higher volume of update traffic causes congestion and excessively high packet delays in SLALoM, where as location forwarding results in a much better average delay experienced by both signalling and data packets in ELF. Even with location forwarding, the location discovery phase is carried out faster in ELF since location query/response packets reach the intended recipients with a lower delay. Due to network congestion, data packets suffer high delays in SLALoM, and with the increase in number of nodes, only those packets transmitted between source-destination pairs that are connected by few hops are successfully received (as indicated by the poor throughput and the average path length). Since we took into consideration the delay of successfully received data packets only, the delay of data packets in SLALoM starts to decrease beyond a threshold, indicating a system breakdown. Overall, considering location update/discovery overhead/delay, and data

delivery probability, ELF outperforms SLALoM for all practical purposes.

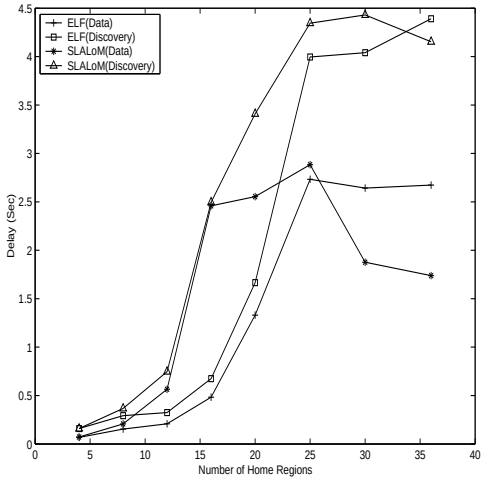


Figure 4.5: Average Packet Delay

4.5.3 Scalability with Mobility

In this scenario, we studied the scalability of the SLALoM, HGRID and SLURP for increase in node mobility, while keeping the network size constant. To test the efficiency of the protocols, we imposed high traffic on the network with 50% of the nodes initiating constant bit rate sessions to other nodes. Each session has a rate of 2 packets/second, randomly starting after 20 seconds into the simulation and terminating randomly at 250 seconds into the simulation. Parameters for the simulations are specified in table 4.2.

Since the performance of SLALoM is dependant on the selection of K, the recommended value of K from [83] is 7 for our scenario. However, that analysis does not consider the optimal node density for geographic routing, and keeping this in mind as well as the tractability of simulations, we decided to simulate two versions of SLALoM, with $K = 2$ and $K = 4$ (hereafter called SLALoM-K_2 and SLALoM-K_4 respectively).

Table 4.2: Simulation parameters for varying mobility scenario

Simulation Time	300 sec	Mobility Model	Random Waypoint
Simulation Area	2000×2000m	Maximum Speed	0-25 m/sec
Unit region size	250m	Minimum Speed	0 m/sec
Number of Nodes	320	Pause Time	0 sec
Transmission Range	350m	Traffic Type	Random CBR
Transmission Speed	54 Mbps	Number of Connections	160
MAC Protocol	IEEE 802.11g	Data Payload	512 bytes
Beacon Interval	1 sec	Buffer Size	1000 packets

For the scenario considered HGRID defines 3 hierarchical levels. The different packet types and location management overhead in bytes for all protocols are given in table 4.3.

Table 4.3: Packet types and overhead

Packet Type	SLURP	SLALoM	HGRID
Update	33	35	34
Query	37	37	37
Response	53	53	53
Notification	33	33	33
Maintenance	33+20n	33+20n	33+20n

The results shown in this section show the effect of mobility on location management and how different location management protocols affect the performance of ge-

ographic routing. We varied the maximum speed in the Random Waypoint model to change the average mobility of the nodes. An increase in mobility is proportional to the rate at which nodes cross grid boundaries, and hence the rate at which new updates are sent out to location servers. We study the effect of this phenomenon on the performance of the network. Each plot point presented here is an average of seven simulation runs.

Figures 4.6 and 4.7 show the average data throughput and delay achieved by each protocol. Throughput decreases with mobility for all the protocols, with SLURP being affected most by mobility, and HGRID performing best. HGRID gives a near steady performance, delivering a throughput above 90% in all the cases. SLALoM-K_2 performs slightly better than SLALoM-K_4. Packet delay increases with mobility, with SLALoM-K_2 performing worst, indicating that network congestion due to mobility causes network under–performance, and that the rate at which updates are sent out affect each protocol with different degree. Since control overhead is least for HGRID (see fig. 4.10), this easily explains why HGRID performs best. Recall that SLALoM

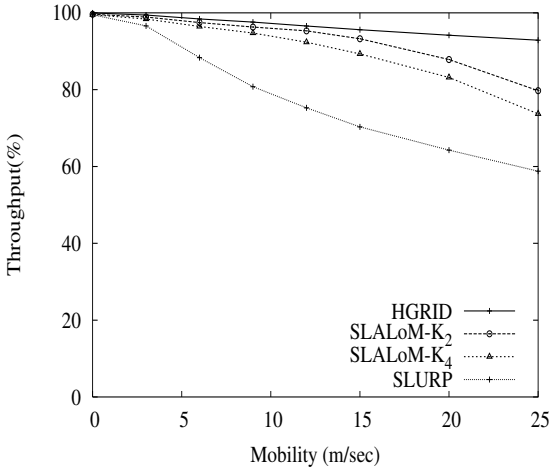


Figure 4.6: Data Throughput

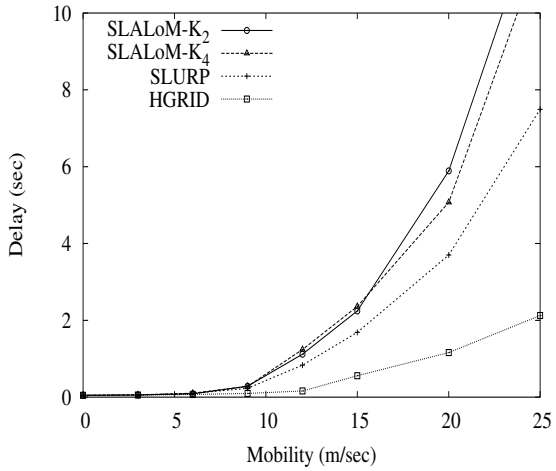


Figure 4.7: Average data delay

groups K^2 Order-1 regions to form Order-2 regions. Thus, in our simulations, each node in SLALoM-K_2 has 16 home regions, both near and far, while nodes in SLALoM-K_4 only have 4 home regions, all being near. Hence, a boundary crossing in results in more updates in SLALoM-K_2 than SLALoM-K_4. On average, both overheads would be more than that of SLURP, since each node has exactly one home region in SLURP (as indicated by figure 4.17). Since higher control overhead adversely affects data on the shared channel, one would expect better performance for SLURP than SLALoM. This can be explained by figures 4.8 and 4.9.

Figure 4.8 shows the probability that a query for a destination returned successfully, and figure 4.9 shows the average delay for location discovery via the query-response phase. Since control overhead is least for HGRID, most queries are not affected by network congestion, and destinations are easily located within milliseconds. However, the high control overhead affects SLALoM-K_2 severely, with a lot of discovery packets being dropped due to MAC layer contention (a quick look at the number of CTS packets

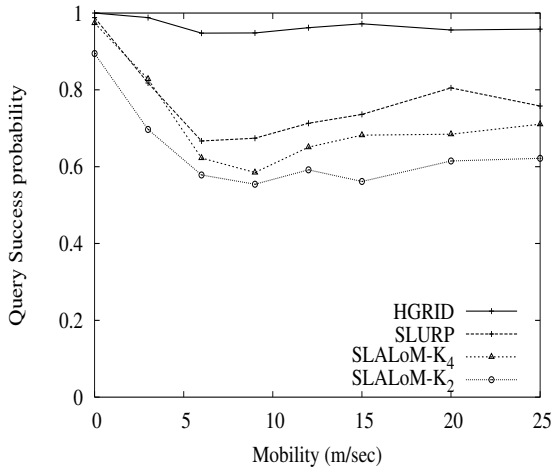


Figure 4.8: Query success probability

ignored by IEEE802.11 due to software carrier sense easily confirmed this). Although the home regions are located closer to the enquirer than that in SLALoM-K_4 or SLURP, the probability that a destination's location is discovered is smaller in SLALoM-K_2. This accounts for the reduced throughput for SLALoM-K_2. However, since the home region of an arbitrary node is located further from an enquirer in SLURP (since home regions are chosen randomly, there is no guarantee that the region is actually close to the enquirer), the location discovery takes longest to complete in SLURP. In the worst case, the query-response phase takes more than 10 seconds in SLURP. This behavior also explains why SLALoM-K_2 performs better than SLALoM-K_4 in terms of location discovery delay. Since data packets have a lifetime of only 4 seconds in the buffer, the delayed responses in SLURP are useless since the packets awaiting the destination's location would already have been discarded at the source. Delayed responses are more costly than no responses at all, as indicated by the poor throughput of SLURP. However, given the same number of hops for a data session, SLURP performs better than

SLALoM in terms of packet delay, since higher congestion due to update traffic causes low priority data to be queued longer in SLALoM.

Figures 4.10 and 4.11 show the control overhead incurred by each protocol in number

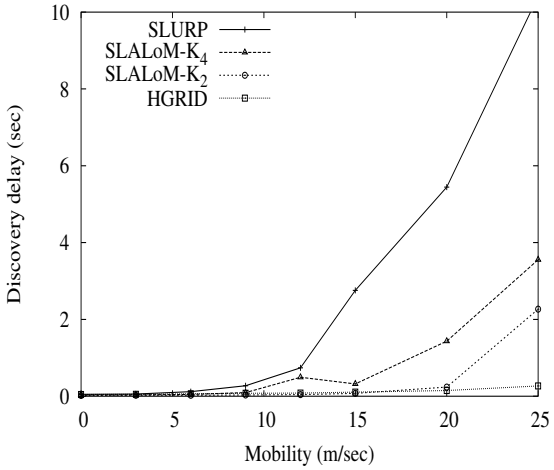


Figure 4.9: Average location discovery delay

of packets. Clearly, the overhead due to update traffic increases with mobility for all protocols, and is highest in SLALoM because a grid crossing results in updating multiple home regions simultaneously. SLALoM-K_4 performs better than SLALoM-K_2 since it only has to update 4 home regions at a time. HGRID performs best, since updates are forwarded to higher level servers only when the movement is across a grid boundary that is not contained within the current hierarchical leader grid. Hence the number of updates are much smaller in HGRID than other protocols. However, this is no longer true if one considers the overhead in bytes than the number of packets per received data packet, as shown by figure 4.12. HGRID location server grids are concentrated towards the center of the terrain, and as the hierarchy of the server grid becomes higher, so does the number of location entries in the location database in a location server which is

present in it. When a node moves into a server grid, it has to receive all the entries of nodes that are registered under that grid as part of database maintenance, and the number of such entries increases the overhead in location maintenance. Server grids become points of congestion and thus the bottleneck in performance as indicated by the increase in data discovery and data delay with increased mobility.

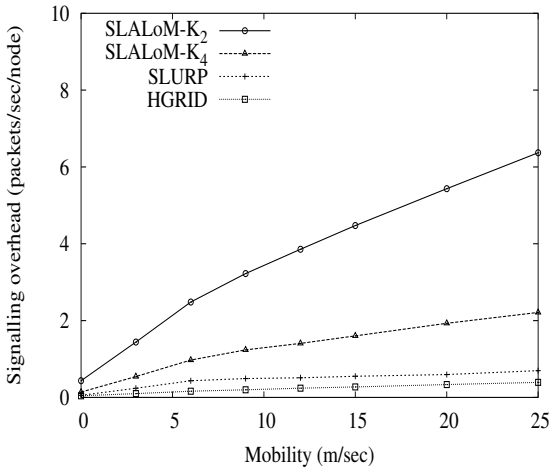


Figure 4.10: Control Overhead (packets/sec/node)

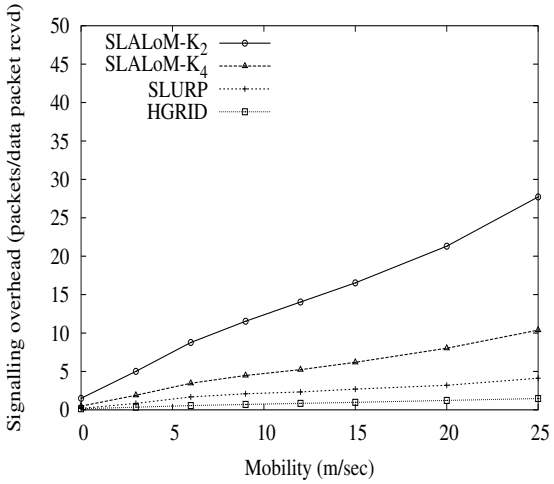


Figure 4.11: Control overhead (packets/received data)

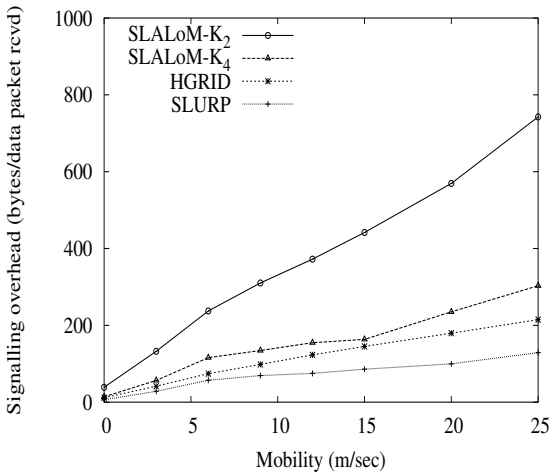


Figure 4.12: Control overhead (bytes/received data)

4.5.4 Scalability with Network Size

In this scenario, we studied the scalability of SLALoM, HGRID and SLURP for increase in network size while keeping the node mobility constant. Specific parameters for our simulations are listed in table 4.4.

Table 4.4: Simulation parameters for varying network size scenario

Simulation Time	300 sec	Mobility Model	Random Waypoint
Simulation Area	varying	Maximum Speed	10 m/sec
Unit region size	250m	Minimum Speed	0 m/sec
Number of Nodes	80 - 2880	Pause Time	0 sec
Node density	80 nodes/km^2	Traffic type	CBR
Transmission Range	350m	Traffic pattern	Random
Transmission Speed	54 Mbps	Number of Connections	1000
MAC Protocol	IEEE 802.11g	Data Payload	512 bytes
Beacon Interval	1 sec	Buffer Size	1000 packets

Since the performance of SLALoM is dependant on the selection of K, we decided to vary K from 2 to 6, such that the total number of home regions per node remained the same for all scenarios . HGRID defines up to 5 hierarchical levels for the scenarios considered. The different packet types and location management overhead in bytes for all protocols are given in table 4.3.

In order to test the network under stable conditions, we let the nodes move around for the first 150 seconds of the simulation, so that location server and database set up is initialized appropriately and the control traffic is stabilized. To test the efficiency of the protocols for location discovery as well as efficient delivery of data, we initialized 1000 CBR connections, where the source and destination nodes are chosen randomly for all the scenarios. Each connection sends one data packet, randomly starts after 150 seconds into the simulation and terminates randomly at 250 seconds into the simulation.

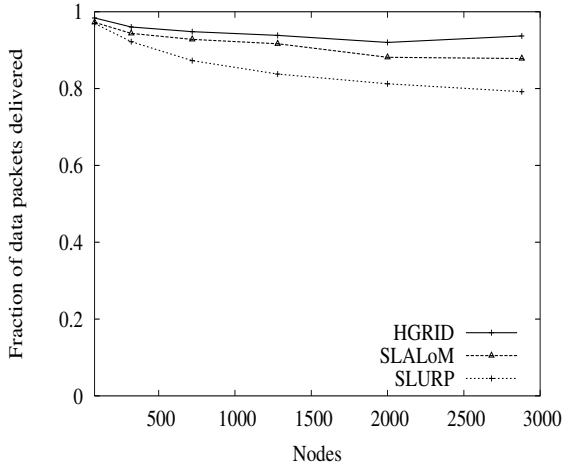


Figure 4.13: Data Throughput

Figure 4.13 shows the fraction of data packets successfully delivered by each protocol. HGRID is able to deliver packets in more than 90% of the cases for all scenarios,

and hence performs best. SLURP performs worst, with significant performance degradation for network sizes of 1500 nodes and more. The under performance of SLURP can attributed to the higher percentage of queries that fail to return the location of the destination and higher delay of location responses. Recall that each data packet is held for only 4 seconds in the buffer, and if a query–response fails to terminate within this period, the packet is dropped. Hence, the delayed responses in SLURP are useless since the packets awaiting the destination's location would already have been discarded at the source.

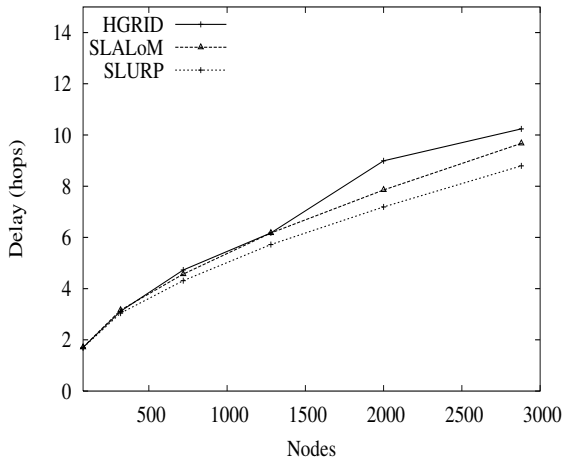


Figure 4.14: Data Delay (hops)

Figure 4.14 shows the delay experienced by the successfully delivered data packets in number of hops (transmissions). Since the location responses convey the exact location of the destination in SLURP, packets take the shortest path in SLURP. HGRID and SLALoM take longer paths, since on average, packets are routed to near home regions in SLALoM or hierarchical location servers having more accurate knowledge of the destination in HGRID, before being handed over to the destinations finally. However,

the average delay experienced by data packets follows a different trend as shown by figure 4.15. Even though packets take the longest paths in HGRID, they have the lowest average delay, since the network is least congested. Since higher control overhead adversely affects data on the shared channel, a lower data delay in SLURP indicates that the network is less congested in SLURP than SLALoM, and is easily verified by figure 4.18.

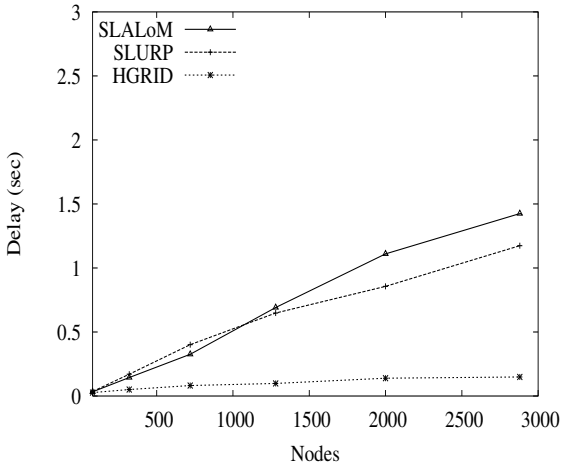


Figure 4.15: Data Delay(secs)

Figures 4.16 and 4.17 show the query success probability and the average delay for location discovery for the three protocols. All protocols perform well, with more than 90% of the queries returning the location of the queried destination. Packets can be dropped due to two reasons in our simulations – *geographical holes* [71] and IEEE802.11 induced congestion. Since the home region of an arbitrary node is located furthest from an enquirer in SLURP (since home regions are chosen randomly, there is no guarantee that the region is actually close to the enquirer) compared to the other protocols, the location discovery takes longest to complete in SLURP. Also, since the home

region is located further, this increases the dropping probability of a query or response packet, and thus decreases the query success probability. SLALoM has the closest home region for any node for the scenarios considered, and has the lowest delay for location discovery. However, high network congestion due to the higher control overhead causes additional packet drops in SLALoM, and thus has lower query success probability than HGRID.

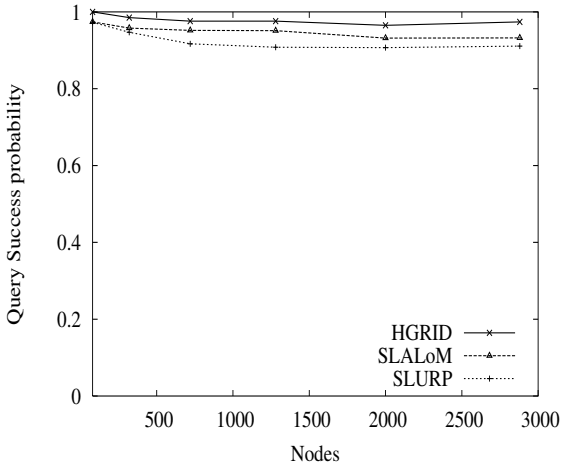


Figure 4.16: Query Success Probability

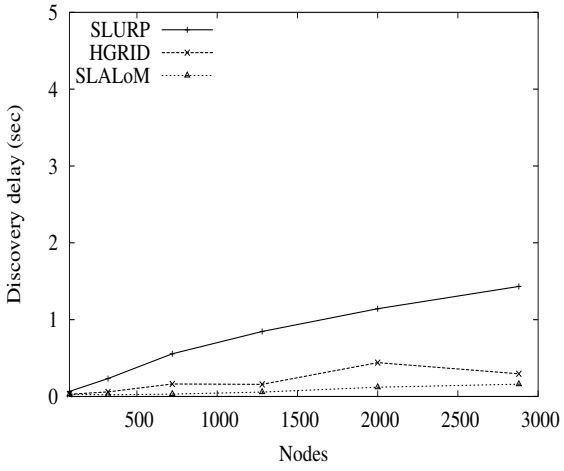


Figure 4.17: Discovery Delay

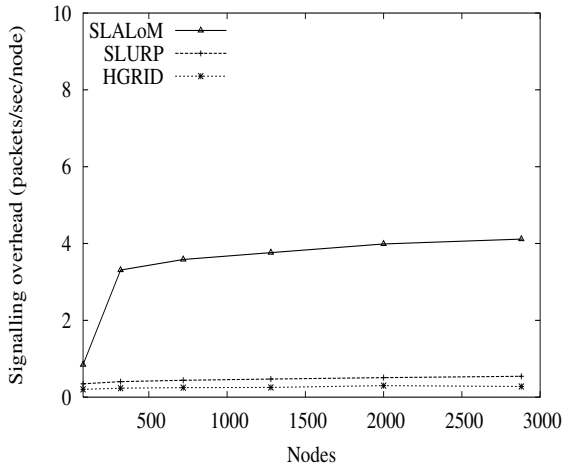


Figure 4.18: Control overhead

Figure 4.18 shows the average control overhead for all the protocols. With the number of CBR sessions being low, most of the signalling overhead is due to update traffic. As indicated by the analysis, HGRID has the least control overhead, since the update overhead increases only logarithmically with the number of nodes. For practical scenarios such as the ones considered in our simulations, SLURP performs better than SLALoM with respect to control overhead. Also, SLURP performs best when one considers the overhead in bytes instead of number of packets. With the packet overhead being nearly equal for all the protocols, the higher overhead for HGRID and SLALoM can be explained by the overhead incurred in location maintenance. The average number of location entries that have to be transmitted is more for SLALoM and HGRID (see figure 4.20), resulting in additional bytes being transmitted in these two protocols. Finally, figure 4.20 shows the increase in location database size of each protocol with increase in number of nodes. SLURP has minimal memory requirements, since each node can choose any unit region randomly from the available set of Order–1 region as

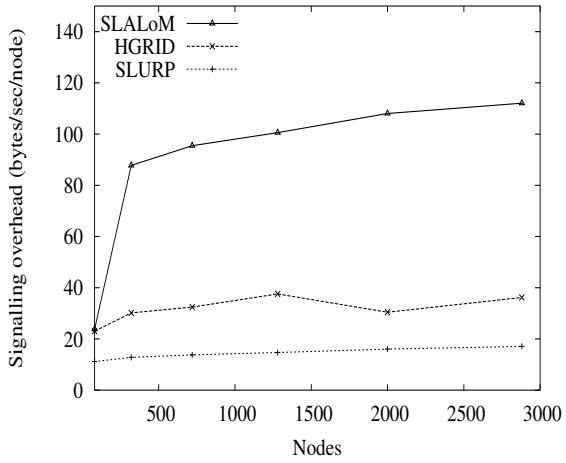


Figure 4.19: Control overhead

its home region. In SLALoM, the set of available Order–1 region is restricted to only K^2. While lower order servers have a few location entries, the higher order servers have to retain location information about almost all nodes in HGRID. However, the average of these grows only slightly with the increase in network size. Thus, all protocols are scalable, with the database size being nearly constant or increasing marginally with network size.

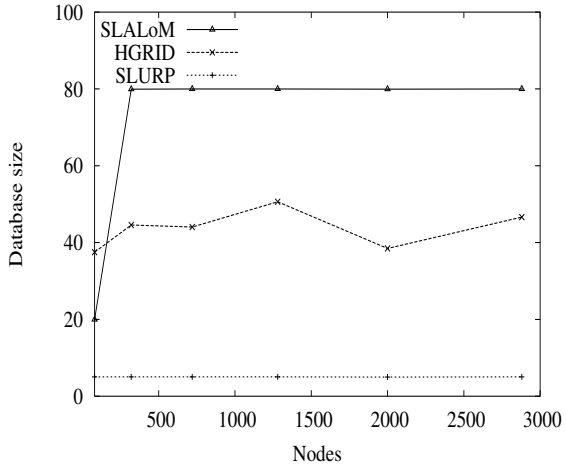


Figure 4.20: Database size

4.6 Summary

In this chapter, we formally analyzed the protocols and the showed that all the three protocols are scalable with respect to network size using a specific framework. In particular, we have shown that the average location management cost increases only as $O(vN^{\frac{4}{3}})$ for SLALoM, and that the location update cost increases only as $O(vN\log N)$ for HGRID for uniformly and randomly distributed ad hoc networks. In scenarios where node mobility and communication are both localized, HGRID can incur an overhead which increases only as logarithmic in the number of nodes. We implemented the protocols in Glomosim, and carried out extensive simulations to study the performance of these protocols for practical scenarios that could not be incorporated into the theoretical framework. Our results indicate that all protocols perform well, and only affect the performance of geographic routing minimally. In particular, the Hierarchical Grid Location Management protocol (HGRID) performs the best for both increase in mobility as well as increase in network size.

Chapter 5

Effect of Node Density on Location Management

As shown in the previous chapter, the proposed protocols were shown to be scalable under dense node conditions, with greedy forwarding being chosen as the position based routing protocol. However, in a sparse network, both the location management scheme as well as greedy forwarding becomes inefficient and can be problematic. More specifically, since most of the proposed location management schemes delegate the location service task to particular unit regions in the topography, it is imperative that there is *at least* one node in the location server region and that this region be accessible to the network. Under low density conditions, unit regions may not contain any nodes. We denote such a condition, in which a location server is devoid of member nodes by the term *empty server region*. Intuitively, the probability that a unit region becomes empty becomes significant with decreasing network density and high node mobility.

Figure 5.1 shows an example of a connected 70 node static random network in a 2000x2000 m region (average density of 1.75×10^{-5} nodes/m^2), in which the unit region is a $250m \times 250m$ square region. The nodes are represented by circles in the figure. At this node density, there are quiet a few empty server regions in the network that are denoted by unit regions that do not contain a circle. Thus, if a node selects any of these empty regions as its location server, then location updates to these regions will simply be dropped at some intermediate node at which the packet cannot be forwarded further. Similarly, any location query (and hence the data) for this node (and any other node that selected this empty server region) will also be dropped since there is no server for the node. Hence, these nodes are temporarily unreachable in the network due to the incorrect operation of the location management scheme even though they are part of the network, and the network throughput suffers.

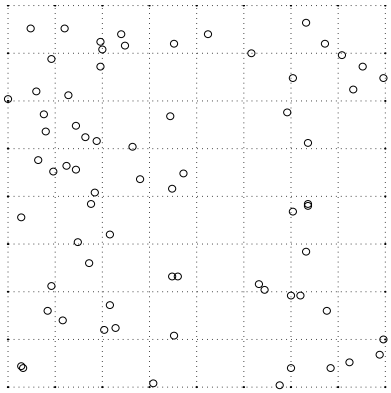


Figure 5.1: Empty server regions in a 70 node network in a 2000x2000 m playground

In addition, greedy forwarding can fail due to the occurrence of local maxima during the forwarding process. Hence, even if a valid path exists to the location server, the packet

will be dropped by the intermediate node at which the local maxima occurs. Note that routing protocols that route along the faces of planar graphs such as Greedy–Face–Greedy (GFG) [22] and the Greedy Perimeter Stateless Routing (GPSR) [71] or depth first search based approach such as the Geographic Routing Algorithm (GRA) [72] can be used to overcome this problem. Thus, the bottleneck in position based routing in sparse networks seems to be the incorrect operation of the location management scheme.

5.1 Characterizing the Empty Server Region Probability

To understand the impact of the problem caused by empty server regions, we find the probability that empty server regions are caused in a uniformly randomly distributed ad hoc network with node density of N nodes per unit region. Assume that each node crosses grid boundaries at the rate of $\frac{1}{\lambda}$ per second, such that the time interval between two grid boundary crossings is represented by an exponential random variable with mean λ. For a system with only two unit regions and assuming a boundless simulation model (nodes that hit the boundary appear on the opposite side of the simulation area), the mobile movement process can be modeled by a one–dimensional Markov chain as shown in Figure 5.2. A state (i, j) in the model represents the number of mobile nodes in each cell, where $i, j \in [0, 2N]$. The steady state probability can easily be found from

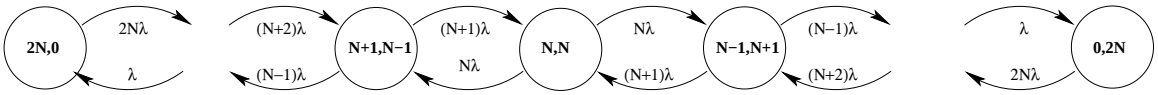
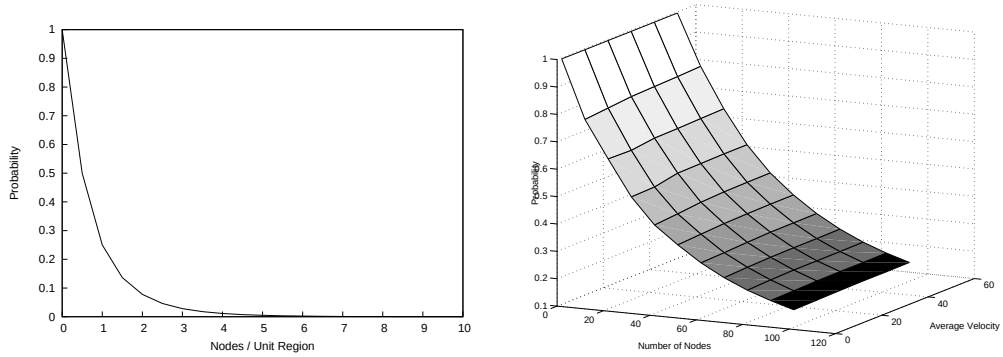


Figure 5.2: State diagram for a two unit region system



(a) Analysis (b) Simulation

Figure 5.3: Empty server region probability.

solving the chain as follows.

$$\begin{aligned}\pi_{2N,0} &= \frac{1}{1+\sum_{i=0}^{2N}\frac{\alpha_0\alpha_1\ldots\alpha_{i-1}}{\mu_1\mu_2\ldots\mu_i}} \\ &= \frac{1}{2(1+2N+\sum_{i=1}^{N}\prod_{j=1}^{i}\frac{2N(2N-j)}{j+1})}\end{aligned}$$

for a finite N and where α and μ represent the outgoing and incoming transition rates in each state in Figure 5.2. The empty server probability is then the sum of probabilities $\pi_{0,2N}$ and $\pi_{2N,0}$. Figure 5.3(a) shows the resulting probability distribution. Since the grid crossing rate of all nodes have been assumed to be the same, it does not have an effect on the steady state probability. We note that a similar analysis can be carried out for systems with more than two unit regions as well as higher order regions [86], but do not elaborate on this in detail.

For a more realistic sampling of the empty server region probability, we ran simu-

lations on a 2000m $\times$ 2000m terrain consisting of unit regions of size 250m $\times$ 250m in which nodes move using the Random Waypoint mobility model [36]. Both the node densities and the average speed of nodes were varied to observe the effect of each on the empty server region probability. Figure 5.3(b) shows the results on the empty server region probability at steady state. It can be seen that low node density has a severe effect on the probability distribution, while increased node velocity has almost no effect on the server regions becoming empty. This is largely due to the fact that although higher velocities cause unit regions to become empty faster, nodes enter empty regions at a similar rate to make them non–empty at steady state. While our analytical model is not directly comparable to the simulations, we observe that the trends in the probability distributions are similar.

5.2 Proxy based Location Management

As one of our main contributions, we for the first time, propose to use a proxy–based location management scheme to address the empty server problem. If a server region R_E is empty, then the duty of that server needs to be delegated to a *proxy* server R_P under the proxy based scheme. Although the concept of proxies is simple, the task of distributed proxy selection and management is non trivial in ad hoc networks due to the inherent tradeoff between complexity and control overhead. Consider the following two schemes: in the first scheme, a node mapped to an empty server region R_E simply selects the first non–empty grid as its proxy server R_P, and a network–wide broadcast

is carried out to make all nodes aware of the new server. All future updates and queries are routed directly to R_P. In the second scheme, after R_P is selected as a proxy for R_E, no network–wide broadcast is done, but instead all location management packets are redirected from R_E to R_P based on a *mapping* between R_E and R_P locally. While the first scheme is conceptually simpler, it incurs significant overhead. On the other hand, the second scheme is resource efficient, but is more complicated. For example, if some time later R_P itself becomes empty, then the protocol has to readjust the mapping without creating *race conditions* that adversely affect the protocol operation.

In the following sections, we outline our proxy based location management scheme. We assume that there is a connected planar graph defined on the non–empty regions of the terrain (referred to as an overlay graph) and a routing protocol that can traverse the graph faces. The routing protocol can detect the presence of empty regions when regular packet forwarding fails, and switches from the regular forwarding mode to the face routing mode to get around empty regions. Later on, we describe centralized and distributed algorithms to construct overlay planar graphs as well as the routing protocol in detail.

We also assume that the empty region is within a single face and that the graph faces remain stable with respect to packet routing time (graph face changes that occur in between the time two packets are routed do not affect the proper routing of packets by the protocol). We note that there is a MAC protocol that can reliably transmit a packet. All protocol packets from a node are uniquely identified by a sequence number. We use the term location server loosely to indicate any node who is a member in an unit region,

since all nodes in an unit region carry out the task of location management.

5.2.1 Proxy Selection

Proxy selection refers to the set of actions that need to be taken when update packets from mobile nodes to an empty server region need to either set up a proxy server (if one has not yet been set up) for that region, or be routed to a proxy server (if a proxy server has already been defined). The proxy selection phase for region R_E is started by an unit region R when regular forwarding of an update packet to the location server region R_E fails in R. It enters this update into a temporary database, indexed by the destination region R_E, in the case that R may have to be the proxy server for R_E later on. R then retransmits the update in face routing mode along *all* its graph edges. A proxy *timer* is also started for this entry whose timeout indicates that the packet either resumed the regular forwarding mode or reached its intended destination (either R_E or a proxy that was already set up for R_E). The value of the timeout is set to the time required to traverse the terrain perimeter. When the timeout occurs, all entries in the temporary database are deleted.

R also snoops into location update packets in face routing mode, and if R notices that it has to forward an update packet to any of the grids in its temporary database in face routing mode, it checks the face route start point (unit region identifier) of the packet, as recorded by the routing protocol. If the closest point of that region to the empty server region is closer than its own, or if the distances are equal but the identifier of the start point is greater than its own, then it stops the timer, deletes the temporary database and

continues the packet's routing. This rule avoids potential race conditions that may occur during proxy selection as indicated by Figure 5.4. If two location updates for empty server region R_E arrive at regions R_{10} and R_8 simultaneously, there is a possibility that we may have *two* proxy servers for R_E. However, R_8 has a lower identifier than R_{10}, and gives up its candidacy to be a proxy for R_E. Packets that loop around the graph

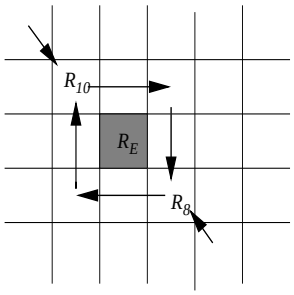


Figure 5.4: Potential race condition during proxy selection

faces are handed up by the routing protocol to the location service layer as described in Section 5.3. If all copies of a previously retransmitted update looped around at R, it means that R_E is indeed empty, and that a proxy has not yet been assigned for R_E. R then assumes itself to be the proxy server for R_E, stops the set up process, and copies all entries for R_E from its temporary database into its proxy table. All future updates to the empty server region R_E are entered into the proxy table of the proxy server R.

Based on the above discussion and the observation that an empty server region R_E lies inside an interior face or outside the exterior face of the grid graph, we have the following theorem and its informal proof.

Theorem 5.2.1. *In a connected network, the selection phase creates exactly one proxy grid R_P for an empty server region R_E and lies in the graph face containing R_E.*

Proof. Face routing guarantees that the retransmitted update packets will visit all edges of the face containing the empty server, and select one of the adjacent grids closest to R_E that lies on the graph face. Moreover, if the update packet started its journey via face routing due to a temporary void, then regular forwarding resumes at an intermediate node that continues to route the packet to either the server region, or to the face containing the empty server region. Finally, race conditions that may create multiple proxies for R_E are avoided by either the distance rule or the unit region identifier rule. □

We note that the proxy selection phase is similar to the Perimeter Refresh Protocol to select a *home node* in [89]. However, the selection is done using face routing on a planar graph formed from the unit disk graph radio network and does not address race conditions that may arise due to simultaneous selection packets.

5.2.2 Proxy Delegation and Maintenance

Proxy delegation is required when a node is about to leave a proxy server region R_P empty and move into a new region R_x or if such a node is about to be shut down (either by an user or due to low battery power of the device). If a location or proxy server becomes empty due to mobility, then a new proxy server has to be selected which is closest to the original empty region R_E and which lies within the face containing R_E. To this end, the last node to move out of R_P (when two or more nodes move out simultaneously, the tie can be broken by using node addresses) starts the proxy delegation phase by transmitting a *delegation* message in face route mode with R_E as the destination and R_x as the grid where face route mode was initiated. If the message loops at R_x, then R_x

is delegated as the new proxy server for R_E and the message is discarded. On the other hand, if R_x notices that the message returned with a closer grid $R_{x'}$ to R_E than itself, it forwards all the entries for R_E to $R_{x'}$, and delegates $R_{x'}$ to be the new proxy server for R_E. In case that the proxy delegation phase was initialized due to low battery power, the process of server selection is carried out by excluding R_P as a potential candidate.

Similarly, proxy maintenance is required when a region R_x, which was hitherto empty, now contains a node u which just moved into it from an adjacent grid. If R_x was a server sometime before, then u must claim all location entries for which R_x was a server prior to it being empty. Similarly, if R_x partitions/closes a previously close/open face in the planar graph, then u must claim all entries for an empty server region along the graph face which R_x just closed, from its proxy. This can be accomplished as follows. Similar to the proxy selection phase, node u starts the proxy timer and transmits a proxy *maintenance* message along all its graph edges in face route mode. All proxies promiscuously listen to maintenance messages, and a server responds to it with a proxy *response* if this grid is currently a proxy for R_x or if R_x is closer to any of the grids that it is currently a proxy for, than itself. Moreover, a node in R_x caches any update, query or maintenance message that it forwards in face routing mode if it is currently seeking a proxy response, indicated by an active timer. In the event that a proxy response is received, the grid is updated with the new location entries, and all previously cached messages are responded to, if possible.

It is worth noting that the proxy delegation and maintenance phases preserve the face containing the empty server R_E such that its proxy R_P is also along the same face.

5.2.3 Location Discovery

A node y that wishes to find a destination node x's current location sends a location query to x's known server R_x, by using the unique mapping between x and R_x. The query is then routed until it reaches either R_x, or its proxy R_E. The server that receives the query responds with a location response with node x's current location. We have the following theorem and its informal proof to establish the correct operation of the proxy location management scheme.

Theorem 5.2.2. *Assuming a reliable MAC protocol, and a routing protocol that can visit all unit regions along the face of a planar overlay graph, the location query is guaranteed to return the last known location of a queried node in a connected network.*

Proof. Since all phases of the proxy management result in the proxy server lying along the face containing the empty server, the query will be responded to as long as the query packet reaches the face under consideration. Similar to an update packet, a query packet is forwarded until it reaches the graph face containing the location server (and hence its proxy). Thereafter, face routing guarantees that the packet visits the appropriate server who can respond to the query. A race condition could have occurred if a query had passed through a region which was in the process of executing the maintenance phase. However, since any region which is currently waiting for a maintenance response caches every packet that passes through it, the query will be answered to eventually. □

5.3 Choice of Routing Algorithms

So far, we discussed the issue of empty server regions, and a localized proxy based mechanism to overcome this issue in sparse networks. However, we note that the proxy scheme is highly dependant on the routing protocol and requires a lot of support from it to operate efficiently and correctly. Particularly, the routing scheme needs to

- Detect empty regions and alert the location management layer upon such detection. However, the void may be a temporary one on the path to the packet's destination, in which case the protocol must intelligently bypass it.
- If a server region is empty, its proxy may be located anywhere along the graph face containing that region, and the routing scheme must guarantee that the proxy be located as fast as possible.
- In the case that a server region is disconnected from the network due to node mobility, packets to such regions should not loop forever, wasting network resources.

While greedy forwarding is not suitable for the operations that we desire, face routing on planar graphs seems attractive for the proposed location management scheme. Since the planar graph due to either scheme spans the non–empty grids in the terrain, face routing has the potential to route around *voids* to locate a proxy. However, even when the empty server region lies inside an interior face of the graph, face routing may operate incorrectly for locating the proxy, as shown by Figure 5.5. The reason why face routing fails in this example is due to the fact that none of the faces containing the terminating node (node u) contain a node which is a member of the proxy region.

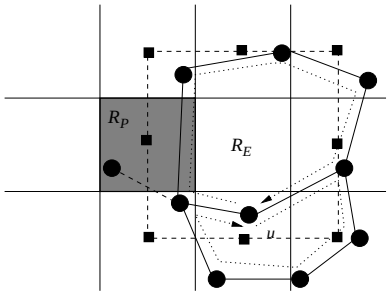


Figure 5.5: An example where face routing fails to detect the proxy server for a query from node u. Node u is where face routing started, and R_P is the proxy server for the empty server region R_E. Alternately, face routing on the subgraph extracted from the overlay graph (indicated by the square vertices and dashed edges) will detect the proxy region.

Another critique of face routing as suggested by either GFG or GPSR is that the quality of routing is inherently tied up with the nature of the planar graph. If we know that the current region is *not* a proxy, then we would like to move to the adjacent choice as fast as possible. Since routing is done on a edge–to–edge basis of the planar graph, proxy management can be slower using either of the schemes.

5.4 Connected Overlay Planar Graph Construction Problem

Considering the drawback of face routing on planar graph extracted from the unit disk graph, we realize that the proxy server management can be carried out correctly only if the planar graph considers the non–empty regions of the network. To this end, we define an *overlay* graph on the unit disk graph and consider the problem of construct-

ing a connected planar graph from the overlay graph. The following subsections outline an overlay graph, the Connected Overlay Planar Construction problem and centralized/distributed solutions for this problem.

5.4.1 Problem definition

Given a set of mobile nodes N and their Euclidean co-ordinates, the network terrain is tiled into an uniform infinite grid such that the unit square regions within the grid are of diameter r, where r is the radio range of a node. The center point (x_i, y_i) of each unit region R_i uniquely represents the unit region and forms a vertex in the overlay graph. A total ordering on the vertices are defined as follows:

- $(x_1, y_1) < (x_2, y_2)$ if $x_1 < x_2$ or $x_1 = x_2$ and $y_1 < y_2$
- $R_i < R_j$ if $(x_i, y_i) < (x_j, y_j)$

A mobile node $u \in N$ can now be uniquely assigned to one of the vertices by defining a total ordering on the node locations on the grid using the following three rules:

- A node $u(x_u, y_u)$ is assigned to R_i if $(x_u, y_u) \in R_i$
- A node $u(x_u, y_u)$ is assigned to $\min(R_i, R_j)$ if $(x_u, y_u) \in R_i, R_j$
- A node $u(x_u, y_u)$ is assigned to $\min(R_i, R_j, R_k, R_l)$ if $(x_u, y_u) \in R_i, R_j, R_k, R_l$

The last two rules assign nodes to a unique vertex if they fall on the boundary of two unit regions or the corner of four unit regions.

Once the vertices are defined, the adjacency matrix of the overlay graph $G(V,E), R_i \in V$ is constructed as follows: two vertices R_i and R_j are adjacent, and a bidirectional edge $e_{R_i \to R_j} \in E$ exists between them if there is a node pair u,v ($u,v \in N, u \in R_i$, $v \in R_j$) that are directly connected to each other in the UDG. Due to this construction, a vertex R_i can be adjacent to at most 20 other vertices as shown in figure 5.6.

To facilitate the explanation of our algorithm and the rest of this paper, we denote by $E_s^{R_i}$ the set of eight short edges (shown in bold solid lines in Figure 5.6) connecting the eight immediate vertices around R_i, and by $E_l^{R_i}$ the twelve remaining edges incident from R_i (shown in dashed lines in Figure 5.6). An edge is also classified as either an *axis*-edge or a *non-axis*-edge depending on whether it is parallel to the $X-Y$ (horizontal/vertical) axes or not. Denote by N_i the neighborhood of R_i, which consists of the 20 vertices that R_i may be adjacent to. With the overlay graph in place, the connected

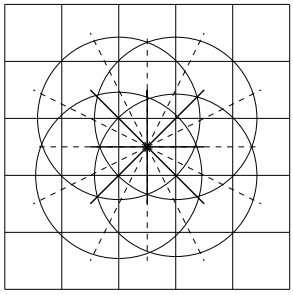


Figure 5.6: A vertex in the overlay graph is adjacent to at most 20 other vertices. Short edges are shown by the bold solid lines while dashed lines indicate long edges. While axis edges are parallel to the grid , non–axis edges are inclined to the grid.

overlay planar graph construction problem is defined as follows: construct a subgraph $G_P(V,E_P), E_P \subseteq E$ which is both planar and connected.

In [90], Frey et. al. studied a similar problem in which the plane is tiled into a mesh of hexagons and each hexagon is referred to as a geographic cluster. Nodes are aggregated into geographic clusters based on their position in the plane, and an aggregate graph is constructed from the radio connectivity between adjacent hexagons. A planar subgraph embedded in the aggregate graph is extracted by a modified Gabriel graph [91] construction using one-hop information advertised by each node in a beacon packet. Routing is then performed on the planar subgraph extracted from the aggregate graph.

Lemma 5.4.1. *The Gabriel graph construction proposed in [90] applied to the overlay graph will always produce a planar subgraph, but the resulting subgraph may be disconnected even if the underlying unit disk graph is connected.*

Proof. The proof follows directly from [90]. As an example, consider the node placements in figure 5.7. Nodes v_1, v_2, v_3 are assigned to vertices R_j, R_i, R_k respectively. Since v_1 and v_3 are disconnected, there are two bidirectional edges $e_{R_i \to R_j}$ and $e_{R_i \to R_k}$ in the overlay graph, indicated by the dashed lines. In the Gabriel graph construction for a node pair (u, v), there must be no other node located in the disk with diameter $|uv|$ than u and v. Thus, to preserve planarity in the overlay graph consisting of vertices R_i, R_j and R_k, edge $e_{R_i \to R_j}$ will be discarded, thereby leaving the subgraph disconnected. □

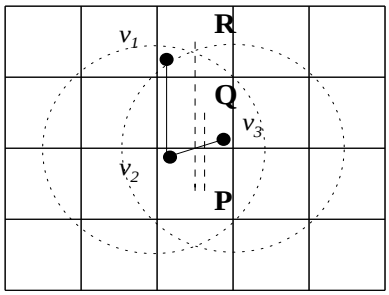


Figure 5.7: The circles indicate the radio range of nodes v_2 and v_3. Dots indicate nodes, solid lines indicate edges in the unit disk graph, solid squares represent the graph vertices and the dashed lines indicate edges in the overlay graph.

5.4.2 Centralized Algorithm

We now describe a centralized algorithm that constructs a subgraph from the overlay graph, which is both planar and connected. The algorithm is based on a specific property of overlay graphs that we refer to as the *redundancy property*. A formal definition of the property is as follows:

Lemma 5.4.2. Redundancy Property*: If two edges in the overlay graph intersect, one of the following must occur: (a) two radio edges intersect in the UDG (b) the two radio edges in the UDG do not intersect, but the location of the four mobile nodes is such that the overlay edge intersection occurs. In either case, there must be at least one node which is directly connected to the remaining three nodes in the UDG.*

Proof. We prove the lemma for the short edges first, and extend the proof to the long edges. We make use of a known result for unit disk graphs at first.

Lemma 5.4.3. *If two radio edges in the UDG intersect, there must be at least one node which is directly connected to the remaining three nodes.*

Proof. The proof follows directly from [92]. □

Since two short axis edges cannot intersect to cause non–planarity, the edges that cross each other must be non–axis edges. There are two cases under which the edges intersect: either the underlying edges in the UDG intersect or they do not. In the former case, from Lemma 5.4.3, we know that there must be at least one node which is directly connected to the remaining three nodes. In the latter case, assume without loss of generality that edge xy is on the right of edge uv as in Figure 5.8, for two edges xy, uv in the UDG. For the edges not to intersect, u, x and y must be located in the shaded regions. Assume that $|xy| \leq r$ and that $|ux|, |uy| > r$. However, this is a contradiction, since $\angle xuy > \frac{\pi}{2}$ and this implies that both x and y are directly connected to u. Thus, lemma 5.4.2 follows for short edges.

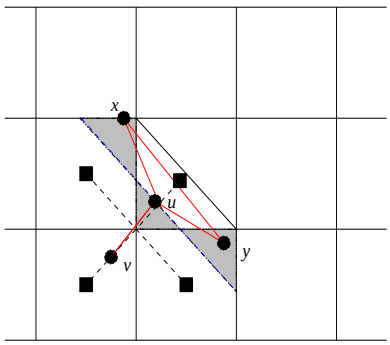


Figure 5.8: Proof of Lemma 5.4.2

For the long edges, Figure 5.9 shows all base cases where edges in the overlay graph may intersect to cause non–planarity. All other possible intersections are symmetrical to those shown in the base cases. For the cases shown in Figure 5.9(a),5.9(b) and 5.9(c), if the non–planarity is caused by intersecting edges in the UDG, then from Lemma 5.4.3

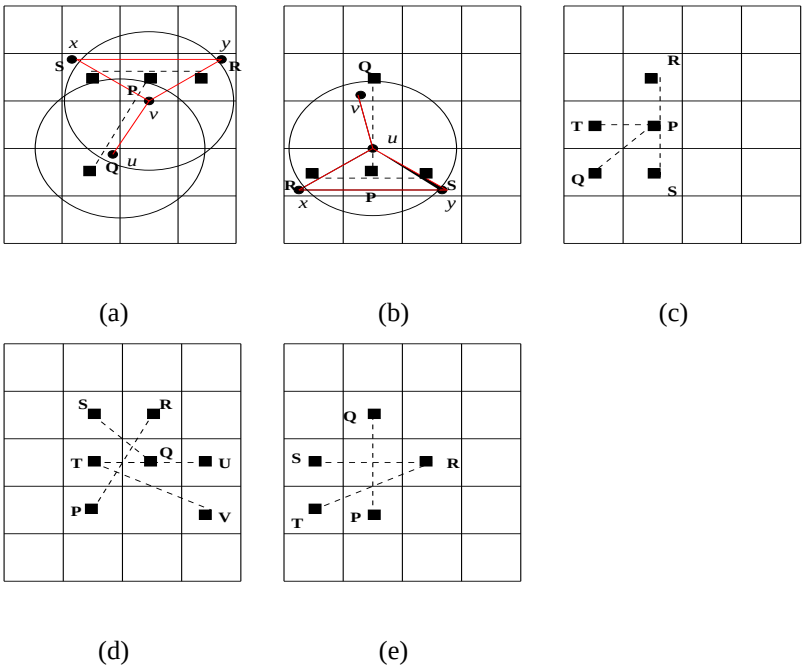


Figure 5.9: Base cases of non–planarity in the subgraph due to edge intersections between a long edge and a short edge, and between two long edges.

we know that there must be at least one node which is directly connected to the remaining three nodes. In the case that the UDG edges do not intersect (for e.g., edges uv and xy in Figures 5.9(a) and 5.9(b)), then it can be shown in these cases that $|\text{chord } xy| = \sqrt{2}r$, (chord xy is defined as the section of the straight line connecting x and y that intersects the circle that indicates the radio range of node v) and thus for $|xy| \leq r$, at least one of the nodes x or y must be connected to node u in the UDG. Since the UDG edges must intersect in Cases 5.9(d) and 5.9(e), the Lemma holds for long edges based on Lemma 5.4.3. □

Based on this property, our algorithm consists of two phases: the first phase creates a set of edges from the adjacency list that could potentially be part of the final edge set of the subgraph, and the second phase removes edges from this set that intersect while

preserving connectivity. Define $d(R)$ to be the degree of a vertex R in the overlay graph, $l(e_{R_i \to R_j})$ and $w(e_{R_i \to R_j})$to be the label and weight associated with an edge $e_{R_i \to R_j}$ in the overlay graph.

Algorithm Description

Algorithm 1 To construct the overlay graph (Phase 1)

```
1:  for all R_i do
2:      for all e_{R_i→R_j} ∈ E_s^{R_i} do
3:          Add e_{R_i→R_j} to E_P
4:      end for
5:  end for
6:  for all R_i do
7:      for all e_{R_i→R_j} ∈ E_l^{R_i} do
8:          if e_{R_i→R_j} is an axis-edge then
9:              R_k ← short vertex on e_{R_i→R_j}
10:             if e_{R_i→R_k} && e_{R_k→R_j} then
11:                 continue
12:             else if e_{R_k→R_j} then
13:                 Discard e_{R_i→R_j}
14:                 Add virtual edge e_{R_i→R_k} to E_P
15:             else if e_{R_i→R_k} then
16:                 Discard e_{R_i→R_j}
17:                 Add virtual edge e_{R_k→R_j} to E_P
18:             else
19:                 Add e_{R_i→R_j} to E_P
20:             end if
21:         else
22:             Add e_{R_i→R_j} to E_P
23:         end if
24:     end for
25: end for
```

Lines $1-5$ of Phase 1 add the short edges to the potential edge set E_P, and lines $6-25$ add the long edges between any two vertices in the overlay graph. While non–axis long edges can be readily added to the edge set (lines $21-22$), axis edges that

are long have to be added cautiously. Particularly, node configurations such as the ones shown in Figures 5.10 and 5.11 can cause overlay edge intersections along infinite points. For example, since all nodes are physically connected, there will be three edges in the overlay graph in figure 5.10 - namely, two short edges $e_{R_i \to R_k}$, $e_{R_k \to R_j}$ and a long edge $e_{R_i \to R_j}$. Since the long edge intersects with the short edges at infinite points, it is discarded without lose of connectivity due to the existence of path $R_i R_k R_j$ (lines $10 - 11$).

On the other hand, if the intersection due to axis edges is caused by a long edge and a short edge, and if the removal of one of the edge would cause disconnection, the algorithm does the following: it discards the long edge, and adds a virtual edge between the vertices that did not previously have a short edge between them (lines $12 - 17$). The virtual edge represents a 2-hop physical path in the underlying UDG. For example, in figure 5.11, long axis edge $e_{R_i \to R_j}$ is discarded and a virtual edge $e_{R_k \to R_j}$ is added to the edge set E_P. Edge $e_{R_k \to R_j}$ is constituted by the physical path $v_3 \to v_2 \to v_1$. Finally, If there is no intersections between axis edges, then the long edge can be safely added to E_P(lines $18 - 19$).

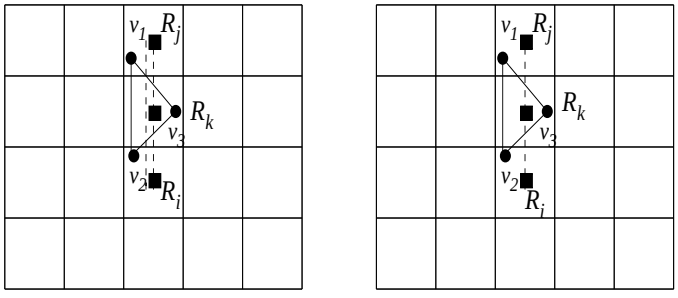


Figure 5.10: Long edge $e_{R_i \to R_j}$ can be discarded since the edge vertices $R_i$and R_j are reachable via the short edges $e_{R_i \to R_k}$ and $e_{R_k \to R_j}$.

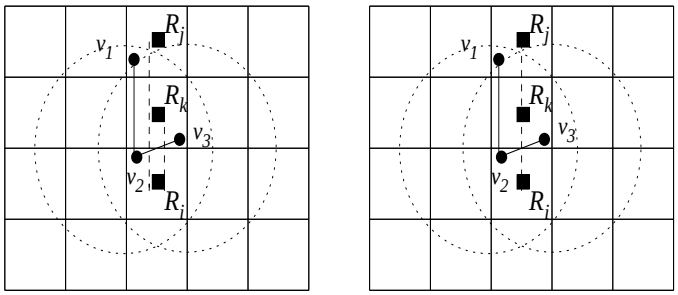


Figure 5.11: A short virtual edge $e_{R_k \to R_j}$ is introduced to eliminate the long edge $e_{R_i \to R_j}$. The virtual edge $e_{R_k \to R_j}$ represents the path $v_3 \to v_2 \to v_1$ in the unit disk graph.

After Phase 1 is completed, the degree of each vertex and the weight of each edge is computed in Phase 2.For Phase 2, we assume that an efficient data structure Q can be used to store the vertices sorted in ascending order of their degrees in line 6. Once Phase 1 completes, two edges can intersect in two ways: either at their vertices or at a single point along the edges, referred to as *regular* intersections. Clearly, only regular intersections can cause non–planarity. Phase 2 then considers *all* such regular intersections. Due to the fact that regular intersections can be detected by at least one node in the node pairs that caused the edge intersection, we can remove one of the edges. The procedure $remove - and - update$ considers an edge-pair involved in a regular intersection and removes one of the edges according to the following criterion:

- if the edges have unequal weights, remove the edge with the larger weight
- otherwise, remove the edge with the lower label

The procedure also updates the degree of the vertices as well as the weights of the outgoing edges incident at those vertices whose edge was removed. Q is then updated

Algorithm 2 To construct the planar overlay (Phase 2)

1: compute the degree $d(R_i)$ of each vertex R_i
2: **for all** $e_{R_i \to R_j}$ **do**
3: $l(e_{R_i \to R_j}) \leftarrow \max(i, j)$
4: $w(e_{R_i \to R_j}) \leftarrow \min(d(R_i), d(R_j))$
5: **end for**
6: insert each vertex R_i into Q in ascending order of degree
7: **while** Q is not empty **do**
8: $R_i \leftarrow$ first vertex in Q
9: **for all** $e_{R_i \to R_j} \in E_P$ **do**
10: **for all** $R_k \in N_i$ **do**
11: **for all** $e_{R_k \to R_l} \in E_P$ **do**
12: **if** $e_{R_i \to R_j}$ and $e_{R_k \to R_l}$ intersect **then**
13: $remove - and - update(e_{R_i \to R_j}, e_{R_k \to R_l})$
14: **if** R_i is not the first vertex in Q **then**
15: go to step 8
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **end for**
21: remove R_i from Q
22: **end while**

to reflect the changes due to the edge removal. Phase 2 then continues to process all vertices in order of ascending degree till all intersections have been resolved.

Theorem 5.4.4. *The centralized algorithm constructs a connected planar subgraph from the overlay graph in polynomial time, if the unit disk graph is connected.*

Proof. We prove that the algorithm described in section 5.4.2 converges and that it creates a connected planar subgraph from the overlay graph if the underlying UDG is connected. Note that Phase 1 does not lead to network disconnection since it discards a long edge only if the corresponding vertices are connected via two–edge paths. Thus we only need to show that when Phase 2 terminates, the resulting graph is both planar and connected.

From Lemma 5.4.2, if two overlay graph edges intersect, it must be due to two radio edges in the UDG and that one of the nodes must be directly connected to the remaining three nodes in the UDG. Thus, it follows directly that one of the overlay edges that caused non–planarity can safely be removed by that node without losing connectivity. However, this is true only if the redundant edges in question have not been removed prior to the current step of the algorithm. It could happen that one of the redundant edges could have been removed due to a previous edge intersection, and the assumption of the removed edge being valid in the current step could lead to disconnection (for e.g., see figure 5.13). Thus, as an added precaution, one needs to ensure that no edge with weight one be removed from the graph to prevent the disconnection problem.

Lemma 5.4.5. *Two edges that have overlay edge weights equal to one cannot intersect*

in the overlay graph.

Proof. This follows from the redundancy property. An edge weight of one indicates that one of the end points of that edge (say A) must have a degree equal to one. If one of the edges has a weight one, this means that the end points of the other edge (say B) must be connected to the other vertex in A. Thus both the vertices in B must have a degree of at least two. □

It follows from lemma 5.4.5 that if an edge with weight one intersects with another in the original overlay graph, then the other edge can be removed without causing disconnection. Thus, we only need to prove that when the algorithm proceeds to delete edges, an edge that had previously a weight of two cannot have a regular intersection with another edge of weight one after its weight was updated to one due to some edge deletion. We will prove this using contradiction. Consider a general scenario depicted in figure 5.12. Without lose of generality, assume that edge $R_1 \rightarrow R_3$ is removed to for the regular edge intersection $R_1 \rightarrow R_3$ and $R_2 \rightarrow R_6$. This would lead to edge $R_1 \rightarrow R_4$ to have a new weight of one. This would mean that by removing either $R_1 \rightarrow R_4$ or $R_5 \rightarrow R_6$, the algorithm would disconnect either vertex R_1 or R_5. However, this is not possible, since due to the redundancy property. There must be one of the edges $R_4 \rightarrow R_5$ or $R_1 \rightarrow R_6$, or both present in the graph, which would leave the graph connected.

At each stage, phase 2 of the algorithm considers any two edge-pairs that cause non–planarity and removes one edge by preserving the connectivity. In fact, an edge is removed if and only if the node-pair that belongs to that edge are reachable through

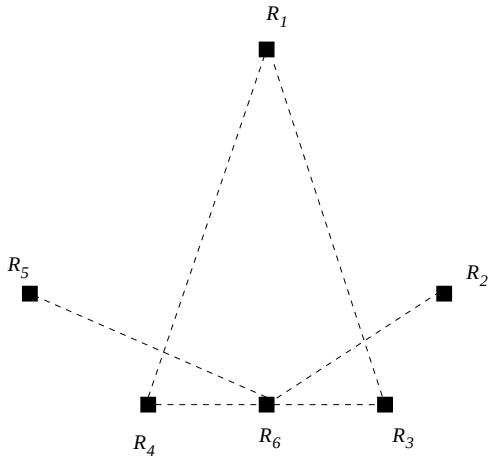


Figure 5.12: Proof of theorem 5.4.4

another node or a path in the UDG; i.e. each vertex in the overlay graph has a degree of at least one at any time. Since there is only a finite number of edges ($O(N^2)$) in the overlay graph, there are only finite cases of edge-pairs that cause regular intersections.

□

5.4.3 Distributed Algorithm

From the centralized algorithm, we note that the complete overlay graph information needs to be made available to all nodes in order to construct an unique planar graph view across all nodes. This would require a network flood of the overlay topology, which goes against the principle of localized protocol operation. It may be possible to create a localized version of the centralized algorithm using a time divided schedule and prioritized unit regions such that each unit region analyzes its connectivity, computes possible edge removals and broadcasts its set of preserved edges during its allotted slot. However, this imposes strick synchronization between neighboring unit regions as well

as its member nodes. In lieu of these obstacles, we modify phase 2 to remove regular intersections based on only the local view of the nodes. Thus, each node in region R_i includes its incident edges in the overlay graph in the periodic beacon to announce its location for geometric routing. Note that this information can be efficiently coded in a bitmap consisting of 20 bits, since there are at most 20 vertices that any vertex can be adjacent to in the overlay graph. After gathering adjacent region connectivity from such beacons, each node runs phase 1 and the procedure shown in algorithm 3 to compute the planar graph.

Algorithm 3 Distributed algorithm

```
1:  for all e_{R_i→R_j} in E_P do
2:      l(e_{R_i→R_j}) ← max(i, j)
3:  end for
4:  for all e_{R_i→R_j} ∈ E_P do
5:      for all R_k ∈ N_i do
6:          for all e_{R_k→R_l} ∈ E_P do
7:              if e_{R_i→R_j} and e_{R_k→R_l} intersect then
8:                  if e_{R_i→R_k} && e_{R_i→R_l} then
9:                      if e_{R_k→R_i} && e_{R_k→R_j} then
10:                         remove edge with lower label
11:                     else
12:                         remove e_{R_k→R_l} from E_P
13:                     end if
14:                 else
15:                     remove e_{R_i→R_j} from E_P
16:                 end if
17:             end if
18:         end for
19:     end for
20: end for
```

Once each node computes its final edge set in the overlay graph, it includes this information in the periodic beacon using an additional 20 bit flag. A node that receives such a broadcast packet from its neighbor updates its connectivity, and marks the outgo-

ing overlay graph edges to be preserved as advertised in the broadcast beacon. Clearly, this scheme is purely distributed, localized and energy efficient in nature. The overhead required to construct an overlay planar subgraph is at most 40 bits more than that required to keep track of the location of radio neighbors to facilitate geometric routing. However, it is possible that the graph which is locally constructed -albeit planar- is disconnected, as shown by theorem 5.4.6.

Theorem 5.4.6. *The subgraph constructed locally by the distributed algorithm is always planar, but does not guarantee connectivity.*

Proof. The proof is demonstrated by a conflict graph in which disconnection occurs. While this is not the only case in which disconnection occurs, it is sufficient to show that there is at least one case in which the algorithm leads to disconnection. In figure 5.13, nodes *u*,*v*,*w*,*x* and *y* can be placed in overlay vertices R_5, R_{10},R_{11},R_2 and R_3 respectively to produce the overlay graph as shown. For the regular intersection between edges $R_5 \rightarrow R_{11}$ and $R_2 \rightarrow R_{10}$, edge $R_2 \rightarrow R_{10}$ will be removed since all the four vertices in question can be reached either vertices R_{10} or R_{11} and the tie is resolved by removing the lower labeled edge; i.e., $R_2 \rightarrow R_{10}$. For the regular intersection between edges $R_3 \rightarrow R_{10}$ and $R_2 \rightarrow R_{11}$, edge $R_2 \rightarrow R_{11}$ will be removed since vertex R_{11} cannot connect to vertex R_3 directly. As a consequence, vertex R_2 will be disconnected in the final subgraph. □

Although the distributed algorithm cannot guarantee connectivity by the redundancy property alone, this many not be a severe problem since the cases in which specific node placements that lead to disconnection may occur rarely in practise. In section 5.5, we

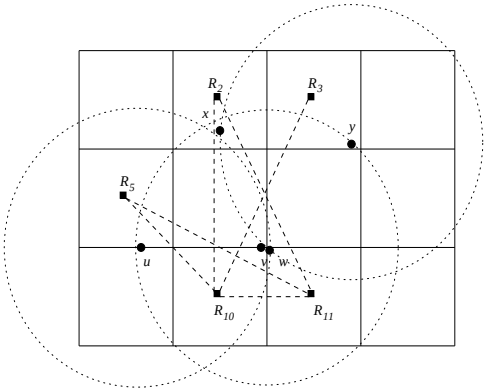


Figure 5.13: Proof of theorem 5.4.6

will show that this is indeed the case in most practical scenarios.

5.4.4 The GTA Routing Protocol

To better facilitate the location service layer, we propose a new routing protocol that we call the Grid Traversal Algorithm (GTA). Routing in GTA is ordinarily greedy forwarding using the neighbor location table (i.e. the UDG). However, the routing switches to face routing mode when greedy forwarding fails, similar to face routing schemes such as GFG [22] and GPSR [71]. The key difference in GTA is that face routing is carried out on the planar graph constructed from the overlay graph than one extracted from the unit disk graph. Periodic beaconing and the distributed implementation from section 5.4.3 is used to construct the overlay planar graph. The occurrence of local maxima could lead to two possible cases: either the destination point lies in an empty region inside the current face or the failure is a temporary void in the path to the destination. In the first case, face routing on the planar graph constructed from the overlay is car-

ried out and the packet will be successfully delivered to the proxy region if the planar overlay is connected. If the failure is temporary, then some intermediate node *y* would find itself closer to the destination location than the location at which face routing was initiated, and would continue greedy forwarding.

GTA also supports the proxy management phases by handling packets in face routing mode as follows. If a location update packet looped around, it is handed up to the location service layer. An intermediate node who is also a proxy server needs to look at all location update, query and proxy maintenance packets promiscuously, and hence GTA hands all such packets up to the location service layer. Finally, when proxy maintenance is being executed, GTA caches all control packets in face route mode that passes through the current region and passes them up to the location service layer when a proxy response is received. This avoids possible race conditions as mentioned in Section 5.2.1.

5.5 Numerical Study

5.5.1 Simulation Environment

To test the performance of our proxy management scheme, we implemented our proxy enhancements on SLURP [23], a simple and flat location management protocol in GloMoSim [88]. Since our objective is to test the effectiveness of our proxy enhancement and the new routing protocol, we implemented GTA as the routing protocol for SLURP with proxy, and compared the scheme against a combination of GPSR (as the routing

protocol) and SLURP without using the proxy enhancement. GTA has a adjacency vector table to keep track of adjacent vertices, and a list of next best hops to reach each adjacent vertex. A boolean variable is associated with each graph edge to indicate its membership in the planar graph. GPSR implementation was borrowed from the NS–2 implementation of the protocol at [93] and verified against published results by running the protocol using an ideal location management layer. The location of a destination node is known *a priori* using an *ideal* location management scheme, and serves as an upper bound for the performance of a practical location management protocol. The RNG scheme [94] was used to create the planar graph for perimeter routing in GPSR, since this scheme yields a less densely connected graph and leads to better performance of the routing protocol.

For GTA, proxy maintenance in a previously empty grid is done by transmitting a proxy maintenance packet along all graph edges and indicating its destination to be an invalid node located at a very small distance δ from the current location of the node that initiates the proxy maintenance process. The packet goes into perimeter mode right away, and loops around the perimeter of one of the faces to find the proxy server. If a proxy maintenance packet starts to loop at the initial point, it is assumed that the proxy server lies outside the face traversed by the packet, and is dropped immediately. If a proxy server receives the maintenance packet, it broadcasts to its current grid the end of proxy duties for the grid specified by the maintenance packet, and routes a proxy response containing a set of location entries back to the source grid that nodes in that grid should store.

We ran our simulations on a 2000x2000 m terrain consisting of 70 mobile nodes (average density of 1.75×10^{-5} nodes/m^2), in which the unit region is a $250m \times 250m$ square region. Although this is not a very sparse network, choosing scenarios that are even sparse may lead to frequent network disconnections and possibly meaningless results. For the simulation scenario, 1000 Constant Bit Rate (CBR) connections were randomly generated, with each session sending one packet with a 512 byte data payload. A session terminates successfully if the location discovery phase returns the correct location so that the data sent to the said location successfully reaches the destination before the simulation ends. Simulation parameters for the scenario are shown in Table 5.1. Each plot point presented in the next section is an average of seven simulation runs.

Table 5.1: Simulation parameters for GTA vs. GPSR

Simulation Time	900 sec	Mobility Model	Random Waypoint
Simulation Area	2000×2000m	Maximum Speed	0-30 m/sec
Unit Grid Size	250m	Minimum Speed	0 m/sec
Number of Nodes	70	Pause Time	0 sec
Transmission Range	350m	Traffic Type	Random CBR
Transmission Speed	54 Mbps	Number of Connections	1000
MAC Protocol	IEEE 802.11g	Data Payload	512 bytes
Beacon Interval	1 sec	Buffer Size	1000 packets

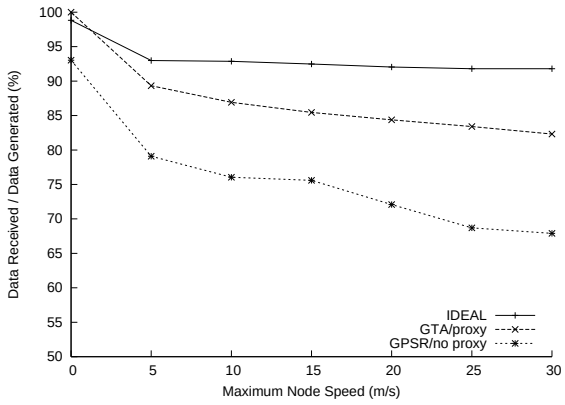
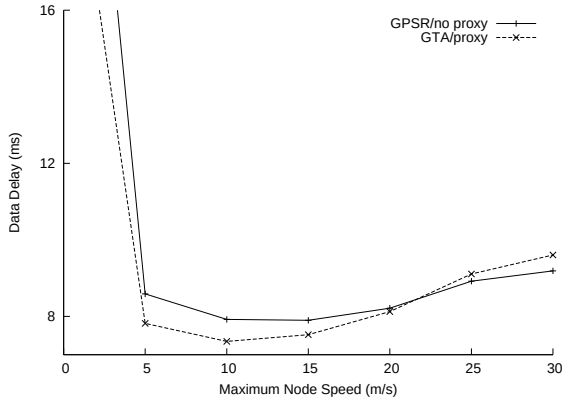


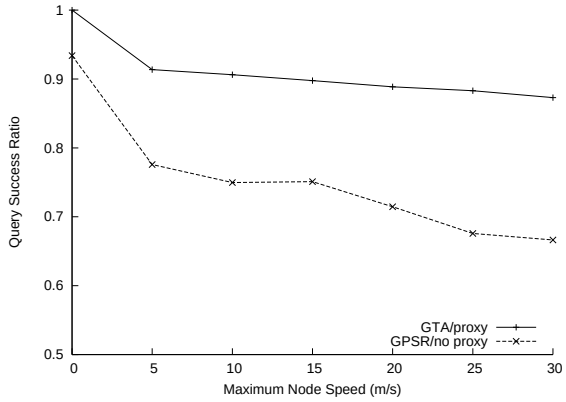
Figure 5.14: Fraction of successfully received data

5.5.2 Simulation Results

Figure 5.14 shows the fraction of data packets that were correctly received by destination nodes using both GPSR and GTA. The ideal location management plot serves as an upper bound for the maximum performance achievable by either routing protocol, since the exact location of a destination node is known as soon as the data packet arrives at the network layer. Clearly, the proxy based scheme is effective in tackling the empty server problem as shown by the increase in the fraction of data packets received by either routing protocol over that of GPSR without the proxy enhancement. As the average node speed increases, link changes occur more frequently, and fluctuations in the planar graph causes temporary loops, resulting in packet drops. Since the location of the destination needs to be found before data can be forwarded, the fraction of successfully received data packets is directly proportional to the success of the location discovery phase. As shown by Figure 5.15(b), the success ratio of queries is slightly more for GTA, indicating a more resilient proxy discovery as well as more stable planar



(a) Average end–to–end data delay

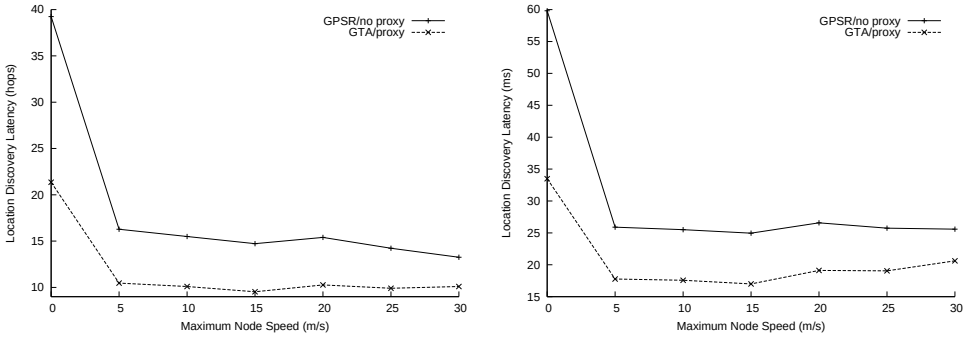


(b) Query success ratio

Figure 5.15: Data delay and Query success ratio

graph for GTA. Thus, GTA combined with our proxy enhancement performs better than GPSR without proxy in terms of the fraction of data packets successfully received.

Figure 5.15(a) shows the average end–to–end delay experienced by data packets. When the network is static, local maxima in greedy forwarding causes packets to be routed in face route mode most of the time, causing additional delay. However, node mobility removes some of the voids that were present in the static network, causing more packets to be forwarded greedily. Additionally, loops due to mobility cause face traversal to be unsuccessful. In general, successful face traversal also produces longer

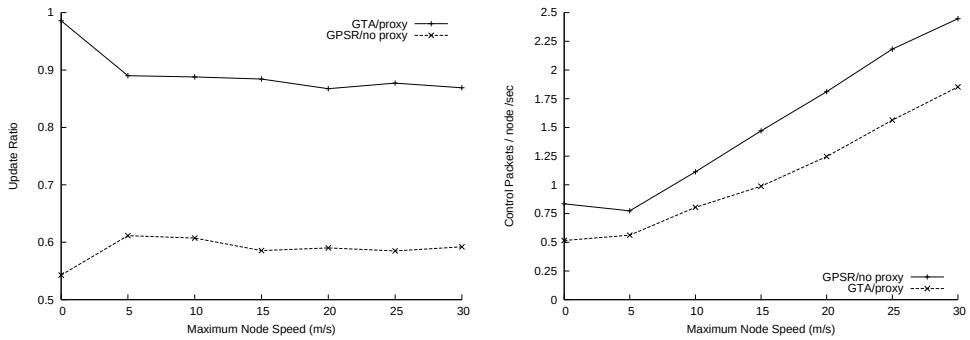


(a) Location discovery latency (hops) (b) Location discovery latency (msec)

Figure 5.16: Discovery delay

paths than greedy routing. Thus, the measurement does not take all unsuccessful long paths produced by face traversal into account which finally leads to shorter end-to-end delays and explains why there is a drop in data delay for both the plots. While RNG based routing is quite effective in GPSR, the 2–hop adjacency vector based routing and reduced control overhead (see Figure 5.17(b)) assists GTA to obtain a slightly lower average end–to–end delay for data packets. The increased delay towards the end of the plot is due to a higher fraction of successfully received data.

Figures 5.15(b), 5.16(a) and 5.16(b) show the performance of the location discovery process using the proxy scheme over the regular location management scheme. We note from Figure 5.15(b) that not all the queries are responded to in either schemes. Although face routing guarantees packet delivery when the graph faces remain stable and connected, this is no longer true when increased node mobility changes faces of the graph. Thus, packets that are in transit start looping using face routing in GTA as well as GPSR if graph faces change, and are dropped eventually. Location discovery can fail either when the query itself is dropped or when a previous update from a node was



(a) Update ratio (b) Average control overhead

Figure 5.17: Update ratio and control overhead

dropped, and the server failed to respond to the query correctly. Thus, query success ratio is below 100% even for GTA. However, as shown by the relative performance in Figure 5.15(b), the proxy scheme fares better in responding to location query packets than GPSR without proxy, which fails to answer queries for nodes who have empty servers. Additionally, GTA is effective in finding proxy servers using the overlay subgraph, and is able to route queries faster, as indicated by the delay in hops as well as seconds in Figures 5.16(a) and 5.16(b).

Figure 5.17(a) shows the effectiveness of the routing protocol in routing location update packets to their respective servers. Since the regular scheme does not have proxy servers standing by for empty servers, packets to empty servers simply loop around before being dropped eventually. The update ratio is quite low in this scheme, indicating the potential breakdown of the location management protocol. However the proxy based scheme is able to recover from the empty server problem, and update packets are typically routed to their respective proxy servers. Since graph faces do not remain constant in practice and changes with node mobility, a change in a graph face causes

packets that are face routed to loop around that face, and are eventually dropped when their time–to–live hits the limiting value. Thus, even in the GTA scheme, update ratio is below 100%.

Figure 5.17(b) shows the control overhead in number of packet transmissions incurred per node per second. Since packets loop around in the absence of proxy servers, GPSR without proxy suffers from increased resource wastage. Also, since routing is based on edges of the RNG graph, control packets take longer paths and transmissions to reach their respective destinations. GTA performs better in terms of control overhead, since it is able to route packets to servers/proxy servers using shorter paths and with lower overhead.

5.6 Summary

In this chapter, we have shown that the proposed location management protocols suffer from the empty server region problem, which becomes pronounced when the network is sparse. Our analytical model combined with simulations confirm the adverse effect of node density on the empty server region problem. We have presented a proxy based location management enhancement by delegating the task of empty server regions to adjacent non–empty unit regions. While the concept of proxies is simple, distributed proxy selection and management can be quite complicated in ad hoc networks, and we have shown that our scheme avoids most of the pitfalls encountered while designing such a protocol. We have also presented a centralized algorithm that constructs a con-

nected planar graph from the overlay graph defined on the unit regions in the terrain. The distributed version of this scheme, known as the Grid Traversal Algorithm (GTA), is a novel geometric routing protocol that only uses the redundancy property of overlay graphs to construct planar overlay graphs. Using this property alone does not guarantee connectivity, but simulations show that disconnection is rare in practise. The proxy enhancement combined with GTA outperform SLURP/GPSR, a conventional location management scheme using a known face routing protocol that routes on a connected planar subgraph extracted from the unit disk graph.

Chapter 6

Location Based Multicasting

An important criterion for a unicast protocol to be be contender as a standard routing protocol is its ability to support efficient multicasting. The *Steiner tree* problem for graphs is well known in literature to be NP-Hard. In this chapter, we outline an extension to our proposed location management schemes to support position based multicasting in large ad hoc networks, and study the performance of the scheme against an existing location based multicast protocol for wireless ad hoc networks described in literature.

6.1 Related Work

6.1.1 Steiner Tree Problem in the Internet

Multicasting in packet switched networks has been widely researched due to the benefit obtained in bandwidth reduction and communication costs where many nodes are active

participants in the same data session. In such group communication consisting of N sources and receivers, significant reduction in communication cost can be obtained by transmitting a packet along the *minimum* set of links in-order to reach the subset of receivers than sending $N-1$ copies of the same packet from the source. The primary problem that arises from such a delivery framework is to compute the delivery tree that minimizes the network cost. The formal definition of the problem, called the *Steiner tree* problem for graphs is as follows: given a weighted graph in which a subset of vertices are identified as terminals, find a minimum-weight connected subgraph that includes all the terminals. The problem has been shown to be NP-Hard [29], and no optimal solution is known that solves the problem in polynomial time for asymptotically large networks.

A more general problem for computing the Steiner tree on the point set is well known and many any heuristics have been proposed for the problem (see [95] for a survey). However, the main objective in the computer network domain is to compute the minimum delivery tree with minimum overhead. Under the assumption that the topology is known (for e.g., via an unicast routing protocol), heuristics based on the *minimum spanning tree* and *Traveling Salesman problem* was suggested in [96] and [97]. Distributed solution using *reverse path forwarding* [98] and its variants to create the multicast tree in the Internet and LANS were suggested in [99]. Enhancements to IP protocols such as OSPF and RIP to support multicasting using source based shortest path trees and *pruning* unused links were suggested in [100] and [101]. The use of shortest path trees ensure that the packets take the minimum delay path to each multicast

receiver.

A main drawback of having protocols that forward multicast packets using source based shortest path trees is the issue of scalability as the number of sources per multicast group increases. If there are S sources per group in a network, the router storage required for state maintenance increases as $S \times N_g$, where N_g is the total number of multicast groups in the network. Additionally, the network incurs extra overhead in maintaining each tree as multicast nodes dynamically leave and join the network. In order to provide better scalability, the *Core Based Tree*(CBT) architecture was suggested in [31]. The idea is to construct a core tree for each multicast group such that the storage required to maintain the state of multicast trees is reduced to N_g. The core routers are used to deliver multicast packets to receivers via non-core routers attached to the core routers via shortest routes. Since the very idea of a core contradicts the minimum delay path between a source and a multicast receiver, [102] proposed combining the best of shortest path trees and CBTs to support sparse multicast groups over wide area networks.

6.1.2 Multicasting in Mobile Ad hoc Networks

Similar to unicast ad hoc routing, the inclusion of node mobility presents a cumbersome problem in using the schemes proposed in the previous section for multicasting in mobile ad hoc networks. Constructing and maintaining multicast delivery trees with mobility incurs considerable bandwidth usage in the form of control information. Noting the efficient operation of on-demand protocols in the ad hoc environment, [103], [25] suggested constructing source based delivery trees by flooding *join* requests. Similarly,

an extension to support multicasting using on demand route requests and pruning was suggested for AODV in [104]. Other on-demand protocols for ad hoc networks are ADMRP [28], AMRIS [105] and ABAM [106]. Similar to the approach from CBTs, protocols that adopt a shared tree or mesh based on topology information have also been proposed and can be found in [27], [26] and [107].

We note that most of the above protocols suffer from similar drawbacks as that of unicast routing protocols in large ad hoc networks and high node mobility. State maintenance via flooding or periodic messaging to keep the router states afresh can become quite expensive in rapidly changing topologies where the multicast sessions can scale both vertically and horizontally (i.e. increase in number of nodes per multicast group as well as the number of groups). Recently, a few proposals have been suggested for scalable multicasting in ad hoc networks. These include *Differential Destination Multicast*(DDM) [108], hierarchical DDM [109] and Overlay multicast [110]. The key idea behind differential destination multicast is to move the state storage responsibility from the router and include a set of multicast destinations in the form of in-band signalling in the packet header. At each intermediate node, the unicast routing protocol is used to deliver the packet to the destinations in the header. Additionally, a *soft state* can be created in each router by caching the set destinations for each forwarded data packet. If the network is static, then the future packets need not have a header containing the next set of destinations. In case of changes in this state, only those new destinations which need to receive multicast packets (or old ones which do not) need to be included in a new data packet. Hierarchical DDM partitions the network into a two-level hierarchy in which

each partition consists of a set of multicast receivers with an elected leader. DDM is then used to deliver the packets to the leaders and the leaders forward the packet to the destinations under them using the unicast routing protocol. Finally, overlay multicast creates a source based overlay tree consisting of only multicast nodes as the vertices and a virtual edge connecting the vertices. A modified distance vector protocol is used to refresh each multicast receivers' knowledge of its *virtual* neighbors. Packet delivery is carried out using the unicast routing protocol and coding the subgroup tree inside the packet header.

While DDM solves the storage issue, it introduces another bottleneck in the form of large headers in each data packet, especially if the network is highly mobile and the multicast session scales vertically. Additionally, since packets are forwarded using the unicast routing protocol, the detrimental factors affecting the routing protocol can adversely affect packet forwarding, as mentioned in chapter 2. Although hierarchical DDM alleviates the header problem slightly, maintenance of hierarchies with mobility is still a daunting task. Overlay multicast suffers from scalability problems in both maintaining the virtual overlay, as well as the unicast routing problem.

6.1.3 Position Based Multicast Protocols

Noting that the use of locations is potentially beneficial, a few protocols can be found in literature that describe the use of locations in multicast packet forwarding. Multicast extensions to LAR was proposed in [111] using controlled flooding of multicast data in forwarding zones. In [112], Basagni et. al. suggested the use of network flood-

ing of node locations and group information to create an MST approximation of the Steiner tree. The approximate tree consisting of j nodes can then be coded efficiently in the packet header using a *Prüffer* sequence, and consists of $O(j)$ entries. Each intermediate node decodes the tree, and continues packet routing on sub-trees rooted at that node. In order to find a balance between delay and data overhead, [113] suggested localized packet splitting and a combination of greedy geographic forwarding and perimeter multicast on planar graphs. Additionally, [114] proposed multicast group membership management and routing similar to that in [82].

Perhaps the most sensible and simplistic location based multicast protocol in recent times has been proposed by [30]. Assuming that each multicast node in the network is aware of the locations of its group members, the work in [30] proposes three location guided tree construction algorithms: *Location-Guided Directional Tree (LGD*, *Location-Guided K-ary Tree (LGK)* and *Location-Guided Steiner Tree (LGS)*. Of the three schemes, the LGS scheme was shown to perform best in terms of network cost and delay for small groups. The basic idea of the scheme is to use a modified Takahashi-Matsuyama heuristic [97] to build an *overlay* tree starting at the source node and that consists of only multicast group nodes, and use greedy geographic forwarding to route packets along each link in the tree.

In general, location based multicast problem can be divided into two sub-problems: (i) to find a location management scheme that incurs the least overhead in updating each member node with other member nodes' (or all the network nodes') current locations and (ii) to find a minimum cost delivery tree using the locations thus obtained. The latter

problem is simply the Steiner tree problem, and an optimal solution may not be found in the near future even if ideal knowledge of accurate locations of all nodes is assumed at each node. On the other hand, the former problem is more contingent, due to the resource constraints associated with ad hoc networks. All of the location based protocols described in this section use periodic flooding to update member nodes' locations and hence are clearly not scalable with large network sizes and high mobility. Additionally, network wide broadcasting can cause frequent contention and control packet losses due to interference/collissions. If a MAC protocol such as IEEE802.11 is used to broadcast the periodic location beacon, there is no guarantee that the packet will be reliably received by radio neighbors. Thus, the Steiner tree which is constructed may not even contain all multicast members as part of the tree, leading to reduced group throughput! Thus, one may need to rely on an unicast based location management protocol to reliably disseminate location information in the network.

6.2 Location Guided Core for Scalable Multicasting

In this section, we describe our solution for a scalable location based multicasting protocol. Our approach uses the best of both a hierarchical solution combined with the notion of a core similar to the CBT concept. The idea is to partition the terrain into sub-areas consisting of multicast nodes of a group. A localized protocol is used to deliver packets within this group, while the core tunnels each multicast packet in a unicast envelope to each sub-area. The key components of the protocol consists of operations to manage

the core membership as well as the dynamic group membership within each sub-area.

Similar to the unit region based protocols described in the previous chapters, we start off with a terrain divided into constant sized Order-1 square regions. Similar to SLALoM, we combine K^2 such regions to form R unique Order-2 regions, and in each Order-2 region, we choose an Order-1 region as a multicast region or $M-region_i$ that acts as a multicast representative for the multicast address i. The mapping between the M-region to the multicast address is based on a function chosen *a priori*, which takes the unique multicast address as the *key* to randomly assign one of the unit regions as the M-region. Thus, any node registered to a multicast address i that wishes to discover an M-region in its current Order-2 region can do so by applying the function to the multicast address. The core regions of the multicast tree for session i consists of all regions M_{ij}, where $j \in R$. A minimum delivery tree connecting the core regions can then be approximated by a minimum spanning tree connecting these regions, and forms the location guided core for the multicast session i.

However, depending on the spatial distribution and mobility of multicast nodes of session i, some the Order-2 regions may not contain any recipients and thus these M-regions would need to remove themselves from the core. Similarly, when a multicast node is the first to enter an Order-2 region, then that M-region should now join the core. Thus, the construction of the core consists two phases: a *location update* phase and a *core maintenance* phase. The former is used to update the current locations of multicast members within an Order-2 region, while the latter is required to update the core by keeping track of which of the core regions currently have multicast members registered

in them.

6.2.1 Location Update and Location Guided Steiner Construction

On each crossing of a region boundary within Order-2 R_j , each multicast receiver of session i must geometrically route a *location update* packet to M_{ij}. Additionally, if the boundary crossing is such that the node moved from an Order-2 region R_k into Order-2 region R_j, it must also send a duplicate update to M_{ik} in R_k indicating its departure from R_k. The updates are geometrically routed to the respective M-regions. The first server node in each M-region to receive the update carries out a broadcast of this information in that region to maintain the consistency of the recipient's location information stored in all servers. Thus, the servers in each M-region are aware of the number of multicast recipients in their respective Order-2 regions and their most recently updated locations. A local multicast delivery tree is then constructed in each Order-2 region using registered recipients' locations using the location guided steiner tree as in []. The algorithm to construct the tree is given below for clarity. l_k represents the location of the k^{th} multicast receiver n_k in R_j, $l_{M_{ij}}$ is the center of M-region M_{ij} and Q is the location guided steiner tree. Figure 6.1 shows an example of how the location guided steiner tree is

Algorithm 4 To Compute the Location Guided Steiner Tree

add pseudo node p with location $l_{M_{ij}}$ to Q
repeat
  $u \notin$Q$\leftarrow$ minimum distance node to Q
  $v \in$Q$\leftarrow$ minimum distance node to u
  add u to Q
  add undirected edge uv to Q
until all n_k have been added to lgs

constructed by the algorithm within an Order-2 region. Node *u* is a server node located in the M-region, and who has an up-to-date information of all the multicast receivers in that region. It adds each multicast member to the steiner tree in increasing order of distance to the center of the M-region, such that the cost of the tree is minimum. Thus, node n_1 is added first, then node n_2, n_3 and so forth, till all the multicast nodes are added to the tree. Note that the dashed line indicates an edge in the tree and may consist of multiple radio hops via intermediate nodes not in the tree.

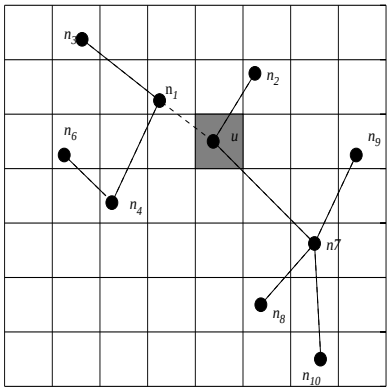


Figure 6.1: Example of a location guided Steiner tree construction

6.2.2 Core Maintenance and Location Guided Core Construction

Once the membership of multicast receivers in an Order-2 region is determined from the location update phase, the decision to include (or not include) an M-region in the core is advertised by each M-region to all other M-regions via a *core update* packet using a minimum spanning tree connecting all core regions. The core update packet consists of the M-region identifier and a status flag to indicate its decision to *join* or *prune* from the multicast core.

Joining the Core: If a receiver r of multicast session i enters an Order-2 region R_j such that it is the first receiver of session i in R_j, the location update phase would indicate to a server in M_{ij} that M_{ij} should now join the multicast core such that the multicast data forwarded through the core can be forwarded to r. The join decision is advertised via the core update packet to all M-regions, using a minimum spanning tree that connects all M-regions, and is identical to that in SLALoM.

Pruning from the Core: Pruning from the core is analogous to the join operation. If a receiver r of multicast session i leaves an Order-2 region R_j, it will send a location update packet to the servers in M_{ij} indicating its departure. If the departure of the node is such that it is the last receiver of session i in R_j, the server that receives the update packet in M_{ij} should now prune the multicast core such M_{ij} is now removed from the core. The prune decision is advertised to all M-regions similar to the join operation.

Location Guided Core Construction: Each server node in an M-region M_{ij} keeps a bitmap of all M-regions to keep track of M-regions that are currently part of the core. The bitmap is updated when a server node receives a new core update packet from an M-region M_{ik}. The server then broadcasts the packet inside M_{ij} to make the bitmap consistent across all servers in M_{ij}, before forwarding the packet to the next M-region(s) in the spanning tree.

Due to the core maintenance phase, all server nodes in M-regions have an up-to-date information regarding the status of the membership of each M-region in the core. A unique location guided core is then constructed by each server node by running algorithm 5 locally. The problem of finding a minimum delivery tree connecting the core

M-regions is known to be NP-Hard. Thus, we use a modified Takahashi-Matsuyama [] to build an approximate delivery tree as follows.

Denote by C the set of k M-regions which are part of the multicast core, by T_R the complete graph in which the vertices correspond to the R unique M-regions in the terrain, and any edge in the graph corresponds to the straight line connecting the centers of two M-regions. The cost of an edge is defined to be the euclidean distance between the center points of its corresponding vertices. Figure 6.2 shows an example of how

Algorithm 5 To Compute the Location Guided Core

$D \leftarrow$ all pair shortest paths of T_R
$V \leftarrow \{M_1\}, M_1 \in C$
$E \leftarrow \phi$
$T \leftarrow (V, E)$
$\hat{c}(T) \leftarrow 0$
for all i such that $2 \leq i \leq k$ **do**
  find $v \in C - V$ with minimum cost path to any of the vertices in V from D
  $V \leftarrow V \cup$ vertices in the minimum cost path
  $E \leftarrow E \cup$ edges in the minimum cost path
  $\hat{c}(T) \leftarrow \hat{c}(T) \cup$ cost of the minimum path
end for

algorithm 5 constructs the location guided multicast core tree. The greyed unit regions indicate M-regions that have multicast receivers registered in their corresponding Order-2 regions (indicated by the inner grid in the Order-2 region) and hence the algorithm should create a minimum tree connecting these regions. In the figure, these correspond to the Order-2 regions R_1, R_2, R_5, R_7, R_8 and R_{11}. R_1 is added to the tree first due to the order in which vertices are processed by the algorithm. Since the edge R_1R_2 is the shortest edge among the edges between the remaining vertices, the M-region in R_2 as well as the edge R_1R_2 is added to the tree. Similarly vertices R_8, R_5 and R_7 are added

to the tree. M-regions R_3,R_4 and R_6 are also added to the tree since the shortest paths between R_2 and R_5 and R_5 and R_7 lie through them. Finally R_{11} is added to the tree via the M-region R_3 since the cost of adding R_{11} to the tree is minimized by connecting it via R_3. The dashed lines indicate the edges in the final multicast core.

Note that the distributed version of the algorithm will produce an unique tree in all the server nodes running the algorithm if all the nodes in the M-regions have a consistent view of those M-regions which have joined the core. In case of ties in two minimum shortest paths, the first path is chosen to be included in the core. Also, algorithm 5 differs from that of the location guided steiner construction in the sense that non-core M-regions may be part of the core while only multicast receivers are part of the delivery tree in any Order-2 region.

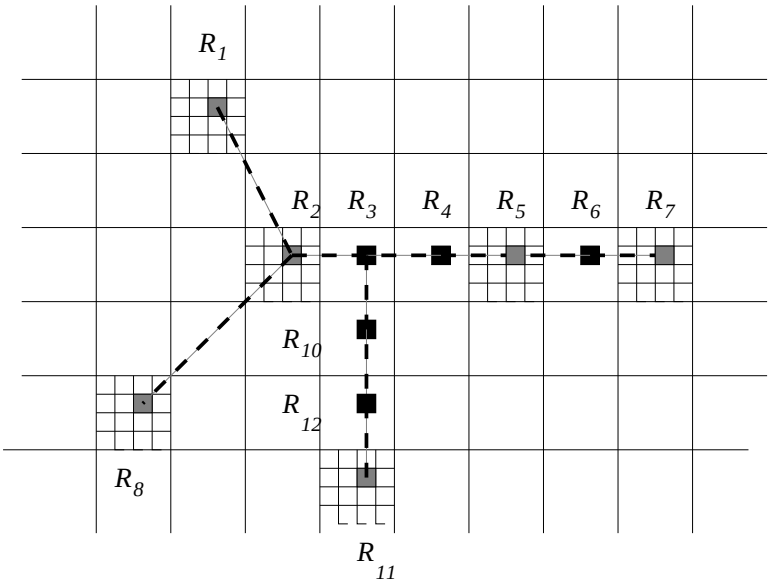


Figure 6.2: Example of the multicast core tree construction

6.2.3 Location Maintenance

Each node must also store the information relevant to an M-region if it crosses a region boundary into a new M-region. The relevant information is the current locations of all multicast receivers registered to that M-region, as well as the core membership status of all M-regions in the terrain. Algorithms 4 and 5 are used to create the local delivery tree as well as the core once the node has this information. Similar to the unicast location management protocols, this maintenance can be done by a *broadcast-reply* protocol that involves only those nodes in the new M-region.

6.2.4 Multicast Data Delivery

Once the core and the local delivery trees have been constructed, multicast data delivery is fairly straight forward. A multicast source S of session i located in Order-2 region R_j forwards the data to its current M-region M_{ij} via geometric routing. The first server node u in M_{ij} to receive the data packet does the following:

- u checks the incoming edge (i.e. previous vertex on the core) on which the packet arrived. If there are outgoing edges from this vertex, u copies the data packet along all outgoing edges in the multicast core. If there are no outgoing edges, u is located in an M-region which is a leaf vertex, and hence stops the forward phase along the core.

- If there are multicast recipients in M_{ij}, u copies the packet onto each outgoing edge of the local delivery tree constructed using algorithm 4. Furthermore, u

codes the subtree under each outgoing edge into each data packet it forwards along that edge. This is done so that the multicast recipients who receive the packet can then forward the packet to other recipients in the local delivery tree.

Finally, each multicast recipient in an Order-2 region which receives the data packet looks in the header to see if there are additional recipients encoded in the packet header. If so, the packet is further copied along all outgoing edges with the corresponding subtrees encoded in the header.

6.3 Scalability Analysis

A framework similar to the one carried out in chapter 4 can be done to analyze the multicast location management cost for the multicast location management protocols. The cost of location management is simply the cost incurred due to different phases of the protocol - namely, location update, core maintenance, location maintenance and additional overhead in data delivery. We assume a terrain of size A, N nodes, a constant node density $\gamma = \frac{N}{A}$, unit region of size a, an average node velocity of v and N_g multicast receiver nodes per group. Denote by ρ_1 the rate at which a node crosses Order-1 regions, and by ρ_2 the rate at which a node crosses Order-2 regions.

Location Update Cost: Recall that each time a multicast receiver node moves into a new Order-1 region, it has to inform its current M-region of its current exact location. This consists of one geocast update to an M-region. Furthermore, if such a move also causes the node to move into a new Order-2 region, then it has to inform its previous

M-region of its departure from the previous Order-2 region. This requires an additional geocast update to the previous region. Thus, we have the location update overhead per unit time for a multicast receiver node to be

$$c_{lu} = O(\rho_1(u+b)+\rho_2(u+b)) \text{ packets/sec/node} \quad (6.1)$$

where u is the number of transmissions required to reach the M-region, and b is the broadcast cost inside the M-region. Noting that $\rho_1 = \frac{v}{\sqrt{a}}$ and $\rho_2 = \frac{v}{K\sqrt{a}}$, we have

$$\begin{aligned} C_{lu} &= O(\frac{v}{\sqrt{a}}(\frac{K\sqrt{a}}{z}+b)+\frac{v}{K\sqrt{a}}(\frac{K\sqrt{a}}{z}+b)) \\ &= O(\frac{vK}{z}) \text{ packets/sec/node} \end{aligned} \quad (6.2)$$

where z is the *average forward progress* as defined in chapter 4.

Core maintenance Cost: A core maintenance packet is sent by a server node to update all M-regions of the core membership status of the M-region that it is currently located in. A minimum spanning tree connecting all the M-regions is used to sent the packet, and hence the cost of sending a packet using the tree is $O(\frac{A}{Kz})$. Since there are $O(\frac{A}{K^2})$ M-regions, this results in $O(\frac{A}{K^2})$ broadcasts, one within each M-region. In the worst case, when a node crosses an unit region boundary into a new Order-2 region, two core maintenance updates are generated: one by the new M-region denoting its decision to join the core and the second by the previous M-region, indicating its decision to prune itself from the core. Thus, the core maintenance cost can be calculated as

$$\begin{aligned} C_{cm} &= O(\rho_2 \times 2 \times (\frac{A}{Kz}+\frac{Ab}{K^2})) \\ &= O(\frac{vN}{K^2 z}) \text{ packets/sec/node} \end{aligned} \quad (6.3)$$

Location maintenance Cost: When a node moves to a new order-1 square, location maintenance covers the cost of two broadcasts: the first to query a node in the current region for M-region information, and the second to reply to the query by a node already present in the unit region. Since nodes cross unit regions at the rate of ρ_1 per unit time, the location maintenance cost can be computed as

$$\begin{aligned} C_{lm} &= O(\rho_1(2b)) \\ &= O(v) \text{ packets/sec/node} \end{aligned} \tag{6.4}$$

Data delivery cost: Due to the location management operation, a multicast data packet is delivered to a recipient indirectly via the core and then the local delivery tree. This leads to additional transmissions that would not have been incurred had the exact locations of all multicast receivers be known to all nodes and an optimal tree could be constructed using all known node locations. The worst case for which a data packet is indirectly forwarded to the next receiver in the core is shown in figure 6.3. Node v is the multicast source, and forwards the packet to the M-region M_1 in its current Order-2 region over a distance d_2. The packet is then forwarded to the next M-region in the core, M_2, over a distance which is the sum of distances d_3,d_4 and d_5. The packet is then forwarded to the next multicast receiver u over a distance d_6. However, the least cost distance between nodes u and v is d_1, and thus the additional distance traversed by the

packet due to the location management cost is

$$\begin{aligned} d &= d_2 + d_3 + d_4 + d_5 + d_6 - d_1 \\ &= d_2 + d_2\text{Cos}\theta_2 + d_6 + d_6\text{Sin}\theta_1 - (d_1 - d_4) \\ &< 4K\sqrt{a} \end{aligned} \tag{6.5}$$

Thus, assuming that packets arrive at each node at a rate of λ packets/sec according to

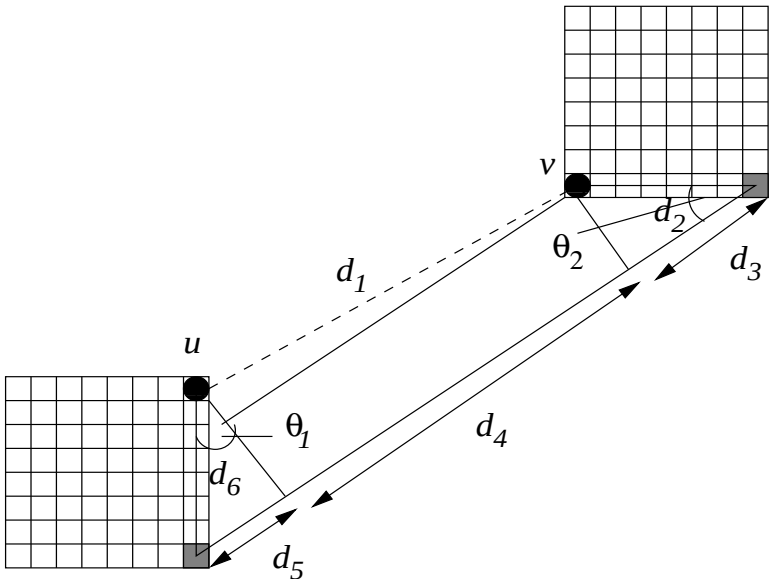


Figure 6.3: Additional overhead due to the core

a Poisson process, the additional cost due to data delivery can be calculated as

$$\begin{aligned} C_d &= O(v \times \frac{4K\sqrt{a}}{z}) \\ &= O(\frac{vK}{z})\text{packets/node/sec} \end{aligned} \tag{6.6}$$

Total Overhead Cost: Combining the results above, we have the total overhead cost

for the multicast group.

$$\begin{aligned} C_t &= N_g \times C_{lu} + N_g \times C_{cm} + N \times C_{lm} + N_g \times C_d \\ &= O(vN_gK + v\frac{N_gN}{K^2} + vN + vN_gK) \\ &= O(vN_gK + v\frac{N_gN}{K^2} + vN)\text{packets/node/sec} \end{aligned} \tag{6.7}$$

Theorem 6.3.1. *The average total overhead cost per group for location management in the location guided core protocol is $O(vN_gK + v\frac{N_gN}{K^2} + vN)$ packets per second, which is minimized when $K = \theta(N^{1/3})$. That is, when K is chosen appropriately, the average total overhead cost of our protocol is $O(vN^{4/3})$ packets per second per multicast group.*

Proof. The proof follows from minimizing the total overhead cost with respect to K and assuming that $N_g = O(N)$. □

6.4 Numerical Study and Discussion

To test the performance of our multicast scheme, we implemented both our protocol (LGC) and the Location Guided Steiner (LGS) protocol in Glomosim. In order to have a fair comparison between both protocols, the location update rate in LGS was kept to be similar to that of LGC. Thus, in LGS, a node does a network broadcast of its current location when its crosses an unit region boundary. We kept the size of an Order-2 region to be four unit regions, since this will have maximum impact on the control overhead in LGC. Simulation parameters for the scenario are listed in table 6.1.

In this study, we studied the performance of both protocols for varying node mobility. The minimum speed in the Random Waypoint model was kept at 1 m/sec to alleviate

Table 6.1: Simulation parameters for multicast scenario

Simulation Time	300 sec	Mobility Model	Random Waypoint
Simulation Area	2000×2000m	Maximum Speed	0-30 m/sec
Unit Grid Size	250m	Minimum Speed	1 m/sec
Number of Nodes	400	Pause Time	0 sec
Transmission Range	350m	Transmission Speed	54 Mbps
Traffic Type	Random CBR	Multicast groups	10
Data Rate	2 packets/sec	Multicast receivers per group	10
MAC Protocol	IEEE 802.11g	Data Payload	512 bytes
Beacon Interval	1 sec	Buffer Size	1000 packets

the average speed decay problem [115]. The maximum speed was varied to study the effect of increasing average node velocity on protocol performance.

Figure 6.4 shows the fraction of data packets received for either protocol. LGC has a superior performance over LGS for the entire range of node velocities. When the nodes are static, the number of data packets that the protocol is successfully able to deliver is less than 100% in LGS. The throughput decreases as node mobility increases and slightly increases towards the end. This behavior can be explained due to the broadcast storm problem [19] in ad hoc networks, and the stochastic properties of the Random Waypoint model [116]. When the average node density is high, the large number of 802.11 broadcast collisions lead to many of the location updates being lost when the network is static. Thus, the approximate Steiner tree constructed in LGS may not con-

tain all the receivers, and leads to a fraction of successfully delivered data packets lower than 100%. Increasing average mobility reduces the average number of neighbors in the Random Waypoint model, and helps alleviate the broadcast storm problem. However, this does not reduce loss of data packets. Since increases node mobility outdates the perceived view of the delivery tree in each node, packets sent to expired locations may be unsuccessful. Towards the end, as the number of location updates sent by each node increases with mobility, it helps to maintain a more up-to-date delivery tree in LGS, and helps data throughput. The number of data packets received in LGC decreases steadily due to outdated trees caused by increased mobility.

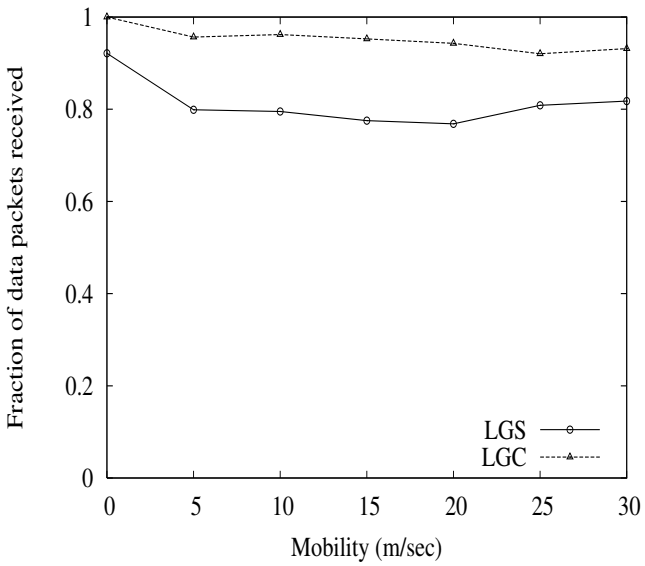


Figure 6.4: Data throughput

Figure 6.5 shows the average delay experienced by data packets. Average delay may be considered less important, since our objective is to minimize the overhead and not delay. However, average delay sheds light on the behavior of the multicast trees constructed by either protocol and is useful to evaluate the performance of the protocols.

Clearly, the LGS tree is a better approximation of a minimum delivery tree compared to the location guided core, since a packet at an intermediate node is forwarded to a next hop along the straight line connecting both the nodes in LGS. On the other hand, the core is a simple approximation of where multicast nodes are present, and may incur extra transmissions to reach a multicast receiver from the M-region. Additionally, since the locations are flooded in the network by LGS, there is no need to include any other information in the data packet except the next hop node information. On the other hand, each data packet in LGC must contain the coded subtree of the local delivery tree in its header in order to continue packet forwarding by intermediate nodes. Both the above increase the average data overhead in LGC 6.6, leading to a higher average delay for data packets compared to LGS.

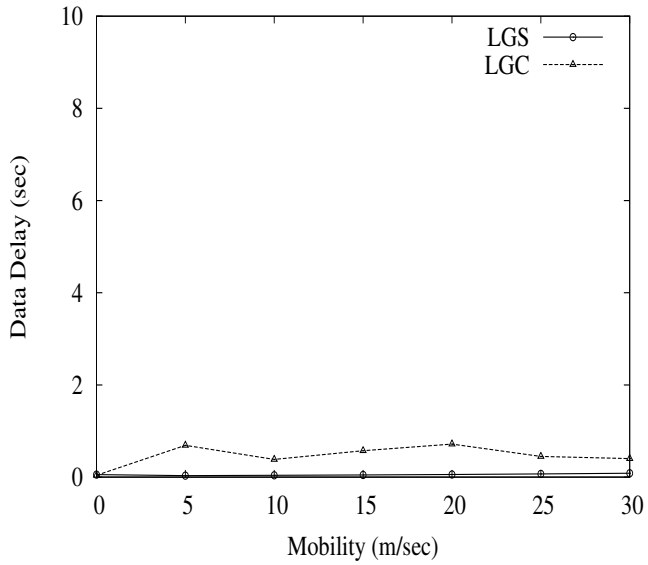


Figure 6.5: Average data delay

Finally, figure 6.7 shows the protocol control overhead incurred by both protocols in number of bytes transmitted per node per second. For LGS, the protocol packets are the

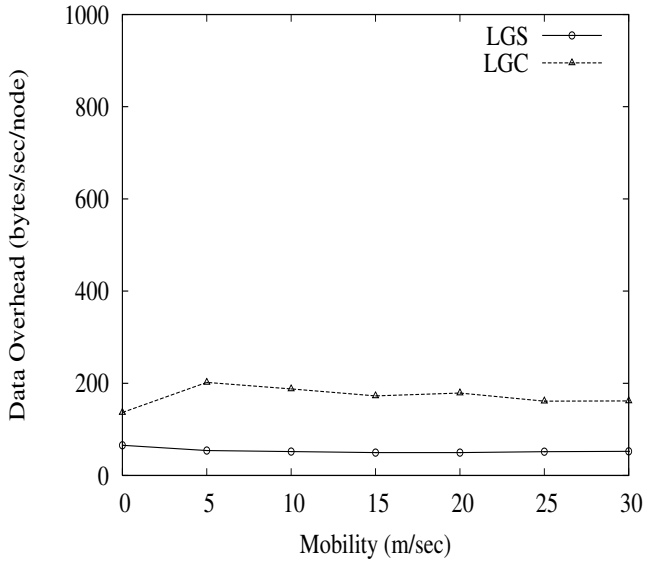


Figure 6.6: Data overhead (bytes)

location updates that are flooded across the network, and for LGC, these are location updates, core maintenance and location maintenance packets that the protocol uses to create and maintain the local delivery tree and the multicast core. The overhead of both protocols increase with increasing node mobility due to increasing rate of sending control packets. As expected, LGS control overhead increases at a much faster rate than LGC with increasing node mobility. The initial drop in the overhead in LGS can be explained by the reduced average number of neighbors with increasing average velocity, which helps the broadcast storm problem. It can be clearly seen that broadcast flooding of control packets leads to an excessively high overhead, as compared to unicasting using only a subset of nodes in the network. Thus, in terms of higher throughout and low overhead, LGC shows superior performance over the LGS protocol

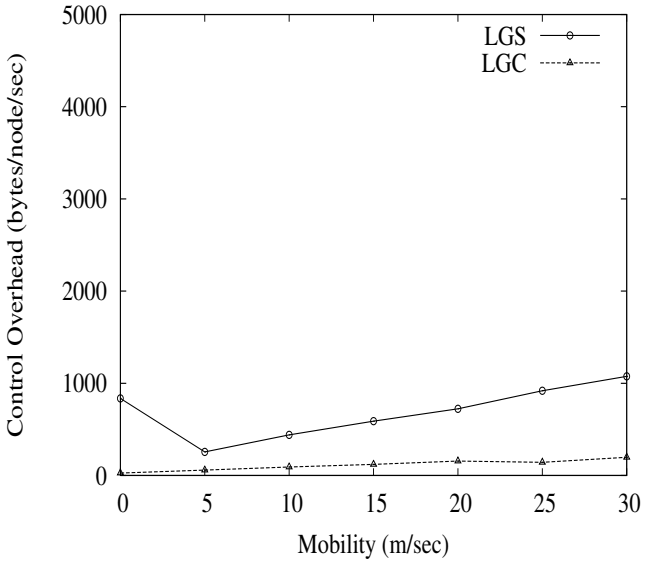


Figure 6.7: Control overhead (bytes)

6.5 Summary

In this chapter, we have described an extension to the unicast location management problem to support efficient location based multicasting in large ad hoc networks. Literature describes a few protocols for location based multicasting, but most of the existing work assume that the node locations are flooded across the network in order to compute the multicast delivery tree in a distributed fashion. Flooding may be not be a suitable protocol for disseminating location information in a bandwidth constrained environment, and hence we have introduced a novel protocol called Location Guided Core (LGC) that integrates location management and multicast tree construction into a single framework. The protocol is purely distributed and localized, and we have shown that the average per group overhead for the protocol increases only as $O(vN^{\frac{4}{3}})$ asymptotically, which is a sub-linear reduction over flooding. Simulations corroborate our analysis, and we show that LGC outperforms the Location Guided Steiner protocol, an existing

location based multicast scheme that uses flooding to disseminate multicast members' location information.

Chapter 7

Concluding Remarks and Future Work

This dissertation has addressed the challenging and fundamental issues related to scalability of routing protocols in the field of mobile ad hoc networks. While current solutions are known to be lacking in scalability with respect to network size and mobility, there has not been considerable effort in the research of topology driven protocols that are localized. However, knowledge of node locations have proved to be useful in designing geographic routing protocols that are simple, localized and which guarantee packet delivery. An major bottleneck in realizing geographic routing is the location management problem, in which the dynamic locations of all nodes have to be distributively managed so that the location of any destination can be found by querying the location management service. The location management problem is the main focus of this thesis, and we have investigated the design of resource efficient location management protocols, which are scalable with respect to both increase in node mobility as well as network size. We conclude this dissertation by summarizing its major contributions and

giving directions for future work.

7.1 Major Contributions

In this research, we have investigated novel location management protocols that operate efficiently in conjunction with geographic routing. Our major contributions lie in the design of efficient algorithms, and analytical mathematical models to effectively analyze the scalability of these protocols.

Novel Location Management Protocols

Our preliminary research in location management resulted in a protocol known as Scalable Location Management (SLALoM) that divides a known terrain into unit regions and delegates certain regions to be location server regions. Nodes located in these regions are responsible for keeping track of other nodes in the network. The optimal division of the terrain such that the cost of location management is minimal is when $N^{\frac{2}{3}}$ unit regions are combined to form second order regions, and a location server region is selected within each second order region. Thus, under random mobility and communication requirements, SLALoM achieves an average asymptotic location management cost of $O(vN^{\frac{4}{3}})$, which improves upon existing location management protocols described in literature. We also provide an optimization to SLALoM known as Efficient Location Forwarding (ELF) to reduce the location update cost incurred due to large multicast updates in SLALoM.

Under a more practical scenario for ad hoc networks in which mobility and commun-

ication requirements are more localized than random, we note that an alternate partitioning of the terrain into a multi-level hierarchy can effectively reduce the location update cost to $O(vN\log N)$, i.e. only logarithmic in the number of nodes in the network. Since location updates form majority of the control overhead as network size and mobility increases, this reduction is surely beneficial for efficiently supporting geographic routing. We also prove that all the three proposed protocols are scalable with respect to network size under a specific analytic framework.

Location Management in Sparse Networks

While the proposed protocols operate in uniform and dense networks, irregularity and sparse networks can cause protocol incorrectness due to the empty server region problem in unit region based location management protocols. We analyze the problem and present a novel proxy enhancement to combat the problem. The proxy enhancement requires that adjacent non-empty regions be discovered accurately, and this introduces a new challenge which we call the Connected Overlay Planar Graph construction problem. There are no algorithms that solve the problem in literature. As another contribution, we propose centralized and distributed algorithms to construct an overlay graph which is both planar and connected. We also prove the correctness of the algorithm and outline a new routing protocol called Grid Traversal Algorithm (GTA) that combines both greedy geographic routing and planar graph face routing on the overlay graph. The proposed routing protocol is expected to perform well compared to face routing algorithms such as GPSR or GFG due to the contrasting spanning properties of the different planar graphs on which each protocols carries out routing.

Location Based Scalable Multicasting

We also investigate location based multicasting in order to enhance the proposed protocols to support multicasting in mobile ad hoc networks. While there are location based multicast protocols described in literature, they may neither be scalable nor reliable with the increase in group membership or number of groups due to the flooding nature of these protocols. We combine the strength of our unicast location management schemes and the idea of Core Base Trees (CBT) to construct a hierarchy consisting of multicast group partitions connected via a location guided core. The partitioning of the group is based on node locations in the terrain, and is more resilient to changes in topology induced by node mobility. We prove that a sub-linear reduction in control overhead can be achieved by our protocol compared to that of flooding.

7.2 Directions for Future Work

Although a significant amount of research work has been accomplished in the area of wireless ad hoc networks, we have identified additional topics based on our research that need further attention. In this section, we list a summary of potential topics that may be pursued in the field of location management and wireless networks.

- **Energy Efficiency:** Although the proposed location management protocols are efficient in the sense that they bound the number of update packets transmitted, HGRID suffers from additional energy expenditure since hierarchical roles of leaders consume more energy than regular nodes. This may not be fair in ad hoc

networks where nodes have restricted battery power for operation. An obvious solution to the problem is cyclic rotation of roles among all the participating unit regions temporally. Alternately, a leader region can be treated as an empty region, and transient data packets can be routed around these regions so as not to burden server nodes with routing. This may have a detrimental effect on the delay performance of data, but may be well worth in terms of network longevity.

- **Properties of Overlay Graphs:** An unique advantage to overlay graphs compared to that of topology driven graphs is that the former is more stable under node mobility. It has been shown that the Gabriel graph and its sub-graph - the Relative Neighborhood graph - although planar, have poor spanning properties. In other words, since long edges in the unit disk graph are removed to maintain the planarity property, this may lead to longer paths between source-destination pairs. On the other hand, the algorithm proposed in Chapter 5 simply tries to remove the least number of edges to make the graph planar, without considering the type of edge. We will study the effect of such edge removal on the spanning property of the overlay graph.

- **Quality of Service in Wireless Networks:** Recent studies have shown that the topology driven shortest path approach to routing in multi-hop wireless networks may not be a good approach as it discounts the link layer congestion and physical layer interference [117]. Thus, QoS routing protocols based on shortest paths fail to meet the guarantees required by the applications. Research in the design of

QoS protocols will benefit from cross layer interaction that takes into account the loss characteristics of the physical layer and link layer delays.

While greedy geographic routing or its planar graph version is localized and efficient for wireless networks, it lacks an inherent flexibility to support QoS routing. GTA on the other hand, routes on the overlay planar graph in which the graph edges represent connectivity between two adjacent unit regions. By appropriately defining link costs of the overlay graph such that it reflects the traffic conditions within an unit region, a better perspective of the traffic flows in the network may be obtained.

- **Wireless Network Security:** An important factor that deters the widespread use of distributed wireless networks is pricing and network security. Any research in building wireless systems must take into account the issues that threaten system reliability at several layers of the protocol stack such as denial-of-service (DoS) and intrusion (application, MAC), route disruption and resource consumption (network), and eavesdropping (PHY/MAC). A direct application of our current work from hierarchical location management is the area of key distribution for wireless sensor networks. Due to computational restrictions and power considerations, private key algorithms are preferred over public key encryption techniques. The problem of key management is to efficiently distribute and revoke keys such that the capture of a set of keys by an attacker does not lead to network failure [118], [119], [120]. We will look into extending our work to design protocols that efficiently solve the key management problem.

Bibliography

[1] W. Lee, "Mobile cellular telecommunications systems," *McGraw-Hill*, 1990.

[2] J. F. Whitehead, "Cellular system design: An emerging engineering discipline," *IEEE Communications Magazine*, vol. 3, no. 1, pp. 8–15, 1986.

[3] D. M. Balston and R. C. V. Macario, "Cellular radio systems," *Artech House*, 1993.

[4] Charles E. Perkins editor, "Ad hoc networking," *Addison-Wesley*, 2001.

[5] Z. J. Haas et al. editors, "Special issue on wireless ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, 1999.

[6] David B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp. 158–163, December 1994.

[7] Jui-Hung Yeh, Jyh-Cheng Chen, and Chi-Chen Lee, "WLAN Standards," *IEEE Potentials Magazine*, vol. 22, no. 4, pp. 16–22, Oct-Nov 2003.

[8] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," *ISO/IEC 8802.11:1999(E)*, 1999.

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cyirci, "Wireless sensor networks: A survey," *Computer Networks (Elsevier) Journal*, vol. 38, no. 4, pp. 393–422, March 2002.

[10] Stefano Basagni et. al. editors, "Mobile ad hoc networking," *Wiley Publishers*, 2004.

[11] E. Royer and C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications Magazine*, vol. 6, no. 2, pp. 46–55, April 1999.

[12] T. W. Chen and M. Gerla, "Global state routing: A new routing scheme for ad-hoc wireless networks," *In Proceedings of IEEE ICC*, June 1998.

[13] S. Murthy and J.J. Garcia-Luna-Aceves, "A Routing Protocol for Packet-Radio Networks," *Proceedings ACM/IEEE MobiCom*, November 1995.

[14] Josch Broch, David B Johnson, and David A Maltz, "Dynamic Source Routing in ad hoc wireless networks," *Mobile Computing, Kluwer Academic Publishers*, pp. 153–181, 1996.

[15] Charles E. Perkins and Elizabeth M. Royer, "Ad hoc On-Demand Distance Vector Routing," *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, February 1999.

[16] M.R. Pearlman and Z.J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1395–1414, August 1999.

[17] Venugopalan Ramasubramanian, Zygmunt J. Hasas, and Emin Gun Sirer, "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks," *Proceedings of the 4th International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, June 2003.

[18] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, and Mahesh K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal Communications Magazine*, vol. 8, no. 1, pp. 16–28, April 2001.

[19] Y. C. Tseng, S. Y. Ni, Y. S. Chen, and J. P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, no. 2/3, 2002.

[20] Ivan A Getting, "The global positioning system," *IEEE Spectrum*, December 1993.

[21] T. Eren, D.K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur, "Rigidity, computation, and randomization in network localization," in *in Proceedings of the conference of the IEEE Communications Society (INFOCOM)*, 2004.

[22] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," *Wireless Networks*, vol. 7, no. 6, pp. 609–616, 2001.

[23] SeungChul Woo and Suresh Singh, "Scalable Routing Protocol for Ad Hoc Networks," *Wireless Networks*, vol. 7, pp. 513–529, January 2001.

[24] Y. Xue, B. Li, and K. Nahrstedt, "A scalable location management scheme in mobile ad-hoc networks," *Proc. of the IEEE Conference on Local Computer Networks*, 2001.

[25] S.H. Bae, S.-J. Lee, W. Su, and M. Gerla, "The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks," *IEEE Network Magazine*, vol. 14, no. 1, pp. 70–77, 2000.

[26] C.-C. Chiang, M. Gerla, and L. Zhang, "Adaptive shared tree multicast in mobile wireless networks," *Proceedings of IEEE Globecom*, 1998.

[27] J.J. Garcia-Luna-Aceves and E. L. Madruga, "The core assisted mesh protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1380–1394, 1999.

[28] Jorjeta G. Jetcheva and David B. Johnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, pp. 33–44, 2001.

[29] M. Garey and D. S. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness," *W. H. Freeman and Co.*, 1979.

[30] Kai Chen and Klara Nahrstedt, "Effective location-guided tree construction algorithms for small group multicast in manet," *Proceedings of the IEEE Infocom*, pp. 1180–1189, 2002.

[31] Tony Ballardie, Paul Francis, and Jon Crowcroft, "Core based trees (CBT)," *Proceedings of the Conference on Communications architectures, protocols and applications*, pp. 85–95, 1993.

[32] S. Ramanathan and Martha Steenstrup, "A survey of routing techniques for mobile communications networks," *Mobile Networks and Applications*, vol. 1, no. 2, pp. 89–104, 1996.

[33] Andrew S. Tannenbaum, "Computer Networks, 4/E," *Prentice Hall Publishers*, 2002.

[34] Charles Perkins and Pravin Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994, pp. 234–244.

[35] T.H. Clausen, G. Hansen, L. Christensen, and G. Behrmann, "The optimized link state routing protocol, evaluation through experiments and simulation," *In*

Proceedings of IEEE Symposium on Wireless Personal Mobile Communications, September 2001.

[36] Josh Broch, David Maltz, David Johnson, Yih-Chun Hu, and Jorjeta Jetcheva, "A Performance comparison of multi-hop wireless Ad-Hoc network routing protocols," *Proceedings ACM/IEEE MobiCom*, pp. 85–97, October 1998.

[37] C-K. Toh, "Associativity-based routing for ad-hoc networks," *Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems*, vol. 4, no. 2, pp. 103–139, March 1997.

[38] R. Dube, C. Rais, K. 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.

[39] Vincent D. Park and M. Scott Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *IEEE Infocom (3)*, 1997, pp. 1405–1413.

[40] P. Johanson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad-hoc networks," *Proceedings ACM/IEEE MobiCom*, pp. 195–206, 1999.

[41] Yih-Chun Hu and David B. Johnson, "Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks," *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[42] Yih-Chun Hu and David B. Johnson, " Implicit Source Routes for On-Demand Ad Hoc Network Routing," *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc),*, pp. 1–10, 2001.

[43] Sung-Ju Lee, Elizabeth M. Belding-Royer, and Charles E. Perkins, "Scalability study of the ad hoc on-demand distance vector routing protocol," *International Journal of Network Management*, vol. 13, no. 2, pp. 97–114, 2003.

[44] Ionut D. Aron and Sandeep K. S. Gupta, "On the scalability of on-demand routing protocols for mobile ad hoc networks: An analytical study," *Journal of Interconnection Networks*, vol. 2, no. 1, pp. 5–29, 2001.

[45] F Kamoun and L Kleinrock, "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks," *Computer Networks*, vol. 3, pp. 337–353, November 1979.

[46] P. F. Tsuchiya , "The landmark hierarchy: a new hierarchy for routing in very large networks," *Applications, Technologies, Architectures,and Protocols for Computer Communication*, pp. 35–42, 1988.

[47] G. Pei, Mario Gerla, and X. Hong, "Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility," *Proceedings of IEEE/ACM MobiHOC 2000*, pp. 11–18, August 2000.

[48] G. Pei and Mario Gerla, "Mobility management for hierarchical wireless networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 6, no. 4, pp. 331–337, 2001.

[49] A. Iwata, C. Chiang, G. Pei, M. Gerla, and T. Chen, "Scalable routing strategies for ad-hoc wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1369–1379, August 1999.

[50] Elizabeth M. Belding-Royer, "Multi-level hierarchies for scalable ad hoc routing," *Wireless Networks*, vol. 9, no. 5, pp. 461–478, September 2003.

[51] Alan D. Amis, Ravi Prakash, T.H.P. Vuong, and D.T. Huynh, "Max-Min D-Cluster Formation in Wireless Ad Hoc Networks," *IEEE Infocom*, vol. 1, no. 7, pp. 32–41, March 2000.

[52] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, September 1997.

[53] A. Banerjee and S. Khuller, "A clustering scheme for hierarchical control in multi-hop wireless networks," *IEEE Infocom*, vol. 2, pp. 1028–1037, 2001.

[54] P Enge and P Misra, "Special Issue on GPS: The Global Positioning System," *Proceedings of the IEEE*, pp. 3–172, January 1997.

[55] Nirupama Bulusu, John Heidemann, and Deborah Estrin, "Gps-less low cost outdoor localization for very small devices," *IEEE Personal Communications*, October 2000.

[56] J. Caffery and G. Stber, "Overview of radiolocation in cdma cellular systems," *IEEE Communications*, vol. 36, no. 4, pp. 38–45, April 1998.

[57] J. H. Reed, T. S. Rappaport, and B. D. Woerner, "Position location using wireless communications on highways of the future," *IEEE Communications*, October 1996.

[58] Jeffrey H. Reed, Kevin J. Krizman, Brian D. Woerner, and Theodore S. Rappaport, "An overview of the challenges and progress in meeting the e-911 requirement for location service," *IEEE Communications*, vol. 36, no. 4, pp. 30–37, April 1998.

[59] Paramvir Bahl and V. N. Padmanabhan, "Radar: An in-building rf based user location and tracking system," *IEEE Infocom*, March 2000.

[60] A. Chakraborty and H. Balakrishnan, "The cricket location support system," *ACM Mobicom*, August 2000.

[61] L. Doherty, L. El Ghaoui, and K. Pister, "Convex Position Estimation in Wireless Sensor Networks," *IEEE Infocom*, April 2001.

[62] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) using AoA," *IEEE Infocom*, 2003.

[63] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes in large scale sensor networks," in *ACM/IEEE International Conference on Mobile Computing and Networkin(MOBICOM)*, 2003.

[64] Krishna Chintalapudi, Ramesh Govindan, Gaurav Sukhatme, and Amit Dhariwal, "Ad-hoc localization using ranging and sectoring," in *in Proceedings of the conference of the IEEE Communications Society (INFOCOM)*, 2004.

[65] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *ACM/IEEE International Conference on Mobile Computing and Networking(MOBICOM)*, 2001, pp. 166–179.

[66] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network Magazine*, , no. 15(6), pp. 30–39, November 2001.

[67] Ivan Stojmenovic, "Position based routing in ad hoc networks," *IEEE Commmunications Magazine*, vol. 40, no. 7, pp. 128–134, July 2002.

[68] Gregory G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," *ISI Research Report ISU/RR-87-180*, March 1987.

[69] Hideaki Takagi and Leonard Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 246–257, 1984.

[70] T.C. Hou and V.O.K. Li, "Transmission range control in multihop packet radio networks," *IEEE Transactions on Communications*, vol. 1, no. 34, pp. 38–44, 1986.

[71] B Karp and H T Kung, "GPSR : Greedy perimeter stateless routing for wireless networks," *Proceedings ACM/IEEE MobiCom*, pp. 243–254, August 2000.

[72] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *IEEE Personal Communications*, vol. 8, pp. 48–57, February 2001.

[73] Prosenjit K. Bose, Luc Devroye, W. Evans, and David Kirkpatrick, "On the spanning ratio of gabriel graphs and beta-skeletons," *Proc. 5th Latin American Symp. Theoretical Informatics, Lecture Notes in Computer Science 2286, Springer-Verlag*, pp. 479–493, 2002.

[74] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," *in Proceedings of the fourth ACM international symposium on Mobile and ad hoc networking and computing (MobiHoc '03)*, pp. 267–278, June 2003.

[75] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger, "Asymptotically optimal geometric mobile ad-hoc routing," *in Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, September 2002.

[76] S. Datta, I. Stojmenovic, and J. Wu, "Internal node and shortcut based routing with guaranteed delivery in wireless networks," *in Proceedings of the 21st International Conference on Distributed Computing Systems*, 2001.

[77] Sami Tabbane, "Location Management Methods for Third-Generation Mobile Systems," *IEEE Communication Magazine*, pp. 72–84, August 1997.

[78] Young Bae Ko and Nitin H Vaidya, "Location-Aided Routing (LAR) in mobile Ad-Hoc networks," *IEEE Infocom*, pp. 66–75, October 1998.

[79] S Basagni, I Chlamtac, V R Syrotiuk, and B A Woodward, "A distance routing effect algorithm for mobility (DREAM)," *Proceedings ACM/IEEE MobiCom*, pp. 76–84, 1998.

[80] Zygmunt J Haas and Ben Liang, "Ad hoc mobility management with uniform quorum systems," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 228–240, April 1999.

[81] Ivan Stojmenovic, "Home agent based location update and destination search schemes in ad hoc wireless networks," *Advances in Information Science and Soft Computing, WSEAS Press*, pp. 6–11, 2002.

[82] Jinyang Li, John Janotti, Douglas S J De Couto, David Karger, and Robert Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," *Proceedings ACM/IEEE MobiCom*, pp. 120–130, August 2000.

[83] Christine Cheng, Howard Lemberg, Sumesh Philip, Eric van den Berg, and Tao Zhang, "SLALoM: A Scalable Location Management Scheme for Large Mobile Ad-hoc Networks," *Proceedings of IEEE Wireless Communications and Networking Conference*, March 2002.

[84] Sumesh Philip and Chunming Qiao, "Elf: Efficient location forwarding in ad hoc networks," *Proceedings of the IEEE Globecom*, 2003.

[85] Sumesh J. Philip and Chunming Qiao, "Hierarchical grid location management for large wireless ad hoc networks," *ACM SIGMOBILE Mobile Computing and Communications Review(MC2R)*, vol. 7, no. 3, pp. 33–34, 2003.

[86] Sumesh Philip, Joy Ghosh, and Chunming Qiao, "Performance analysis of a multilevel hierarchical location management protocol for ad hoc networks," *Elsevier Computer Communications*, vol. 8, no. 10, pp. 1110–1122, 2005.

[87] Cesar Santivanez, Bruce McDonald, Ionnis Stavrakakis, and Ram Ramanathan, "On the Scalability of Ad hoc Routing Protocols," *IEEE Infocom*, June 2002.

[88] Xiang Zeng, Rajive Bagrodia, and Mario Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," *Proceedings of the 12th Workshop on Parallel and Distributed Simulations – PADS '98*, May 1998.

[89] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A geographic hash table for data-centric storage in sensornets," *In Pro-*

ceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), September 2002.

[90] Hannes Frey and Daniel Goergen, "Planar graph routing on geographical clusters," *Ad Hoc Networks*, Article in Press.

[91] K.R. Gabriel and R.R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Zoology*, vol. 18, pp. 259–278, 1969.

[92] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric spanner for routing in mobile networks," *In Proceedings of the second ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pp. 45–55, 2001.

[93] Brad Karp, "Greedy perimeter stateless routing (gpsr)," *http://www.icir.org/bkarp/gpsr/gpsr.html*.

[94] Toussaint G.T, "The relative neighborhood graph of a finite planar set," *Pattern Recognition*, vol. 12, pp. 261–268, 1980.

[95] Ding-Zhu Du, Bing Lu, Hung Q. Ngo, and Panos Pardalos, "The Steiner Tree Problem," *Encyclopedia of Optimization, Kluwer Academic Publisher*, 2001.

[96] Kadaba Bharath-Kumar and Jefferey M. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Transactions on Communications*, vol. 31, no. 3, pp. 343–351, 1983.

[97] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner Problem in Graphs," *Math Japonica*, vol. 24, no. 6, pp. 573–577, 1980.

[98] Y. K. Dalal and R. M. Metcalfe, "Reverse Path Forwarding of broadcast packets," *Communications of the ACM*, vol. 21, no. 12, pp. 1040–1048, 1978.

[99] Stephen E. Deering and David R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 85–110, 1990.

[100] John Moy, "Multicast Routing Extensions for OSPF," *Communications of the ACM*, vol. 37, no. 8, pp. 61–66, 1994.

[101] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," *RFC 1075*, 1988.

[102] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching gung Liu, and Liming Wei, "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 153–162, 1996.

[103] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-demand multicast routing protocol," *Proceedings of IEEE WCNC'99*, pp. 1298–1302, 1999.

[104] Elizabeth M. Royer and Charles E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol.," *Proceedings of IEEE/ACM MobiCom*, pp. 207–218, 1999.

[105] C. W. Wu and Y. C. Tay, "AMRIS: a multicast protocol for ad hoc wireless networks," *Military Communications Conference Proceedings (MILCOM 1999)*, vol. 1, pp. 25–29, 1999.

[106] C-K. Toh, Guillermo Guichal, and Santithorn Bunchua, "ABAM : On-Demand Associat ivity-Based Multicast Routing for Ad Hoc Mobile Networks," *Proceedings of IEEE Vehicular Technology Conference*, pp. 987–993, 2000.

[107] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan, "MCEDAR: multicast core-extraction distributed ad hoc routing," *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 3, pp. 1313–1317, 1999.

[108] Lusheng Ji and Scott M. Corson, "Differential destination multicast-a MANET multicast routing protocol for small groups," *IEEE Infocom*, vol. 2, pp. 1192–1201, 2001.

[109] Chao Gui and Prasant Mohapatra, "Scalable multicasting in mobile ad hoc networks," *IEEE Infocom*, vol. 3, pp. 2119–2129, 2004.

[110] Chao Gui and Prasant Mohapatra, "Efficient overlay multicast for mobile ad hoc networks," *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2, pp. 1118–1123, 2003.

[111] Young-Bae Ko and N. H. Vaidya, "Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms," *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 99)*, 1999.

[112] Stefano Basagni, Imrich Chlamtac, and Violet R. Syrotiuk, "Location aware, dependable multicast for mobile ad hoc networks," *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 99)*, vol. 36, no. 5-6, pp. 659–670, 2001.

[113] Martin Mauve, Jorg Widmer Holger Fubler, and Thomas Lang, "Position-based multicast routing for mobile Ad-hoc networks," *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, vol. 7, no. 3, pp. 53–55, 2003.

[114] Matthias Transier, Holger Fubler, Jorg Widmer, Martin Mauve, and Wolfgang Effelsberg, "Scalable Position-Based Multicast for Mobile Ad-hoc Networks," *Proceedings of the First International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWim 2004)*, 2004.

[115] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," *IEEE Infocom*, 2003.

[116] W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model," *IEEE Transactions on Mobile Computing*, vol. 3, no. 1, pp. 99–108, 2004.

[117] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03)*, September 2003.

[118] L. Eschenauer and V. Gligor, "A Key Management Scheme for Distributed Sensor Networks," *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 41–47, 2002.

[119] Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen, and Pramod K. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *IEEE Infocom*, vol. 1, 2004.

[120] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," *In Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, pp. 52–61, 2003.