A Framework for Multi-Mode Routing in Wireless Ad Hoc Networks: Theoretical and Practical Aspects of Scalability and Dynamic Adaptation to Varying Network Size, Traffic and Mobility Patterns

A Thesis Dissertation Presented
by

César A. Santiváñez

to
The Electrical and Computer Engineering Department

in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

in the field of
Electrical Engineering

Northeastern University
Boston, Massachusetts

November 2001

NORTHEASTERN UNIVERSITY

Graduate School of Engineering

Thesis Title: A Framework for Multi-Mode Routing in Wireless Ad Hoc Networks: Theoretical and Practical Aspects of Scalability and Dynamic Adaptation to Varying Network Size, Traffic and Mobility Patterns.
Author: César A. Santiváñez.
Department: Electrical and Computer Engineering.

Approved for Thesis Requirements of the Doctor of Philosophy Degree

Thesis Advisor: Prof. A. Bruce McDonald — Date

Thesis Reader: Prof. Elias Manolakos — Date

Thesis Reader: Prof. David Brady — Date

Thesis Reader: Prof. Ioannis Stavrakakis
(external reader) — Date

Thesis Reader: Dr. Ram Ramanathan
(external reader) — Date

Chairman of Department — Date

Graduate School Notified of Acceptance:

Director of the Graduate School — Date

Preface

This dissertation presents a novel framework for the development of mechanisms that support scalable routing in wireless ad hoc networks—a class of network architecture characterized by its dynamic topology, randomly varying channel quality and limited resources. In general, the nodes of an ad hoc network are mobile and rely entirely on wireless transmission without a fixed infrastructure or dedicated communications devices. Consequently, network control functions including routing and traffic management must be performed by the nodes themselves. Hence, an ad hoc network effectively consists of a set of mobile wireless routers participating in adaptive routing algorithms that must be responsive enough to meet application requirements without consuming excessive energy and bandwidth, or leading to system instability.

Despite the diversity of conditions that may be encountered in ad hoc networks, the research community has lacked sufficient depth of understanding of the limitations of current ad hoc routing protocols, and their dependence on interacting system parameters and environmental factors. As such, the majority of research focused on ad hoc network routing protocol design has resulted in strategies that are limited in practical terms to a narrow range of environments. The success of the technology, however, depends on the development of scalable routing algorithms that are capable of adapting efficiently to substantially greater variation in network size and range, as well as significant temporal and spatial variation in traffic and mobility patterns.

The fundamental tenet of this research is based on the observation that engaging in a single ad hoc routing strategy is insufficient for effectively adapting to the wide range of environments present in such networks. Instead, a multi-mode routing strategy should be developed that applies the "mode" determined to be most effective at a given point in time and for the appropriate portion of the network. To achieve this objective, however, would require a deeper knowledge of the tradeoffs inherent in routing algorithm design and performance in dynamically changing environments than that available through the existing literature.

The research results presented in this dissertation represent a significant theoretical and practical contribution that advances the state-of-the-art in the field of ad hoc networks by addressing several difficult challenges that are encountered in the synthesis of a multi-mode routing framework. Specifically, a comprehensive analysis of simplified variants of “limited” link-state routing in the context of scalability demonstrates the impact of link-state dissemination policies on network performance. The results lead to the derivation of a novel algorithm, namely Hazy Sighted Link-State (HSLS) that is shown to provide an optimal balance between control overhead and route optimality. Next, an in-depth theoretical analysis of the asymptotic performance of a representative set of the fundamental classes of ad hoc routing protocols is presented with respect to traffic, mobility and size. The analytical results, which are the first of their kind, provide insight into the fundamental properties and limitations of ad hoc networks (limits, trade-offs, and behavior), and demonstrate the feasibility of utilizing HSLS, an easy-to-implement, low-overhead alternative to complex dynamic hierarchical schemes in order to achieve scalability.

The practical contribution that completes the basic framework focuses on two novel enabling algorithms, namely, the Limited Link State (LLS) and Self-Organizing (SO) algorithms. Together, these algorithms can be combined into an effective multi-mode routing strategy that adapts to any ad hoc network condition—from small networks consisting of low mobility nodes to large networks of highly mobile nodes to heterogeneous networks consisting of different classes of users—in order to make the most efficient routing decisions. These algorithms employ novel metrics that capture the mobility and traffic pattern of a subsets of the network. Based on the multi-mode framework, once the current local structure of the network is determined utilizing these metrics, the 'mode' of operation can be shifted on a localized basis to that which is best suited for the environment.

Dedication

To Ciro and Estela, my parents, and to Rosa, my wife.

Acknowledgments

I am very grateful to my former and current advisors, Prof. Ioannis Stavrakakis and Prof. A. Bruce McDonald, for their guidance and support during this research. This work would not have been possible without them. I would also like to thank Dr. Ram Ramanathan for his valuable comments and observations during the development of this work. Several of the concepts presented in this dissertation are the direct result of fruitful and stimulating discussions with him. My sincere thanks also go to the other two members of my thesis committee: Prof. Elias Manolakos and Prof. David Brady for their time and advice on this research, and to Mr. Mustafa Ozdemir for his assistance in obtaining the simulation results presented in Chapter 5. Finally, I would like to express my appreciation for Prof. Bahram Shafai and Prof. Elias Manolakos, for their kindness and help throughout my Ph.D. studies, especially during the period when my former advisor, Prof. Stavrakakis, was not affiliated with Northeastern University any longer.

Contents

List of Figures

Chapter 1

Introduction

Wireless networks can be classified according to their organization and structure as either infrastructure-based or ad hoc networks. As an example of an infrastructure-based network consider the cellular system, wherein the mobile nodes access a fixed network through well established access points, namely, base stations. Node mobility is handled by means of a location management agent whose function is to discover the fixed network's access point to reach any mobile destination. Once the current access point is discovered, routing is achieved by the control algorithms of the fixed network; that is, in an infrastructure-based network, the routing functionalities are executed in the fixed network only and the mobile nodes have neither routing nor switching capabilities.

The fundamental characteristic of a wireless ad hoc network, distinguishing it from a cellular system, is that it does not assume any well defined fixed network infrastructure. If such an underlying infrastructure exists, it will not be known *a priori* and must be capable of changing dynamically. An ad hoc network may be characterized as a rapidly deployable association of mobile nodes whose mobility patterns may be unpredictable or probabilistic at best, in an environment lacking a supporting infrastructure. In order to support arbitrary communications between end-points in an ad hoc network a subset of the nodes must have routing capabilities. Such nodes are essentially treated as mobile base stations when engaged in the communication between two end-points.

Node mobility may vary widely among the nodes—some nodes may be highly mobile, whereas others may be very slow moving or even stationary. Relative mobility patterns or group movement may relate to specific shared goals, or may be arbitrary, not involving *related* nodes. Heterogeneity is also expected in future ad hoc networks. Some nodes may have only one interface, whereas others may have several interfaces, allowing them to become gateways between different classes of subnetworks. The available transmission bandwidth, processing capabilities and power constraints may also vary significantly from one node to another.

Ad hoc networks are useful whenever and wherever it is impossible, impractical, or too expensive to build, maintain or enhance network infrastructure. Applications for ad hoc networks range from rapidly deployable *instant infrastructure* networks for military and civil operations, to networks of intelligent sensor devices. Sensors are typically power constrained and may be required to operate under extreme conditions for long periods of time without intervention. Ad hoc networks may also be utilized commercially to increase the capacity, range and quality-of-service (QoS) support of infrastructured wireless networks. They may extend network coverage into shadow areas or increase the total throughput in a dense cell (spatial reuse) by allowing the nodes to forward packets directly in a multi-hop fashion. In each of these cases what characterizes the network is not a lack of structure, but the existence of an instantaneous and dynamic structure. The network dynamics may relate directly to a common task among network users, as would be the case in a military operation. However, ad hoc networks may also support unrelated users as a virtual wireless Internet. In this case the pattern of the network structure and its dynamics may be very unpredictable and is not related to any common goal.

In summary, ad hoc networks are rapidly varying, bandwidth and energy constrained networks supporting a wide range of users with different characteristics (mobility, traffic patterns, etc.) and communications requirements. They may consist entirely of nodes that function as routers, or they may be hybrid networks with both routing and non-routing nodes. Furthermore, an ad hoc network may require interconnection with a wired network. All of these properties make the problem of effectively routing and supporting QoS in ad hoc networks very challenging.

The traditional approaches to routing in fixed packet-switched networks are based upon two fundamental classes of routing algorithms: link-state (LS) and distance vector (DV). In general, these approaches are well suited to adaptive routing in slowly changing networks. However, routing in large networks requires complex, fixed hierarchies. Due to slow convergence times, routing table loops and high communications overhead in large networks, basic LS and DV approaches would fail in a highly dynamic ad hoc network environment. A significant problem is that these algorithms would tend to consume a significant portion of the available bandwidth to maintain routes that may no longer be valid by the time a node has made a new routing decision for a given destination. Traditional hierarchical approaches can not be effective in this changing environment as they are designed off-line and remain fixed.

Despite the many challenges associated with routing in ad hoc networks, several protocols have been proposed that appear to be effective in small ad hoc networks with limited mobility. However, the ability of those protocols to adapt or maintain stability rapidly fades as the network size or the mobility level increase, or the available bandwidth or energy decrease. The typical approach in designing most of the early ad hoc network routing protocols was based on consideration of a narrow range of possible environments. Hence, failure to adapt efficiently or effectively to the wide range of conditions expected in ad hoc networks is universal.

The fundamental tenet of this research is based on the observation that engaging in a single ad hoc routing strategy is insufficient for effectively adapting to the wide range of environments present in such networks. Such diverse environments cannot be effectively taken into consideration by simply adjusting the parameters of a single protocol. Instead, a unified multi-mode routing strategy should be developed that applies the “mode” determined to be most effective at a given point in time and for the appropriate portion of the network. This determination is based on a structure-learning/engaging capability that is integral to the multi-mode approach. To achieve this objective, however, would require a deeper knowledge of the tradeoffs inherent in routing algorithm’s design and performance in dynamically changing environments than that available through the existing literature.

This dissertation presents a novel framework for achieving scalable routing in ad hoc networks with widely varying characteristics based on a multi-mode strategy. An integral part of the proposed framework are the two structure-learning/engaging algorithms, namely, the Limited Link State (LLS) and Self-Organizing (SO) algorithms. A central task of such structure-learning/engaging algorithms is to identify the key attributes of the network to be used in defining the state of the ad hoc network. Based on these attributes, the network's current, localized structure can be extracted and the appropriate "mode" of operation can be engaged to provide an effective routing solution.

The development of the proposed multi-mode framework made evident the current lack of understanding of the fundamental limits and dependencies present on ad hoc networks. Such an understanding was required in order to develop the necessary principles for constructing the multi-mode framework. Thus, to facilitate the design of a multi-mode routing protocol, a very challenging theoretical analysis was undertaken that represents a significant and long awaited contribution. This contribution extends the current state-of-the art in the field by studying those limits and dependencies within the context of the structure-learning/engaging algorithms, which are treated separately as stand alone algorithms. In particular, simplified variants of the LLS algorithm are studied in the context of scalability. A better understanding of the impact of limited link state dissemination on the performance of a homogeneous ad hoc network is obtained. This understanding has lead to the derivation of a novel algorithm, namely the Hazy Sighted Link State (HSLS) routing that presents the optimal trade-off between control overhead and route suboptimality. HSLS also presents excellent scalability properties.

In addition, HSLS's scalability properties with respect to network size, mobility and traffic are studied and compared to each one of the fundamental class of ad hoc routing algorithms. This study constitutes the first theoretical attempt to study ad hoc network routing protocols, and provides a valuable insight into the fundamental properties and limits of an ad hoc network. The results presented in this dissertation do not only improve understanding of ad hoc networks in general, but also establish HSLS as a low cost solution for routing in ad hoc networks with better properties than more complex approaches, for example hierarchical routing.

Regarding the Self-Organizing algorithm, as a starting point for its study an initial version of such an algorithm is proposed, specified and implemented. Although the specified protocol represents only an instantiation of the self-organizing principle, it helps to understand the theoretical trade-offs inherent in such an algorithm, as well as its implementation complexity. Future research may produce better or even optimal self-organizing algorithms.

In conclusion, in this dissertation the two main mechanisms identified as enablers of a multi-mode routing approach have been studied separately. Furthermore, a novel, easy-to-implement protocol – HSLS – which presents better scalability properties than traditional hierarchical approaches was developed and analyzed. This theoretical analysis represents the first attempt of its kind within the context of ad hoc network routing. Furthermore, the first non-hierarchical protocol that attempts to learn and exploit group mobility to achieve improved routing has been developed and presented. As a consequence of this research, a better understanding of the mechanisms required to implement the multi-mode routing approach has been obtained, and researchers are significantly closer to the goal of designing and implementing such a multi-mode protocol. In this context, this dissertation significantly improves the state-of-the-art in the understanding of ad hoc networks.

1.1 Previous Work

Routing protocols for ad hoc networks have been the subject of extensive research over the past several years and a substantial body of research appears in the current literature. Recently, practical applications such as intelligent sensor networks have focused attention on understanding the issues and tradeoffs involved with network scalability. An important question that arises is *which routing protocol scales the best*? The typical answer is: *it depends*. Unfortunately, the networking community lacks a tenet for understanding the fundamental properties and limitations of ad hoc networks. Hence, a fundamental understanding of *what* scalability depends on, and *how* is currently lacking.

One reason for this shortcoming is the lack of research aimed at developing general principles and analytical models. Scalability and other performance aspects of ad hoc routing protocols have predominantly been studied using simulations and not theoretical analysis. These simulation results, although extremely useful, tend to be limited in their scope to the particular scenarios simulated, and fail to provide a deeper understanding of the limitations of the protocols and their dependence on system parameters and environments beyond the scope of the simulation. The lack of much needed theoretical work in this subject is due, in the author's view, in part to the lack of a common platform to base theoretical comparisons on, and in part to the abstruse nature of the problem.

This subsection presents a brief summary of some of the more important or fundamental research from the literature. This is intended to provide the reader with an idea of past and current trends in ad hoc network routing research. It is not intended to be a comprehensive study but instead focus on work on adaptation to dynamically changing network scenarios. Hence, it focuses primarily on algorithms that attempt to learn and exploit mobility and traffic patterns in order to scale with size, mobility and traffic. Subsequent chapters, while addressing particular issues and mechanisms, will expand the current literature review as necessary within the context of the discussion.

Routing protocols, in general, may be classified as either proactive or reactive. Proactive protocols attempt to find a route before it is needed, as for example the Standard Link State (SLS) and Standard Distance Vector (SDV) protocols (see [1, 2, 100]). Reactive protocols, on the other hand, attempt to find routes only when they are needed. Reactive protocols (also called on-demand) invoke some route discovery procedure when a packet needs to be transmitted and no route is currently available in the source node's route cache.

Conventional routing protocols in fixed networks almost exclusively implement proactive approaches. These have been extensively investigated. The family of Link-State protocols has been shown to provide loop-free routes with limited convergence time so long as the network is relatively stable and not too large. Large networks must use hierarchical techniques to avoid excessive latency which can leave a network in an un-converged state, thus, leaving the routing tables susceptible to loops. Examples of Link-State based protocols that have been implemented in large fixed networks are the 'new' ARPANET [3], the IS-IS [4], and the OSPF [5] protocols. The most important Distance vector protocols are the 'old' Arpanet [6], RIP [7], and a number of related algorithms designed to reduce overhead, speed convergence and reduce or eliminate the possibility of routing loops. These include: link-vector, path-vector, path-finding and diffusing computation update protocols.

The DARPA's Packet Radio Network (PRNET) project [20, 21] and its successor the Survivable Adaptive Networks (SURAN) project [22, 23, 24] have lead to significant research in the area of packet radio (ad hoc) networks. Several approaches to link-layer and routing in small [25, 26, 27, 28, 29, 30, 31, 32, 33, 34] and large [35, 36, 37, 38, 39, 40, 41, 42] packet radio networks have been considered. The DARPA research is focused on a military scenario where low probability of interception and other military constraints are as important as routing efficiency.

DARPA's research has established the feasibility of packet radio networks, but further research is needed to improve the routing performance in highly dynamic, bandwidth-constrained environments. It has also been shown that in order to efficiently route packets in an ad hoc network, both the link-layer and the network-layer should interact closely. For example, an adaptive power gain [32, 33] and a receiver-directed transmission [34] functions have a notable impact on the routing protocol design. Similarly, a MAC (Medium-Access-Control) protocol that supports reliable broadcast could improve the performance of route discovery algorithms. MACs for ad hoc networks have been the focus of continued research as may be seen from the work in [85, 86, 87, 88, 89, 90, 91].

When standard Distance Vector algorithms are applied to mobile networks they suffer from slow convergence time, looping and instability. On the other hand they produce lower routing overhead than link-state approaches. The research initiated in the DARPA project and others that followed has led to variations to the Distance Vector protocol that are loop-free [12, 13, 14, 15, 16, 17] and eliminate some of the other problems associated with traditional distance vector approaches. These approaches, however, do not scale well to large, highly dynamic networks, largely due to the fact that they require significant internodal coordination over several hops to ensure erasure of stale information, and require the frequent exchange of complete routing tables between neighbors.

Over the past several years significant effort has been concentrated on the development of routing protocols for civilian ad hoc networks and a wide array of new protocols have been proposed for different networking environments [57]-[76]. The Amateur radio community has been working on routing in wireless networks of mobile hosts [43, 44, 45, 46] and the IEEE has begun to consider a network of mobile hosts with the ability to perform switching functions in its standardization efforts. It was precisely the IEEE 802.11 subcommittee that adopted the term *Ad Hoc* network [89]—although they were not envisioning multi-hop networks, merely peer-to-peer wireless without the aide of a base-station.

The Internet community is also looking into the problem of routing in multi-hop wireless networks. Widespread interest in *mobile mesh networking*, later referred to as ad-hoc networks, started with the formation of a *Birds-of-a-feather* (BOF) session in a 1995 Internet Engineering Task Force (IETF) conference. Early discussions centered around military tactical networks, satellite networks and wearable computer networks, with specific concerns being raised relative to adaptation of existing routing protocols to support IP networking in such highly dynamic environments. By 1996 this work had evolved into the Mobile Ad-Hoc Network (MANET) BOF, and finally to the charter of the MANET working group (WG) of the IETF in 1997.

The task of the Mobile Ad hoc Network (MANET) working group [47] is to specify standard interfaces and protocols for support of IP-based internetworking over ad-hoc networks. The group effort is focused on a highly dynamic bandwidth-constrained environment [48, 50] and the majority of their proposed protocols employ reactive approaches that rely on a route discovery and maintenance procedure. Dynamic Source Routing (DSR) [53, 64], Ad hoc On Demand distance Vector (AODV) [54, 61], Temporally-Ordered Routing Algorithm (TORA) [55, 63], Cluster Based Routing Protocol (CBRP) [59], and Zone Routing Protocol (ZRP) [56, 62] all belong to the category of reactive approaches. These protocols take into consideration the instantaneous location of a node only, ignoring mobility patterns and as a consequence their routing decisions are valid for short periods of time only.

CEDAR [58] is a special case among the routing protocols proposed by the MANET network working group. It is focused on supporting QoS requirements (in this case bandwidth) of the applications (as opposed to the best-effort service characteristic of the Internet community's works). CEDAR organizes the network around 'core' nodes which are formed based on the nodes' current position and degree (number of neighbors). This organization does not reflect the mobility pattern of the users. One important characteristic of CEDAR is the employment of 'propagation waves' of link-state information: link-state information of more stable links is propagated deep inside the network whereas information about less stable links is kept local. In this way, CEDAR tries to identify the more stable, higher bandwidth links in the network and if available use them in its routes. Initial results [60] show that CEDAR works fine in low to medium mobility environments and successfully adapts when the network rate of topological change is decreased to even a fixed network. This points to the advantage of using some kind of limited link-state algorithm in order to get some network structure information and adapt to different environments. In CEDAR, however, route discovery is still needed even in the case of a totally stable network. Also, CEDAR's decisions on the propagation depth of the link-state information updates are made in an "ad hoc" fashion, and there is no guarantee that they are close to optimal. Further research has to be conducted in order to apply CEDAR in a large, highly dynamic network.

Associativity-Based Routing (ABR) [65, 66] is a route discovery protocol that attempts to leverage the mobility pattern of the users. ABR assumes that the mobility of a node will follow an associativity rule by which a node will have two 'relative' states with respect to its one-hop neighbors (motion and static) and will be able to detect its current state based on its one-hop-neighborhood information. ABR assumes that the average length of the static period is significant, such that once a node is detected to be 'static' it will remain in this state for a while. ABR prefers routes that involve 'static' nodes. ABR fails to take advantage of nodes whose mobility does not follow the associativity rule ignoring different mobility patterns (e.g., group mobility with nodes in the same group moving close to each other but more than 1 hop away, etc.) . Furthermore, determination of link stability is based on fixed threshold-based timers—the decision is based entirely on history and does not reflect a quantitative model.

More recently, an adaptive clustering mechanism was introduced in [67] where the cluster formation is governed by the probabilistic stability of the paths inside the clusters. The strategy provides mobility-adaptive clusters that are able to grow larger at times or in parts of the network where mobility is low, yet must remain smaller at times or in parts of the network where mobility causes more frequent topological changes. Routing is proactive within clusters and reactive between clusters (hierarchical), hence, the strategy is designed to balance the overall routing overhead based on temporal and spatial dynamics. Path stability is measured according to measured mobility parameters that are applied to an analytically derived model for path-availability that estimates the probability that a path will survive a given interval of time. The analytical model is based on random mobility. Future research is likely to increasingly consider the different dimensions of mobility and its impact in a protocol design.

Proactive protocols for large highly dynamic ad hoc networks have been overlooked since it is presumed that they compute routes that may not be needed. However, this assumption is not based on a rigorous analysis of the probability that a route may be needed. In other words, the traffic pattern has not been considered. For example, if in one region of the network there is a set of servers that is frequently accessed by some large number of nodes distributed throughout the network, maintaining routes toward this region is highly desirable, since the probability that one route be used to reach *any* of the servers is high. As traffic rate and diversity increases, proactive approaches become more attractive.

OLSR [57] is a MANET proposal that considers a proactive algorithm for large ad hoc networks. However, neither the mobility pattern nor the traffic pattern have been considered and it is most likely that OLSR would fail to efficiently route packets when the mobility rate is increased.

Reactive protocols rely on a route discovery procedure that floods the network with a packet (RREQ) probing for the desired destination or an existing route to the destination. Flooding in large ad hoc networks is costly, unreliable, and may induce a high delay before a route is available. Some protocols [58, 56, 73, 74, 75, 76] attempt to reduce the cost of flooding by creating some hierarchy in the network and maintaining it by means of a proactive algorithm. The cost of maintaining the created hierarchy (proactive approach) is usually low and helps to reduce the overall cost while increasing the reliability of flooding. However, the creation of the hierarchy is made independently of the actual network structure (mobility and traffic patterns) and it is based only on the current position. This way, the resulting hierarchy is artificial and does not reflect the peculiarities of the network state. In these protocols, the trade off between the cost of creating a hierarchy and the savings due to reduced flooding cost is addressed in a fixed manner *a priori*; therefore, there is no guarantee that the decision is the best (even good) when applied to a particular environment.

Routing in large ad hoc networks typically involves implementing a complex hierarchical approach, which requires the existence of a location management scheme. Location management schemes for ad hoc networks have been explored in [92, 93] among others.

None of the previously developed protocols considers the different traffic requirements of different users. It may be explained (but not justified) by the intention to respect the boundaries of the OSI's 7-layer model. However, it has been already pointed out that ad hoc networks make this model obsolete, since there is considerable amount of interaction between, for example, the link and the network layers. This has caused the creation of some mixed protocols that include functions of both layers (link and network) together, as for example in [52].

Summarizing the previous work the following may be stated. First, the idea of limiting the propagation depth of the link-state information has been explored in the past for the case of small, stable to low mobility networks. This study has not yet been expanded to large highly mobile environments and there is an absence of formal theory underlying the particular ad hoc implementation. Second, routing in a highly dynamic large ad hoc networks remains a open, challenging problem that has not been verifiably or adequately solved. Initial comparisons among current approaches [79, 78, 80] show that their results depend highly on the assumed environment [79]. The comparisons, however, have been based on simulations and their results, although useful, are limited in their scope to the particular scenarios simulated, and fail to provide a deeper understanding of the limitations of the protocols and their performance dependence on the network parameters outside the range being simulated. The lack of much needed theoretical work in this arena is due in part to the lack of a common platform to base theoretical comparisons on, and in part due to the abstruse nature of the problem. Third, there is no current self-organizing algorithm that takes into account mobility as well as user traffic patterns. Fourth, pure proactive or reactive protocols are suitable for particular environments, and mixed (adaptive) approaches need to be followed when the protocol is expected to operate in diverse environments. Finally, designing a multi-mode routing protocol that works efficiently under *any* scenario (different mobility and traffic patterns) is a pending, challenging task that requires underlying mechanisms to be in place. This dissertation explores two mechanisms that, in combination, enable the design of such a multi-mode routing protocol. Furthermore, in developing the platform for theoretical comparisons, a significantly improved understanding of the fundamental limits of ad hoc networks is obtained.

1.2 Dissertation Outline

The previous section concluded with a summary of outstanding open issues in the area of routing for ad hoc networks. These issues are listed in inverse order of complexity and scope. However, to resolve this issues, a top-down approach must be followed. Specifically, first the relevant elements for a multi-mode approach must be identified, and a suitable framework must be defined. Based on the framework definition the focus should shift to understanding the particular mechanisms that enable best the desired multi-mode approach. Further research and better understanding of ad hoc network dynamics may subsequently force us to re-study and redefine the aforementioned framework (re-engineering).

The remainder of this dissertation is organized as follows: Chapter 2 begins by presenting an overall framework for the design of a multi-mode routing strategy. The framework identifies two structure-learning/engaging mechanisms – namely the Limited Link State (LLS) and Self-Organizing (SO) algorithms – that will enable the design of a multi-mode routing protocol. The central task of the structure-learning/engaging algorithms is to identify the key attributes of the network to be used in defining the state of the ad hoc network. Based on these attributes, the network's current/localized structure can be extracted and the appropriate "modes" of the protocol engaged to provide for efficient routing.

The framework defined in Chapter 2 focuses on the study of fundamental limits and dependencies of ad hoc networks in the context of enabling a multi-mode routing protocol. To this end, and in order to develop tractable models, the structure-learning/engaging algorithms are studied separately, and treated as stand alone algorithms.

Chapter 3 presents a comprehensive study of simplified variants of the LLS algorithm in the context of scalability. A solid understanding of the impact of limited link state dissemination on the performance of homogeneous ad hoc networks emerges from the analysis. This new understanding leads immediately to the derivation of a novel algorithm, namely Hazy Sighted Link State (HSLS) routing, that presents the optimal trade-off between control overhead and route optimality. Thus, the analysis itself leads to an important contribution. By carefully balancing these two sources of routing degradation, HSLS achieves excellent scalability properties.

In Chapter 4, HSLS scalability properties with respect to network size, mobility and traffic are studied and compared to each fundamental class of ad hoc routing algorithm. This study constitutes the first theoretical study ad hoc network routing protocols of its kind, and provides new and significant insight into the fundamental properties and limits of ad hoc networks. The results of Chapter 4 not only improve our understanding of ad hoc networks in general, but also establish HSLS as a low (implementation) cost solution for routing in ad hoc networks with better properties than those of more complex approaches, as for example hierarchical routing.

Chapter 5 switches the focus from LLS techniques and studies the other structure-learning/engaging algorithm: the SO algorithm. A detailed description of a SO-based protocol – namely SOAP – is provided. SOAP is a stand-alone protocol that represents a particular instantiation of the SO algorithm, and as such represents a practical contribution of this research. However, the main value of SOAP comes from the fact that it illustrates the feasibility of SO algorithms, providing important insight into the trade-offs inherent in such an algorithm as well as its implementation complexity. Finally, Chapter 6 presents a summary of the main results and conclusions of the dissertation along with a brief discussion of future work.

Chapter 2

A Framework for a Multi-Mode Routing Protocol

The objective of this chapter is to present a framework for an ad hoc routing protocol that adapts itself to the present network conditions taking into consideration the mobility levels and patterns, as well as traffic patterns. In order to identify and utilize the network conditions (state information) the multi-mode routing protocol has to rely on some structure-learning/engaging algorithms that extract the network state information (defined in terms of proper metrics) and based on this information it has to engage the proper mode of operation.

The present work identifies parameters (metrics) that may be used to define the state of the network. Based on these metrics, structure-learning/engaging algorithms that extract the network state information and enable the implementation of a multi-mode routing protocol may be developed. As a starting point, two complementary structure-learning/engaging algorithms are discussed in sections 2.3 and 2.4, respectively, along with their associated metrics : the Limited Link State (LLS) and the Self-Organizing (SO) algorithms.

2.1 Routing Considerations for Ad Hoc Networks

In traditional routing protocols (such as the Standard Link-State (SLS) protocol) a message is generated each time a link state changes. Consequently, network nodes are aware of the state of links and can pre-calculate routes to potential destination nodes (proactive routing protocols). The bandwidth overhead (cost) associated with maintaining pre-calculated routes is proportional to the frequency of link-state changes or the rate of topological change. The latter is defined as the average number of link-state changes in the entire network per time unit and it is, thus, proportional to the network size and inversely proportional to the mean time to link failure.

In ad hoc networks, link-state changes are mostly due to user mobility. If the rate of topological change is low (case A), the proactive SLS protocol would still be effective. On the other hand, if the rate of topological change is moderate to high (case B) the SLS protocol would be very inefficient and more effective routing protocols should be employed.

In a diverse ad hoc network with a moderate to high rate of topological change, there would typically be links of low rate of change (high stability or stable) and links of moderate to high rate of change (low stability or unstable). An effective routing protocol for such a network should then be able to :

(A) Discover the stable links, properly propagate their state, and enable nodes to pre-calculate routes to destinations which are reachable by using such stable links.

(B) Provide for an effective mechanism to establish routes to the remaining destination, as well as alternative routes to pre-calculated ones which may be over-utilized and become congested.

An example of an algorithm that supports (A) is the Limited Link-State (LLS) algorithm presented in section 2.3. This algorithm will provide routes toward some destination nodes (even if not necessary) at a low cost (bandwidth overhead).

It should be noticed that the stable paths (along stable links) are expected to be a small portion of the possible paths (stable plus unstable) in a typical ad hoc network. Consequently, these paths may be over-utilized and become congested if no alternative paths are available, leading to network throughput reduction and delay increase. Thus, a mechanism to identify less stable routes to destinations with pre-calculated routes as well as to those without pre-calculated routes needs to be developed.

The traditional approach to identify less stable routes in a highly dynamic ad hoc network is based on route discovery or flooding algorithms, both requiring a potentially large number of broadcasts before delivering the information to the destination. When flooding is engaged, each packet needs to be broadcast over the entire network. Route discovery relies primarily on broadcast search for a route to the destination each time a session is to be initiated. Such a search induces some start-up delay, in addition to consuming bandwidth, unless it is bypassed when a recently utilized route is available (i.e., cached) and selected (at the risk of not being appropriate any longer).

The Self-Organizing (SO) algorithm presented in section 2.4 is an example of an algorithm supporting (B) above, that tries to reduce the number of broadcasts required by the route discovery or flooding algorithms by providing pre-calculated routes *toward* **some destinations that are** *likely* **to be involved in new sessions**. For those routes to be useful, the cost associated with their maintenance should be less than the expected gain of using these routes.

The self-organizing algorithm will base its decision on the mobility as well as traffic patterns of the nodes. The self-organizing algorithm will attempt to choose Reference Nodes (RN) and around them Reference Areas (RA) such that the expected number of new sessions having a destination inside the reference area (Gain, G) be maximized. This gain (G) has to be compared against a threshold (the cost of maintaining the routes) to decide whether it is worth creating routes toward a particular reference area.

If all the nodes are assumed to have the same traffic patterns, then the self-organizing algorithm will attempt to find the mobility pattern of the network. Although it is possible that the mobility pattern of a network be totally random, that is not usually the case. Human mobility, for example, is based on groups (forming clouds) or follows some patterns (streets, highway, searching, etc.). Even automata mobility is shaped by the function they are executing and therefore there is some degree of spatial/temporal correlation. The self-organizing algorithm will attempt to find (or select) the mobility 'leaders' (nodes around which others node move). For example, in networks formed by cars in a highway, the cars in the intermediate position would be the best candidates for mobility 'leaders'. However, node mobility is not the only factor to take into account. Even more important is the traffic pattern of the nodes. There is no need to pre-calculate routes for nodes that are not going to communicate at all, whereas there maybe other nodes that may need to be contacted frequently due to their mission (coordinator, server, etc.). For the latter nodes it should be highly desirable to have routes readily available saving the network from otherwise almost certain broadcasts.

Finally, it was pointed out that a reference area will be created only if it is effective. For networks (or some nodes) with high mobility rate or low traffic demand it may not be effective to create reference areas. To forward packets to those nodes route discovery will be used. Similarly, if the routes toward the destination are invalidated too quickly, or if the traffic per session is low – say one or two packets – to the point that simply flooding the packets is expected to be more effective, then flooding will be used instead of route discovery.

The LLS and SO algorithms motivated and briefly mentioned above will basically help identify the stable routes in the network as well as clouds of nodes (reference areas) which are worth maintaining. That is, such algorithms may be viewed as structure-learning/engaging algorithms. These algorithms are expected to be effective not only in single class ad hoc networks - as implicitly assumed above - but also when multi-class nodes are present forming a multilevel hierarchy. As an example, mobile nodes may have an additional interface (more power demanding) to communicate with a more powerful base station in case they become isolated. Similarly, some nodes may serve as gateways to a fixed network such as the internet. In disaster recovery scenarios, land mobile nodes (forming the horizontal level network) may communicate between each other or some may be equipped with land-air interfaces to communicate with helicopters flying around the area (forming the vertical level network), and may use this interface to reach otherwise isolated land mobile securing network connectivity. In any of the previous cases, the existence of two different levels can be advantageous, and the horizontal-vertical interface is a resource that has to be used wisely (e.g., not to congest it with transmissions that may well be forwarded over the horizontal network). The more general case of multi-class ad hoc networks (with nodes with more than one interface) will be briefly discussed in section 2.5.

2.2 A Multi-Mode Framework

The objective of a multi-mode routing protocol is to adapt itself to the present network conditions taking into consideration the mobility levels and patterns, as well as traffic patterns. In order to identify and utilize the network conditions (state information), the multi-mode routing protocol has to rely on some structure-learning/engaging algorithms that extract the network state information (defined in terms of proper metrics) and, based on it, implement the proper mode of the supported multi-mode routing protocol.

In this dissertation, the novel framework shown in figure 2.1 is followed. This framework proposes that a multi-mode routing protocol – running simultaneously at each node – consists of three elements: two complementing structure-learning/engaging modules that provide network state information to the third module, the *multi-mode routing engine*, which decides the mode to apply based on the state of (parts of) the network.

The *multi-mode routing engine* receives information about mobility events (as, for example, nodes displacement and/or link creation/breakages) as well as traffic events (new session requests, or reception of packets to be forwarded to their destination) and passes this information to the structure-learning/engaging modules. Based on this information as well as exchanges among peer modules in neighboring nodes (for example, Link State Update – LSU – messages), the structure-learning/engaging modules obtain some information that defines the state of the network. This information is then passed to the *multi-mode routing engine*, which uses it to make its

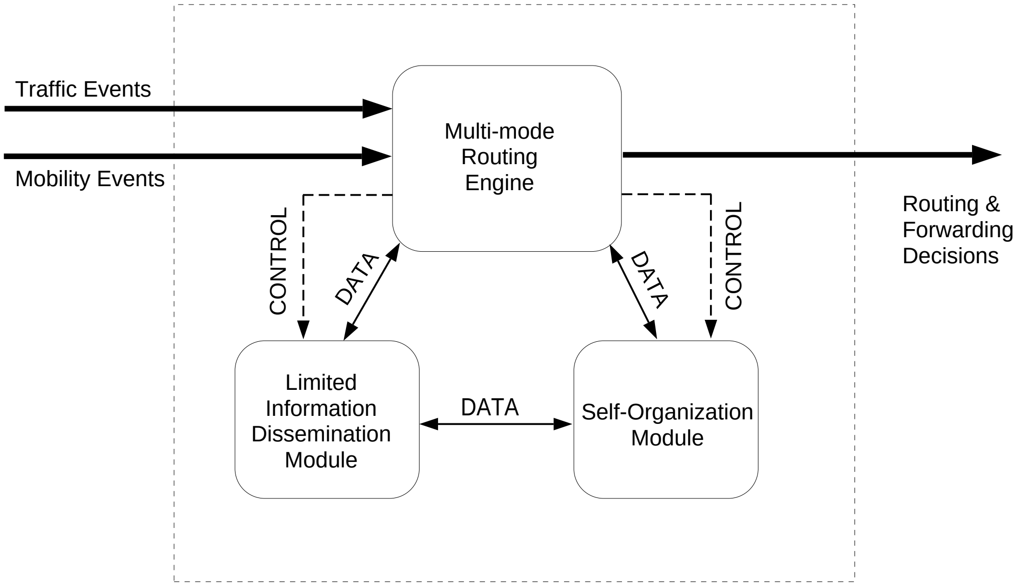

Traffic Events
Mobility Events
Multi-mode Routing Engine
Routing & Forwarding Decisions
CONTROL
DATA
DATA
CONTROL
Limited Information Dissemination Module
DATA
Self-Organization Module

Figure 2.1: A multi-mode routing protocol's framework.

routing mode decisions. The behavior of the structure-learning/engaging modules is not fixed but governed by parameters that are defined by the multi-mode routing engine. Thus, the function of the modules is to provide information to the multimode routing engine, which controls the modules as well as the final routing mode for each particular packet/destination. Note also that there is a feedback between the choice of the module's parameters (controllable to a given extend) and the basic events, since for example link breakages (as seen from a routing protocol) may be triggered due to excessive traffic over a link, and also intermediate packets arrivals – requiring the multi-mode routing engine to forward them to their destination – are correlated to the modules behavior (for example, to the link state information being disseminated to other nodes). Thus, while the control of the multi-mode routing engine over the modules is evident and explicit, the coercion of the modules over the multi-mode routing engine is subtle, giving the erroneous impression that the multi-mode routing engine posseses full control.

The first of the modules, the *limited information dissemination module*, is responsible for implementing the principle : “the closer you are, the more information you have”. This module is in charge of providing detailed information about nodes close by, as well as rough and maybe outdated information about nodes far away. This information may be disseminated in a number of ways. However, to focus the discussion, link state updates were chosen as the bearer of the information. This choice was motivated by the fact that link state-based routing presents several desirable properties as for example : fast convergence, well-understood dynamics, loop freedom, etc. Thus, the **Limited Link-State (LLS) algorithm**, was chosen as the algorithm to be executed by the *limited information dissemination module.* In the remaining of this work, we will refer to the LLS algorithm as the *limited information dissemination module*'s algorithm, but we should keep in mind that alternatively distance vectors or other metrics may be used as information bearer, and therefore, different algorithms may be executed in the *limited information dissemination module.*

The LLS algorithm limits the depth of link-state information propagation, avoiding congesting the network with excessive routing overhead in networks with high rate of topological change (case (B) in section 2.1). Because of the LLS algorithm, every node will have good knowledge about the state of its closer links and stable links far away. This information will be used by the multi-mode routing protocol to construct links toward close destinations and even to destinations far away in the presence of stable links. Also, when the LLS algorithm is applied to a network with low rate of topological change (case (A) in section 2.1) the result would be the same as if standard link-state algorithm were applied. When the LLS algorithm is applied to a network with high rate of topological change (case (B) in section 2.1) nodes will have detailed information for nodes close to them, without incurring excessive network overhead. This information needs to be combined with some rough information about how to route packets to nodes far away. This information may be provided by some complementary algorithm as the Self-Organizing algorithm explained below. In a multi-class network the LLS will try to learn the underlying backbone network, as shaped for example by base stations or gateways to fixed network. In the special case of a two-level network (discussed in section 2.5) the LLS algorithm will try to identify the elements of and interfaces to the (likely) more stable vertical network. LLS is discussed in section 2.3.

The second structure-learning/engaging module, the *self-organizing module* executes the **Self-Organizing (SO)** algorithm presented in section 2.4. The SO algorithm uses metrics that intend to capture the local traffic and mobility information in order to compute a gain and a cost function. Based on these functions, portions of the network are organized in reference areas. By virtue of the SO algorithm, the *self-organization module* provides the multi-mode routing engine with information about links *toward* the reference areas, or with an indication of the highly mobile status of some destination.

The LLS and SO algorithms will run in the *limited information dissemination* and *self-organization* modules, respectively, in a proactive fashion. However, they propagate information to peer modules across the network only when it is justified (cost function evaluation). The information propagated among peer modules is then processed and reported to each node's multi-mode routing engine.

Thanks to the information provided by these structure-learning/engaging modules, each node's multi-mode routing engine will have knowledge of the state of some links (the closer ones and even some links far away that are stable), as well as links towards some regions of the network (reference areas) together with information regarding the location (reference area) of some destinations. Based on this information the multi-mode routing engine may select its 'mode' of operation. Possible decisions include the use of a pre-calculated path of stable links (if available and if stable links are not congested); the use of links toward the destination node's reference area expecting that the packet at some point will find a node with knowledge of routes toward the destination (see Figure 2.3); the use of a query or a broadcast packet to get the destination node's location information; or simply use a combination of route discovery/flooding.

In the next sections, these algorithms (LLS and SO) will be discussed in more detail. In the remaining of this work, it is assumed that the LLS algorithm is executed by the *limited information dissemination module* and the SO is executed by the *self-organizing module* according with the framework presented in this section and in Figure 2.1. Also, it is assumed that the *multi-mode routing engine* will be present, as specified by the aforementioned framework.

2.3 Limited Link-State (LLS) Algorithm

It is well known that link-state algorithms (referred to here as Standard Link-State (SLS)) are effective for stable networks. SLS performance degrades enormously as the network topology becomes dynamic and the network stability decreases.

Nodes in an ad hoc network are expected to have diverse degree of mobility and consequently, some links may be quite stable. Stable links may be associated with nodes with low mobility, with high transmission power or simply with similar mobility pattern (as for example two mobile users walking down the same street or performing a related task). If the link-state information of such stable links is propagated deep into the network, as in Standard Link State (SLS), these links would not produce much bandwidth overhead since they do not require frequent updates (as unstable links do). This is not the case with unstable links, though, and for this reason care should be taken before propagating their link-state changes deep into the network.

The bandwidth overhead due to link-state propagation is typically a function of the rate of *change* of the particular link and the number of nodes that receive the update. If the depth of the propagation of the link-state information is selected to be inversely proportional to the rate of change of the link (related to the stability of the link), then the bandwidth overhead caused by a link will be bounded independently of the link stability (mobility). Alternatively, if the depth of propagation of Link State Updates (LSUs) is bounded – independently of mobility– *and* the rate of generation of LSUs is also bounded, then the resulting bandwidth overhead induced by LSUs will also be bounded. This is the main idea behind the proposed Limited Link-State (LLS) algorithm :

Definition: *The LLS algorithm is a modification of SLS, where the depth of propagation of a link state update (LSU) is a function of time and the past history of the link, attempting to reduce the bandwidth consumed by unstable links' LSUs, and providing closer nodes with more precise link state information.*

It should be noted, however, that the above definition is quite flexible since the particular dependency of the LSU's depth on time and stability is not specified. For example, LLS may be set so that the bandwidth overhead induced by LLS increases *linearly* with the network size, as the bandwidth does. Therefore, the LLS algorithm may be designed in order to not congest the network under any environment. On the other hand, since the propagation depth is limited to a number that is a function of mobility and not of network size, it is possible that in huge networks even almost fixed link updates will not be available to all the nodes but only to a portion of them. The latter may not be an issue since for huge networks it is likely that a complementary approach that facilitates routing (e.g. the SO algorithm) will be employed. Nevertheless, there is a fundamental trade-off between the bandwidth overhead reduction that may be achieved using *agressive* settings for LLS and the loss of routing effectiveness. In order to shed light into this trade-off, simplified variants of LLS are studied in chapters 3 and 4.

In an efficient LLS algorithm, the availability of stable link information to the nodes make it possible for them to detect stable paths over the network if they exist. This way, the nodes will discover some underlying network structure. For example, the existence of some fixed antenas or some high power stations or some highly reliable paths.

LLS's discovering of stable paths allows the forwarding of packets toward certain destinations at a low routing overhead cost (because of infrequent link-state updates). Typically, it will be possible for nodes to forward packets to close by destination nodes, since proper link-state information will be available. In general, the closer to a destination a node is, the more information related to that destination it will have. This property is also used in the proposed Self-Organizing algorithm.

The proposed LLS algorithm is similar to the link-state algorithm in that they both propagate link-state information, but differ in the information being transmitted (metrics being used) and the depth of the transmision. The LLS algorithm proposed here considers three metrics associated with a link : cost, stability, and quality.

The link cost is defined as the ratio of the number of nodes accessing the same associated channel over the available bandwidth. Clearly, the cost will increase with the number of neighbors (more interference) and decrease as the bandwidth increases. The link stability is defined as the average time the link is active. A link is active if it is not detected absent for a period greater that d. Figure 2.2 shows an example of

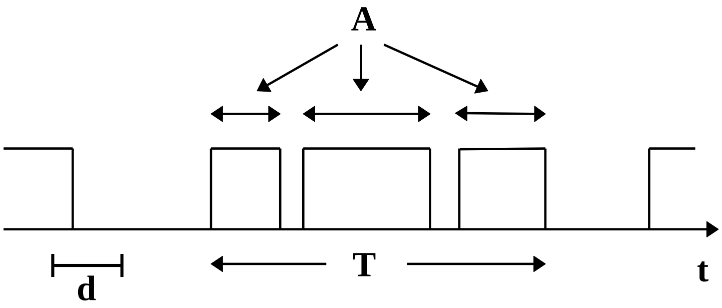


Figure 2.2: Sample realization of link availability

a link availability. In this example the link is active for T seconds. The link quality may be defined to be the fraction of the active time (T) that the link is *actually* available, e.g. A/T in Figure 2.2. The link quality may be regarded as an estimate of the probability that a link is available at a particular time given that the link is in state 'active'. It may be seen that the link stability captures the longer term behavior of the link whereas the shorter term behavior is captured by the link quality.

The link stability will determine how far away the link-state information will be propagated by the limited link-state algorithm. For more stable links the algorithm would propagate the link cost and quality far away whereas for less stable links the link-state information would be transmitted only to the closest nodes. This way, the excessive bandwidth overhead produced by traditional link-state algorithm in highly mobile environments is dramatically reduced (it may even be kept almost constant as indicated earlier, if complementary approaches such as SO are used to forward packets farther away). It should be noted that even in a highly dynamic, large network, the limited link-state algorithm may obtain stable paths (if they exists) at a low bandwidth overhead cost.

2.4 Self-Organizing (SO) Algorithm

This section presents the Self-Organizing (SO) algorithm. The SO algorithm tries to learn the (dynamic) mobility and traffic pattern of the network and use them for effective routing.

2.4.1 Motivation for the SO Algorithm

Suppose that a large network is divided into four regions, each associated with one of the four cardinal points (N, S, E, and W). The border of these regions are not well delimited but most of the nodes can easily figure out which region they belong to. Consider also a source node desiring to send a message to a destination node in region W. Furthermore, assume that the source knows that the destination is within region W but does not have any route toward this destination node. In this scenario the source node may initiate a route discovery procedure that will result in the broadcast (flooding) of a REQUEST over the entire network.

Alternatively, the source node could begin forwarding the data packets (not the REQUEST) in the "West" direction, hoping that along the way the packets will be heard by a node that has knowledge of some routes toward the destination. In this case a broadcast will not be necessary. As the network size increases, the cost induced by broadcast storms increases rapidly and the second alternative – preventing flooding the large network with a route request – become, by far, more attractive than the first one. To implement this second approach, however, two major problems are encountered. First, the nodes do not know what direction West is; and second, the source does not always know in which region the destination currently is.

The first problem can be addressed if one node is chosen inside each region as a "beacon". This node may be for example some node in the center of each region. Each node serving as a "beacon" is referred to as *reference node.* Then, a tracking algorithm – as for example TORA [63], some geographical routing algorithm, or even standard distance vector (SDV) – may be used to track these four reference nodes. Thanks to the tracking algorithm all the nodes (possible sources) would have downstream links toward each reference node. For example, let N_W be the reference node of region W. All the nodes (especially the nodes outside W) would have downstream links toward N_W and therefore toward region W. These downstream links provide each node with a sense of direction toward region W. Thus, the first problem may be solved but at the cost of creating reference nodes and tracking them.

The second problem can also be solved with the inclusion of a location management scheme that takes into account the past history of a node. If a node was in

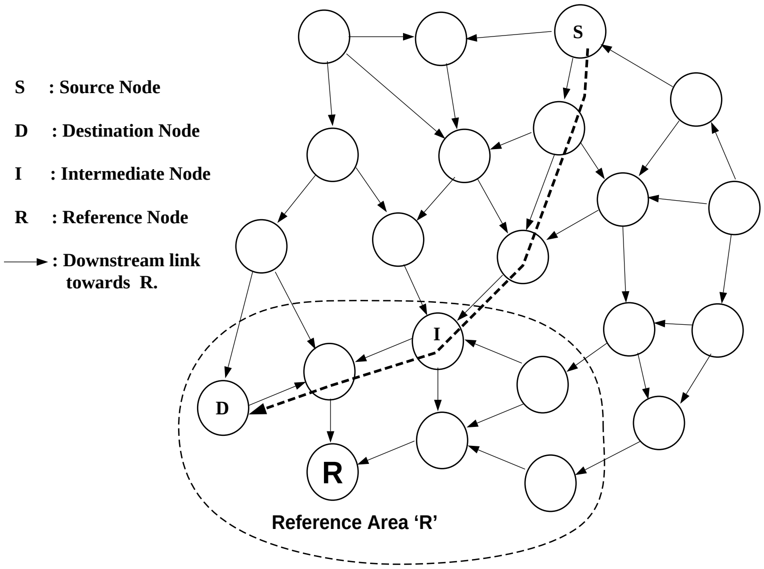


Figure 2.3: Routing using the self-organizing algorithm and the reference area concept

the immediate past inside the West region it is unlikely to be too far away due to the expected spatial/temporal correlation. Thus, the packet could be sent initially toward the West zone. As the packet crosses the network and gets closer to the destination node, more up-to-date information will be available (by means of the LLS algorithm) and the packet will eventually be routed toward the current location of the destination.

An example of routing using the reference area concept is illustrated in Figure 2.3. A node S has a packet to send to node D which is inside a reference area R (the reference area is denoted by its reference node). As stated above, all the nodes (even D) have routes toward node R (using some tracking algorithm such as TORA). In this example it is assumed that node S knows in which reference area node D is by means of a location management agent. Therefore, node S sends the packet in the 'direction' of node (reference area) R. The packet will travel the network, following always the direction 'closer to node R' until reaching node I. Node I knows the exact location of node D (by means of the LLS algorithm, or any other approach following the principle : "the closer you are, the more up-to-date information you have") and interrupts the flow of the packet sending it directly to node D (no longer along the direction to node R).

The above approach will reduce the number of broadcasts (flooding) that would be produced by reactive protocols. This algorithm may be seen as an improvement over traditional route discovery/flooding algorithms that with an additional proactive cost dramatically reduces the number of broadcasts in a large, dynamic, highly loaded network. It should be noted, however, that for some nodes – due to their high mobility or low use – it may not be cost-effective to maintain reference nodes and track their location. For those nodes other alternatives such as route discovery or even flooding may be considered. The proposed multi-mode framework considers these cases by employing route discovery/flooding mechanisms to route packets to these nodes when the cost required to maintain reference nodes to these destinations is greater that the expected cost due to broadcasting route requests to these nodes.

2.4.2 Self-Organizing (SO) Algorithm's Methodology

The self-organizing algorithm is responsible for choosing the best candidate to be a reference node and defining the reference areas. The network will construct routes toward these reference areas in a proactive fashion (before these routes are required).

It is clear that there is a benefit in grouping nodes in a reference area, mainly because the number of broadcasts needed is significantly reduced (gain). On the other hand, there is a bandwidth overhead associated with maintaining a reference area and tracking a reference node (cost).

Route pre-calculation in a highly mobile environment has been generally considered to be inefficient since these routes may become obsolete before used. In the proposed algorithm, the routes constructed are *likely* to be used because reference areas are created only when it is likely to have sessions destined to nodes inside these areas. In other words, the cost of maintaining routes toward the reference area is 'shared' among all the nodes inside it. It may be possible that the self-organizing algorithm decides not to create any reference area (because the gain is lower than the cost), but this decision will depend on the network state and will not be an *a priori* (maybe incorrect) decision of the routing protocol.

The self-organizing algorithm will choose the nodes with the larger expected gain, and if this gain is greater than the expected cost, these nodes will be chosen as reference nodes and define new reference areas around them. The exact calculation of the expected gain of a candidate reference area (node) is not an easy task. In general, that gain would depend on whether the nodes remain inside the reference area (mobility pattern) and whether new sessions having their destination node inside the reference area (requiring otherwise a broadcast) are originated (traffic pattern). Therefore the gain should be a two-dimensional function of mobility and traffic. As a starting point, a tentative gain function is presented next. Also, the cost function (of maintaining a reference area) is a function of mobility and is also difficult to compute. Some guidelines are given in the corresponding subsection.

2.4.3 Gain Function for the SO Algorithm

In [99] the concept of 'footprint size' has been introduced. The footprint size of a given node is equal to the number of k-neighbors of a node (nodes that are at a distance of k hops or less). A classification of the nodes may be based in this footprint size. The larger a node's footprint size the larger the region that the node covers and the less likely that a given node will leave that region in the near future. Therefore, all other things being equal, a node with higher footprint size would be a better candidate to be a reference node, since it could serve as a beacon for a greater number of nodes. Here, the concept of footprint size is extended to a gain function that will be used to determine which nodes are the most suited to be reference nodes.

The gain function at a level L and at time t for a reference area is proposed to be equal to the expected number of broadcasts saved over the next L seconds. A broadcast will be saved if the destination of a new *session* is inside a reference area. Here, the term *session* is applied to a sequence of packets not more than T_s seconds apart. For example two consecutive file transfers to the same destination will be considered to be one session, whereas in a transaction that requires a packet to be send every half hour, each packet will be considered to form a new session.

Let A be a potential reference node and let $V(A,t)$ be its reference area at time t; let $G_L(A,t)$ be the gain function of node A at time t and at level L. Each node

$i \in V(A,t)$ has two parameters : $S_i(A,t,L)$ (percentage of the next L seconds that node i will stay inside node A's zone, i.e. within a radius of k from A) and $R_i(t,L)$ (total expected number of new sessions having node i as destination, over the next L seconds). Then, $G_L(A,t)$ can be defined as:

$$G_L(A,t) \quad = \sum_{i \in V(A,t)} S_i(A,t,L) R_i(t,L) \tag{2.1}$$

$R_i(t,L)$ will be computed by each node i (it is independent of the reference node) and will be broadcast inside a radius k piggy-backed in link-state information packets. Nodes different than node i will learn the value of $R_i(t,L)$ from the link state updates. Nodes different than node i will also compute the value of $S_i(A,t,L)$ based on their topology information history.

Different approaches can be considered to estimate the values of $S_i(A,t,L)$ and $R_i(t,L)$, depending on the desired amount of complexity. For $S_i(A,t,L)$, a first approach could be to set its value to the inverse of the distance (in hops) between nodes A and i elevated to an integer exponent. A better choice may be to consider the percentage of the previous L seconds that the node was inside $V(A,t)$ and assume that this will be repeated. A more complex approach (and likely a more successful one) will be to consider node i's location history over the past L seconds and estimate its trajectory (incoming, outgoing, etc.). Different estimation techniques may be tried to find the more accurate one for a limited complexity implementation.

For the value of $R_i(t,L)$ one simple approach would be to associate a value equal to 1 for nodes already in a session (likely to be broken) and a fixed small value for nodes not in a communication. This approach will result in a gain function closely related to the footprint size [99] (that is, nodes with greater footprint size will have greater gain). A more complex approach should take into consideration different traffic patterns for different classes of users. For example in a client-server architecture, it is likely that the server receives constant requests resulting in a large number of new sessions, however these sessions will be short resulting in almost no broken sessions. On the other hand, if the node is handling voice communications it is likely to have a small number of new sessions, but since their length is greater it is more likely to have broken sessions. Therefore, the class of the user has to be taken into account along with its past history to estimate its traffic behavior.

The established set of reference nodes reveals the structure of the network. In fact, they represent the skeleton of an ad hoc network. For example, if a network is composed of two group of users performing two different tasks, the network should be composed of two reference node (areas). Also, since the reference nodes may be required to perform some extra functionalities (due to their leadership role) it would be appropriate to also weight the computational power and available bandwidth. In multi-class networks the availability of a second, more powerful interface can also be considered as a plus for a candidate reference node as will be pointed out in section 2.5.

2.4.4 Cost Function for the SO Algorithm

The cost associated with a reference area consist of three components : cost of creating a reference area, cost of tracking the reference node, and cost of continuous link-state updating inside a reference area. The last cost is not generated by the self-organizing algorithm necessarily, since the LLS algorithm, as well as others, would propagate the link-state information to the closer nodes anyway. Thus, only the first two cost components need to be considered in the considerations of creating the reference areas.

The cost of creating a reference node is equal to the cost of sending a broadcast packet announcing the new reference node. The cost of tracking the reference node increases with time and is dependent on the network characteristics, mainly the average number of nodes involved in a topological change and the rate of topological changes. In general the above cost is difficult to estimate, and history information of topological changes (rate and number of nodes involved) over the immediate past needs to be employed.

Finally, the total cost will be the sum of a constant term – the cost of one broadcast – and a linearly time-dependent term – tracking cost.

2.4.5 Location Management

The location management scheme should provide a source node with a past (or current if possible) location (reference area) of a destination. This implies that a location table should be constructed only for nodes inside a reference area.

An initial approach consist on appending the address of all the nodes inside a reference area to the initial broadcast sent by the reference node. This way all the nodes in the network may update their location database. However, having a location database with all the nodes inside reference areas (possible all the network) may not be practical since the memory requirements may grow too large. Alternatives may include but are not limited to : store location information of only the most likely destinations (e.g. servers, task mates, etc.); design some reference nodes as *location agents* storing copies of the network location information or a combination of both. In the first case, it may be possible that a packet needs to be sent to a destination that is not in the 'most likely' link, which will require a broadcast. The second approach reduces the bandwidth overhead due to location updates all over the network but may cause delays in establishing a new session or may even fail due to *location agents* unavailability, in which case a broadcast will have to be sent.

Once again, the particular approach to use will depend on the mobility and traffic patterns. In addition to the parameters discussed previously, it should also be taken into account the source-destination node distribution. Such as for example, whether some nodes *only* send packets to a certain destination, or if a destination *only* receives packets from a set of nodes. The parameters involved in taking the best decision need to be further investigated.

2.5 Application to Multi-Class Ad Hoc Networks

This section presents examples of how a multi-mode routing protocol based on the framework presented in section 2.2 and Figure 2.1 (i.e. a synergetic combination of the LLS and SO algorithms) will provide practical solutions to the problem of routing in complex real life scenarios (multi-class ad hoc networks). Two cases will be discussed : a fixed-mobile network, and a two-level network.

2.5.1 Fixed-Mobile Network

In this subsection an hybrid mobile-fixed network is considered. The hybrid network consist of an ad hoc network of mobile nodes moving around (small power) fixed base stations (interconnected between them). The mobile nodes are equipped with a wireless interface. The fixed base stations are equipped with two interfaces. One is a wireless interface allowing them to communicate with the mobile nodes. The other is a wireline interface connecting the the base station among themselves and to the fixed (external) network.

This environment may be considered as the natural extension of a pico-cellular system when the fixed base station can not cover every spot but there are some shadows areas left. Nodes outside the coverage region may relay on neighboring nodes, closer to the base station, to forward packet to and from them. Such a network will allow a reduction in the transmission power requirement of the mobile nodes as well the base station, increasing battery life and potentially increasing the spatial reuse if a large percentage of the traffic is among mobile nodes.

In this environment, the LLS algorithm will successfully identify the stable links between base stations. On the other hand, the SO algorithm should identify base stations as candidates for reference nodes and would permanently maintain route toward them. This is the case if as expected, a relatively stable set of nodes moves around each fixed base station. Also, since the base stations are the only gateways to the external (fixed) network, they concentrate all the outgoing traffic, being likely destinations, resulting in high values for their gain functions. Additionally, the base stations will be considered as reference nodes if in the definition of gain function given in subsection 2.4.3, a factor to account for the base stations' second interface is included (as was previously suggested).

Finally, all the nodes in the network will have routes to the base stations making it possible to take advantage of the fixed network infrastructure to forward packets to mobile nodes far away or to fixed network's nodes. On the other hand, the association

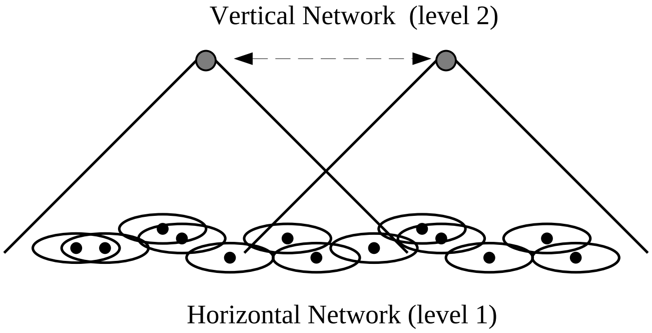


Figure 2.4: An example of a two-level network

between mobile nodes and reference areas (base stations) will allow the external fixed network to forward packets destined to a mobile node to the base station closer to the mobile node, and from the base station to the mobile node itself.

2.5.2 Two-Level Network

In this subsection, an ad hoc network formed by two classes of mobile users is considered. One class of user forms the horizontal network whereas another class constitutes the vertical network. Nodes in the horizontal network are characterized for a small footprint size (set of neighbors) and high mobility (relatively to another horizontal nodes), whereas nodes in the vertical network has larger footprint size and a more stable set of neighbors. As an example of such a two level network, a disaster recovery scenario employing land stations as well as helicopters may be considered. Low powered land nodes will constitute the horizontal network, whereas the helicopters, with more powerful interfaces and line-of-sight may cover a greater area constituting a vertical network. Figure 2.4 gives an example of the topology of a particular two-level network.

Another example is a network of mobile nodes and a high power base station. The nodes may communicate directly to each other and if they become isolated they may need to use the base station as a repeater. If the mobile node and the base stations use the same interface (frequency) it is clear that a transmission from or to the base station may cause the remaining nodes in the network to stop their transmissions. It is clear that in that case (single-class or single-interface network), the base station will be used only if a node is otherwise unreachable. However, if the base station uses a different frequency (interface) to communicate with the mobile nodes than the frequency the mobile nodes use to communicate among themselves (multi-class network), then the mobile-base station interface may be seen as an additional resource that may be exploited. In this example the mobile nodes constitute the horizontal network and the base station (alone) constitutes the vertical network. There may also be more than one base stations.

The above examples constitute multi-class networks since some nodes (horizontal or vertical) may posses only one interface whereas others may have two; that is, a horizontal node may have a horizontal-horizontal interface together with a horizontal-vertical interface. The same is applicable to a vertical node. In any of the previous cases, the existence of two different levels is an advantage, and the horizontal-vertical interface is a resource that has be to used wisely. Since there is a large number of nodes expected to have access to an horizontal-vertical interface, that resource will become easily congested with transmissions that might as well be forwarded over the horizontal network. At the same time, under-utilizing the horizontal-vertical network may reduce the throughput and increase the delay over the horizontal network. Different routing possibilities in a two-level network are shown in Figure 2.5 where a source node (S) tries to reach a destination node (D); node V belongs to the vertical network and may or may not be involved.

Next, a discussion on the expected results of applying a multi-mode routing protocol based on the framework presented in section 2.2 and Figure 2.1 is presented.

By employing the LLS algorithm, each node in the vertical network will have full knowledge of the horizontal network topology. At the same time, nodes in the horizontal network will be aware of the horizontal-vertical links. The vertical nodes and their coverage area will be easily discovered. The SO algorithm may focus on the horizontal network only. In this particular case it is easy to argue that the vertical nodes need to be excluded from the SO algorithm. In general, though, this may not be the case and further research is required to come up with well defined rules to determine which nodes should participate in the SO algorithm. Finally, the SO algorithm will identify the best candidates to be reference nodes. Again, reference nodes should be created only if it is worthy.

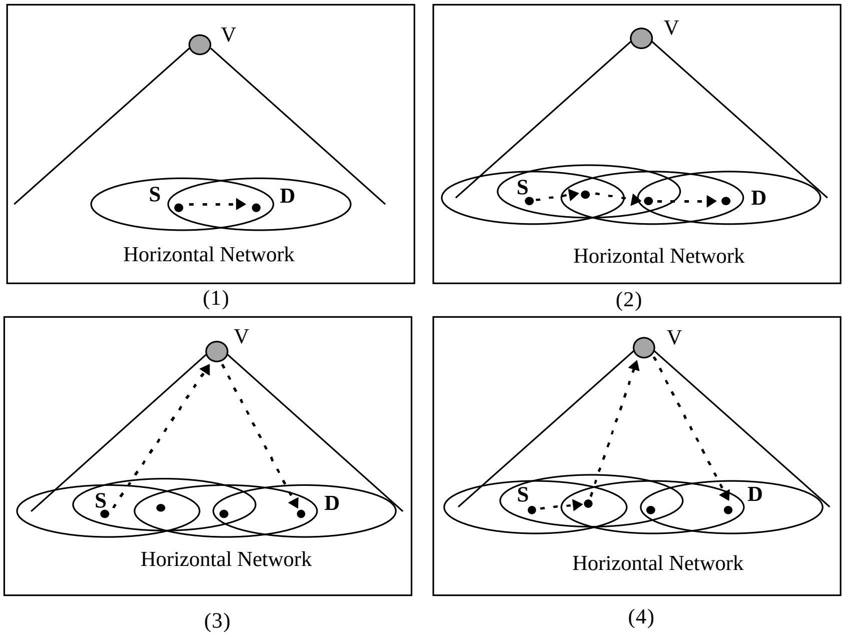


Figure 2.5: Four possibilities for resource utilization between a Source (S) and a Destination (D) in a two level network

Consider the case when traffic and mobility rates make it attractive to create reference areas. In this case, the location management scheme is greatly simplified, as follows. If the source node has outdated location info to the destination the node will send a REQUEST using the vertical network. The destination node will respond to the REQUEST with its current location. At this moment the source may forward successive packets of the session over the horizontal network avoiding congestion of the horizontal-vertical interface (which is needed to forward packets between horizontal nodes that are unreachable through the horizontal network). In case the horizontal-vertical interface is lightly loaded, then a comparison between the cost of using the horizontal network versus using the horizontal-vertical interface should be performed.

The presence of the vertical network is expected to reduce the broadcasts and other congestion-causing packets in the horizontal network. Also, packets that would need to traverse long paths in the horizontal network will be forwarded using the horizontal-vertical interface whereas packets that would need to traverse short paths in the horizontal network will be transmitted over the horizontal network.

When horizontal routes are available, the cost of using the horizontal network will depend on the routes length, congestion, and quality (stability). When the location is not well known but only past information is available, the cost function should also reflect the likelihood that the destination is around the last recorded position and therefore will be dependent on the elapsed time. The cost of using the horizontal-vertical interface will be dependent mainly on the bandwidth availability.

In conclusion, the LLS and SO algorithm working together will allow for a more efficient use of the communications resources on a two level network.

2.6 Summary

This chapter presented a framework for a multi-mode routing protocol that adapts itself to the current network conditions (size, mobility and traffic patterns). Based on the network state information provided by the structure-learning/engaging modules, a multi-mode routing engine should determine and enforce the most efficient 'mode' of operation for each destination.

Two complementary algorithms were proposed to be running in each structure-learning/engaging module. These algorithms will be running permanently at their respective module (at each node) in a proactive fashion. However, they will propagate information (across peer modules) over the network only when it is justified, based on a control overhead versus routing degradation trade off analysis.

Thanks to the information propagated by these structure-learning/engaging algorithms, each node's multimode routing engine (see Figure 2.1) will have knowledge of the state of some links (the closer ones and even some links far away that are stable), as well as links towards some regions of the network (reference areas). Additionally, information regarding the location (reference area) of some destinations will be available proactively. Based on this information a multi-mode routing engine (the core of a multi-mode routing protocol) may select its 'mode' of operation. Possible decisions may include the use of a pre-calculated path of stable links (if available and if stable links are not congested); the use of links toward the destination node's reference area expecting that the packet at some point will find a node with knowledge of routes toward the destination (see Figure 2.3); the use of a query or a broadcast packet to get the destination node's location information; or simply use a combination of route discovery/flooding.

Finally, a discussion of the applicability of the proposed framework to solving some complex real life situations has shown the potential of using a multi-mode routing approach. This potential justifies further research in this area.

The next chapters will further study the LLS and SO algorithms, the basic pieces of the multi-mode routing framework in isolation. This study is intended to provide a better insight into their dynamics that will prove useful when actually designing a multi-mode routing protocol. The work in the next chapters temporarily deviates from the multi-mode routing protocol design, since it was recognized that a deep understanding of the dynamics involved in ad hoc networks was missing. The next chapters close this gap by providing the first theoretical study of the fundamental properties and limits of ad hoc networks. To this extent, LLS and SO are studied separately. Chapters 3 and 4 focus on simplified variants of LLS and the issue of scalability in mobile ad hoc networks. Chapter 5 studies the feasibility of the SO algorithm and presents a detailed description of the first ad hoc routing protocol that tries to learn and exploit the mobility and traffic patterns. Further work should exploit the gained understanding to analyze the LLS and SO interactions and bring them together in a common multi-mode routing protocol, following the framework presented in section 2.2 and Figure 2.1.

Chapter 3

HSLS : Making Link-State Routing Scale

In the previous chapter, a framework for a multi-mode routing protocol for ad hoc networks was proposed. This framework consists of three elements: a multi-mode routing engine, a self-organizing module, and a limited information dissemination module. The main function of the limited information dissemination module is to provide the other two modules with partial information about the state of other nodes in the network. More information is provided for nodes closer to the node, and the granularity and quality of the information provided is allowed to degrade for nodes farther away. This way, the limited information dissemination module may provide necessary information for nodes close by without congesting the network.

The specific algorithm to be executed at the limited information dissemination module has not been specified, but a good candidate, the Limited Link State (LLS) algorithm, was proposed. LLS limits the depth of propagation of Link State Updates (LSUs) by setting the LSU packet's Time To Live (TTL) field to lower values than in Standard Link State (SLS), based on each link stability and past history of LSUs. As a result, nodes will have information about stable links far away and unstable links close by.

However, the previous chapter did not complete the LLS algorithm specification, that is, it did not define the specific functional dependency of propagation depth of

LSUs with time and stability. The reason for leaving the specification incomplete was that the author recognized that, previous to this chapter work, the impact of the limited LSU propagation on the effectiveness of a routing protocol was not well understood. Thus, the author realized that a previous step before fully characterizing the LLS algorithm is to understand the trade-off inherent between the reduction of the control overhead and the degradation of the routing protocol performance due to such a reduction.

In this chapter, simplified versions of LLS are studied. Basically, we relax LLS's dependency on the link stability and characterize the impact of limited information about distant nodes in the routing protocol effectiveness. The results obtained in this chapter and next one, provide a guideline for designing the LLS algorithm. These results also provide a new understanding of the scalability properties of link-state algorithms, in particular, and of the scalability limits of routing protocols, in general.

The family of LLS simplifications studied in this work are referred to as the Fuzzy Sighted Link State (FSLS) family. Members of this family capture the basic LLS property that nodes close by will have more up-to-date information that nodes farther away. Thus, a node view of the network is 'fuzzy'. FSLS may be interpreted as the result of applying LLS to an homogeneous network, where all links are unstables.

FSLS is a contribution in its own right, since it is an improvement/generalization over existing link state approaches. FSLS attempts to scale link-state routing by limiting the scope of link state update dissemination in space and over time. Thus, study of the family of FSLS algorithms lead to an understanding of the fundamental limitation of link-state scalability and to the derivation of the (probably) optimal [1] link-state algorithm: the Hazzy Sighted Link State (HSLS) protocol. HSLS is an easy-to-implement link-state variant that present the best scalability properties among link-state and other protocols. Thus, HSLS is an important, timely contribution of this dissertation. Another contribution is the better understanding of the limits on

[1] We found a feasible solution inducing a total overhead that is less than 1% above a theoretical lower bound. Thus, from an engineering point of view, our solution may be regarded as equivalent to the optimal, i.e. if a better feasible solution existed, since it would also induce a total overhead greater than the lower bound, that solution performance and ours will be less than 1% apart, and may be regarded as equivalent.

scalability on ad hoc networks provided by this and the next chapter.

In this chapter, the first fundamental analysis of the class of FSLS algorithms is presented. Using a novel perspective on the "overhead" of a protocol that includes not only the overhead due to control messages but also due to route suboptimality, an analytical model whose solution automatically leads to the best algorithm in this class is formulated. This algorithm (HSLS) is shown to have (in the next chapter) the best asymptotic overhead among known routing protocols – proactive or reactive alike.

The remainder of this chapter is organized as follows: Section 3.1 gives an overview of the previous work on link state routing and scalability. This review will be later complemented (section 3.3) with a more in-depth discussion/comparison of those protocols in the literature that present similarities with members of the FSLS family. Section 3.2 presents a discussion on scalability that leads to the definition of the *total overhead* and to focus on limited dissemination link state approaches. Section 3.3 introduces the family of Fuzzy Sighted Link State (FSLS) algorithms, that are intended to reduce (limit) the routing information overhead at the expense of some route suboptimality. Section 3.4 presents a mobility-based, probabilistic, analytical model that allows to determine the best algorithm in the FSLS family, namely the Hazy Sighted Link State (HSLS) algorithm. Section 3.5 discusses some implementation issues and present the algorithmic description of the proposed protocol. Section 3.6 complements the analysis with simulation results. Finally, Section 3.7 presents this chapter's conclusions.

3.1 Related Previous Work

Since its inception as part of the ARPANET, link-state routing has become the most widely used approach in the Internet. Its popularity has resulted from its unique advantages, including simplicity, robustness, predictable dynamics, and unmatched support for flexible QoS-based route generation. Unfortunately, as it is widely recognized, link-state routing as used in the wired Internet scales poorly when used in mobile ad hoc networks.

Given its advantages, a sufficiently scalable version of link-state routing would be invaluable for ad hoc networks. Not surprisingly therefore, there are a number of approaches in the literature with this goal. These approaches may be classified into *efficient dissemination* approaches and *limited dissemination* approaches. Both attempt to reduce the routing update overhead, but do so in different ways. In efficient dissemination, updates are sent throughout the network, but more efficiently compared to traditional flooding. Examples include TBRPF[103], OLSR [57], STAR [104], etc. In contrast, limited dissemination consists of restricting the scope of routing updates in space and time. Examples include hierarchical link state [105], FSR and GSR (see [116]), etc. It may be noted that limited dissemination techniques hold a greater potential to reduce routing overhead as network size increases. Limited dissemination techniques are not exclusive to link-state routing, and there have been proposals like DREAM [109] and ZRP [112] that apply this concept to geographical and hybrid (proactive and reactive) routing, respectively.

Scalability and other performance aspects of ad hoc routing have been studied predominantly via simulations. The lack of much needed theoretical analysis in this area is due, we believe, in part to the lack of a common platform to base theoretical comparisons on, and in part due to the abstruse nature of the problem. Despite limited prior related theoretical work, there have been notable exceptions. In [81] analytical and simulation results are integrated in a study that provides valuable insight into comparative protocol performance. However, it fails to deliver a final analytical result, deferring instead to simulation.

In this chapter, limited dissemination techniques are considered from a fundamental viewpoint. This treatment is anchored around the following generalized link-state routing approach: send an update every t_i seconds with a network scope of r_i hops. This represents a family of techniques for each combination of instantiations of t_i and r_i. The family includes many intuitively feasible and useful techniques, including traditional link-state routing. In the context of this generalized approach, the problem of instantiating t_i and r_i so that the performance be optimized is considered. Solving this problem automatically yields the best protocol in this family.

Limited dissemination techniques incur a cost in terms of suboptimal routing that needs to be considered in formulating the optimization problem and conducting the analysis. Indeed, this is a case with many other routing protocols as well, including DSR[107], AODV [108], etc. Traditionally, the cost of suboptimal routing has been ignored, and only the cost of control message overhead has been considered. A new definition of "overhead" that includes not only the control message overhead but also the cost of suboptimal routing is proposed. [2] Such a definition facilitates a fair comparison of protocols not only within the fuzzy-sighted family, but also amongst previously published protocols.

The contributions of this work include the following. A new design space for link state routing protocols is introduced by presenting a family of (potentially scalable) algorithms that are neither global nor local, but where each node may have a different view of the network. A new definition and methodology for computing the overhead induced by a routing protocol is introduced. This methodology allows for comparison among quite diverse protocols (see next chapter). A mobility-based, probabilistic, analytical model for the study of link state routing algorithms is presented. As a consequence, the scalability limits of link state algorithms are well understood and the best link state variant is identified.

Finally, a unique feature of the present work is that the resulting algorithm – the Hazy Sighted Link State (HSLS) – is *synthesized* automatically from the analysis, rather than being followed by the analysis as is tipically the case. Morever, it is performance-driven, focusing on average system performance instead of focusing on handling exceptional (rare) cases [3], or achieving qualitative characteristics (loop freedom, database consistency, etc.) whose impact on the overall system performance is not clear.

[2] While both the control overhead and the suboptimal routing degradation have been considered as performance metrics in the past, this work is the first one that combines both in a unified metric. As a consequence, a closed form expression quantifying the combined effect due to limited information dissemination was possible to derive.

[3] Exceptional cases are best considered *after* the baseline approach has been worked out, provided that the exceptional cases are rare and do not cause the algorithm to break.

3.2 A New Perspective on Scalability

Traditionally, the term *overhead* has been used in relation to the *control overhead*, that is, the amount of bandwidth required to construct and maintain a route. Thus, in proactive approaches overhead has been expressed in terms of the number of packets exchanged between nodes, in order to maintain the node's forwarding tables up-to-date. In reactive approaches, overhead has been described in terms of the bandwidth consumed by the route request/reply messages (global or local). Efficient routing protocols try to keep the aforementioned overhead low.

While it is true that the control overhead significantly affects the protocol behavior, it does not provide enough information to facilitate a proper performance assessment of a given protocol since it fails to include the impact of suboptimal routes on the protocol's performance. As the network size increases above, say, 100 nodes, keeping route optimality imposes an unacceptable cost under both the proactive and reactive approaches, and suboptimal routes become a fact of life in any scalable routing protocol. Suboptimal routes are introduced in reactive protocols because they try to maintain the current source-destination path for as long as it is valid, although it may no longer be optimal. Also, local repair techniques try to reduce the overhead induced by the protocol at the expense of longer, non optimal paths. Proactive approaches introduce suboptimal routes by limiting the scope of topology information dissemination (e.g. hierarchical routing [105]) and/or limiting the time between successive topology information updates dissemination so that topology updates are no longer instantaneously event-driven (e.g GSR [116]).

Thus, it is necessary to revise the concept of *overhead* so that it includes the effect of suboptimal routes in capacity limited systems, that is, *suboptimal routes not only increase the end-to-end delay but also result in a greater bandwidth usage than required.* This extra bandwidth is an overhead that may be comparable to the other types of overhead. Approaches that attempt to minimize only the control overhead may lead to the (potentially erroneous) conclusion that they are "scalable" by inducing a fixed amount of the aforementioned overhead, while in practice the resulting performance be seriously degraded as the extra bandwidth overhead induced by suboptimal routes increases with the network size. Thus, a more effective definition of the overhead – which will be considered in the remainder of this work – is introduced in the next subsection.

3.2.1 Total Overhead

In order to quantify the effect of a routing protocol on the network performance, the *minimum traffic load* of the network as a routing protocol-independent metric needs to be defined first, as follows:

Definition 3.1 *The* minimum traffic load *of a network, is the minimum amount of bandwidth required to forward packets over the shortest distance (in number of hops) paths available, assuming all the nodes have instantaneous* a priori *full topology information.*

The above definition is independent of the routing protocol being employed, since it does not include the control overhead but assumes that all the nodes are provided *a priori* global information. It should be noted that it is possible that in fixed networks a node is provided with static optimal routes, and therefore there is no bandwidth consumption above the *minimum traffic load*. On the other hand, in mobile scenarios this is hardly possible. Due to the unpredictability of the movement patterns and the topology they induce, even if static routes are provided so that no control packets are needed, it is extremely unlikely that these static routes remain optimal during the entire network lifetime. Thus, since suboptimal routes are present, the actual network bandwidth usage would be greater than the *minimum traffic load* value. This motivates the following definition of the *total overhead* of a routing protocol.

Definition 3.2 *The* total overhead *induced by a routing protocol is the difference between the total amount of bandwidth actually consumed by the network running such routing protocol minus the* minimum traffic load *that would have been required should the nodes had* a priori *full topology information.*

Thus, the actual bandwidth consumption in a network will be the sum of a protocol independent term, the *minimum traffic load*, and a protocol dependent one, the *total* *overhead*. Obviously, effective routing protocols should try to reduce the second term (*total overhead*) as much as possible. To this end, the remainder of this section will discuss the different sources that contribute to this *total overhead*. Such a study will provide a methodology for the computation of the *total overhead* induced by members of the FSLS family of algorithms. This methodology will also prove useful for computing the *total overhead* for several representative protocols in the literature (see next chapter).

The different sources of overhead that contribute to the *total overhead* may be grouped and expressed in terms of *reactive*, *proactive*, and *suboptimal routing overheads*. These sources of overhead have been considered in the past, but this work is the first one that effectively combines all of them in a unified framework, leading to the derivation of a tractable model.

The *reactive overhead* of a protocol is the amount of bandwidth consumed by the specific protocol to build paths from a source to a destination, *after* a traffic flow to that destination has been generated at the source. In static networks, the reactive overhead is a function of the rate of generation of new flows. In dynamic (mobile) networks, however, paths are (re)built not only due to new flows but also due to link failures in an already active path. Thus, in general, the reactive overhead is a function of both traffic *and* topology change.

The *proactive overhead* of a protocol is the amount of bandwidth consumed by the protocol in order to propagate route information *before* it is needed. This may take place periodically and/or in response to topological changes.

The *suboptimal routing overhead* of a protocol is the difference between the bandwidth consumed when transmitting data from all the sources to their destinations using the routes determined by the specific protocol, and the bandwidth that would have been consumed should the data have followed the shortest available path(s). For example, consider a source that is 3 hops away from its destination. If a protocol chooses to deliver one packet following a k ($k > 3$) hop path (maybe because of out-of-date information, or because the source has not yet been informed about the availability of a 3 hop path), then $(k-3) * packet\_length$ bits will need to be added to the suboptimal routing overhead.

The *total overhead* provides an unbiased metric for performance comparison that reflects bandwidth consumption. Despite increasing efficiency at the physical and MAC-layers, bandwidth is likely to remain a limiting factor in terms of scalability, which is a crucial element for successful implementation and deployment of ad hoc networks. The authors recognize that *total overhead* may not fully characterize all the performance aspects relevant to specific applications. However, it can be used without loss of generality as it is proportional to factors including energy consumption, memory and processing requirements; furthermore, delay constraints have been shown to be expressed in terms of an equivalent bandwidth [82].

3.2.2 Achievable Regions and Operating Points

The three different overhead sources mentioned above are locked in a 3-way trade-off since, in an already efficient algorithm, the reduction of one of them will most likely cause the increase of one of the others. For example, reducing the 'zone' size in ZRP will reduce ZRP's proactive overhead, but will increase the overhead incurred when 'bordercasting' new route request, thus increasing ZRP's reactive overhead. The above observation leads to the definition of the *achievable region* of overhead as the three dimensional region formed by all the values of proactive, reactive, and suboptimal routing overheads that can be achieved (induced) by any protocol under the same scenario (traffic, mobility, etc.). Figure 3.1 shows a typical 2-dimensional transformation of this 'achievable region' where two sources of overhead (reactive and suboptimal routing) have been added together for the sake of clarity. The horizontal axis represents the proactive overhead induced by a protocol, while the vertical axis represents the sum of the reactive and suboptimal routing overheads.

It can be seen that the achievable region is convex [4], lower-bounded by the curve of overhead points achieved by the 'efficient' (i.e. minimizing some source of overhead given a condition imposed on the others) protocols.

For example, point P is obtained by the best pure proactive approach given that

[4] To see that the achievable region is convex, just consider the points P_1 and P_2 achieved by protocols $\mathcal{P}_1$ and $\mathcal{P}_2$. Then, any point $\lambda P_1 + (1-\lambda)P_2$ can be achieved by engaging protocol $\mathcal{P}_3$ that behaves as protocol P_1 a fraction λ of a (long) time and as protocol P_2 the remaining of the time.

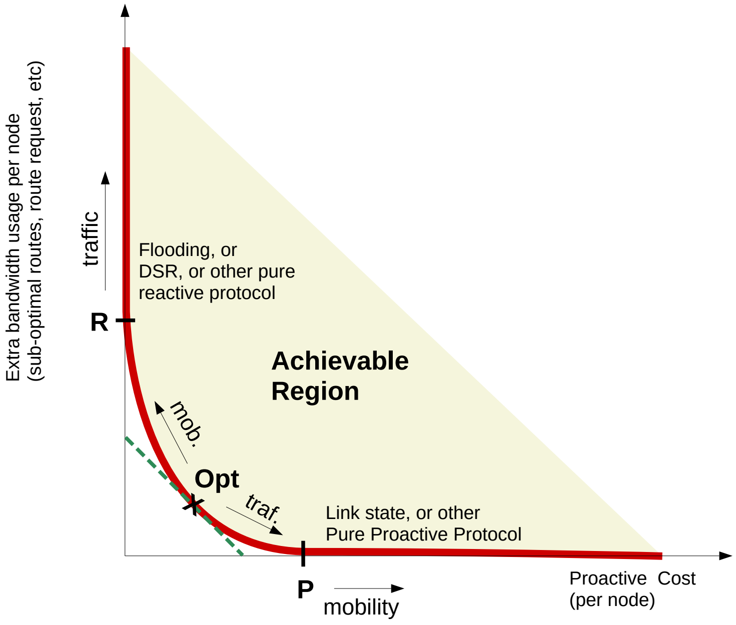


Figure 3.1: Overhead's achievable region.

optimal routes are required, that is, given the constraints that the suboptimal and reactive overheads must be equal to zero. P moves to the right as mobility increases. Similarly, point R is achieved for the best protocol that does not use any proactive information. Obviously, the best protocol (in terms of overhead) is the one that minimizes the *total overhead* achieving the point *Opt* (point tangent to the curve $x + y = constant$).

Different scenarios result in different slopes of the boundary of the achievable region and consequently different points *Opt*. For example, if the traffic increases or diversifies R moves upward and, if mobility is low P moves to the left and may cause *Opt* to coincide with the point P (pure proactive protocol with optimal routes). The reverse is also true as the mobility rate increases and the traffic diversity/intensity decreases. Figure 3.2 shows how the boundary of the achievable region is (re)shaped as the network size increases. The lower curve corresponds to the boundary region when the network size is small. The effect of increasing the network size is to 'pull' the boundary region up. However, the region displacement is not uniform as will be discussed next.

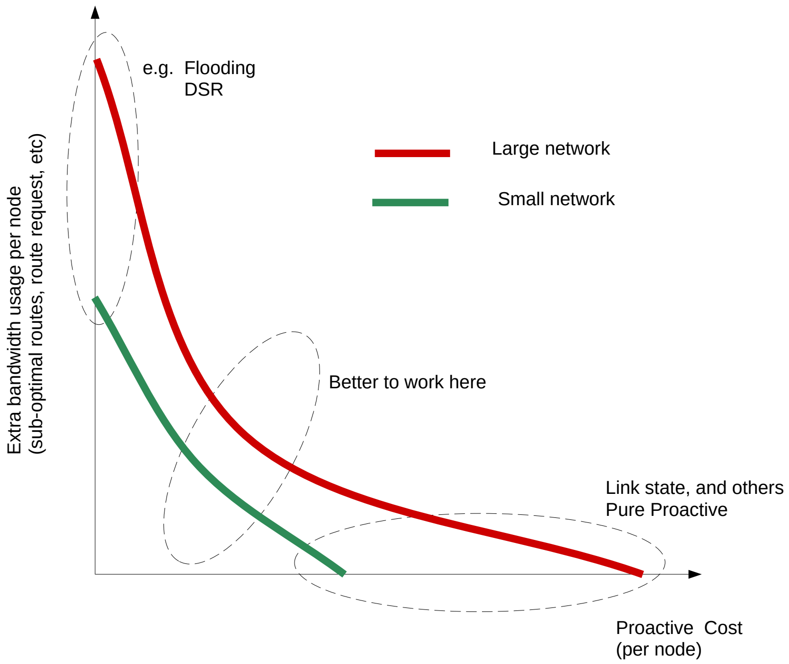


Figure 3.2: Change in achievable region due to size.

Pure proactive protocols, as for example SLS, may generate a control message (in the worse case) each time a link change is detected. Each control message will be retransmitted by each node in the network. Since both the generation rate of control messages and the number of message retransmissions increases linearly with network size (N), the total overhead induced by pure proactive algorithms (that determine the point P) increases as rapidly as N^2.

Pure reactive algorithms, as for example DSR without the route cache option, will transmit route request (RREQ) control messages each time a new session is initiated. The RREQ message will be retransmitted by each node in the network. Since both the rate of generation of RREQ and the number of retransmissions required by each RREQ message increases linearly with N, it is concluded that pure reactive algorithms (and the point R) increases as rapidly as N^2.

On the other hand, protocols inducing 'intermediate points', such as Hierarchical link state (HierLS) and ZRP, may increase more slowly with respect to N. In Chapter 4 it is shown that under assumptions a.1-a.8 (Section 3.4.1) HierLS's and ZRP's growth with respect to N was roughly $N^{1.5}$ and $N^{1.66}$, respectively.

Summarizing, it can be seen that points P and R increase proportionally to $\Theta(N^2)$ whereas an 'intermediate' point as HierLS increases almost as $\Theta(N^{1.5})$. [5] Referring again to Figure 3.2, it is easy to see that the extreme points are stretched "faster" than the intermediate points. Thus, *as size increases, the best operating point is far from the extreme points P and R but in the region where the proactive, reactive, and suboptimal routing overheads are balanced.*

Further research should be focused on protocols such as the Zone Routing Protocol (ZRP) [112] HierLS variants (e.g. [105] and [67]), and other protocols that operate in this (intermediate) region, where suboptimal routes are present.

3.3 Fuzzy Sighted Link State (FSLS) Algorithms

It was previously pointed out that a pure proactive protocol such as SLS may not scale well with size since the overhead it induces increases as rapidly as N^2. However, a reduction of the proactive overhead may be achieved both in space (by limiting which nodes the link state update is transmitted to) and in time (by limiting the time between successive link status information dissemination). Such a reduction on proactive overhead will induce an increase in suboptimal routing overhead, and therefore a careful balance is necessary. This observation has motivated the study of the family of Fuzzy Sighted Link State (FSLS) protocols introduced below, where the frequency of Link State Updates (LSUs) propagated to distant nodes is reduced based on the observation that in hop-by-hop routing, changes experienced by nodes far away tend to have little impact in a node's 'local' next hop decision.

In a highly mobile environment, under a Fuzzy Sighted Link State (FSLS) protocol a node will transmit - provided that there is a need to - a Link State Update (LSU) only at particular time instants that are multiples of t_e seconds. Thus, potentially several link changes are 'collected' and transmitted every t_e seconds. The *Time To Live* (TTL) field of the LSU packet is set to a value (which specifies how far the LSU will be propagated) that is a function of the current time index as explained below.

[5] Standard asymptotic notation is employed. A function $f(n) = \Theta(g(n))$ if there exists constants c_1, c_2, and n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.

After one global LSU transmission – LSU that travels over the entire network, i.e. TTL field set to infinity, as for example during initialization – a node 'wakes up' every t_e seconds and sends a LSU with TTL set to s_1 if there has been a link status change in the last t_e seconds. Also, the node wakes up every $2 * t_e$ seconds and transmits a LSU with TTL set to s_2 if there has been a link status change in the last $2 * t_e$ seconds. In general, a node wakes up every $2^{i-1} * t_e$ $(i = 1, 2, 3, ...)$ seconds and transmits a LSU with TTL set to s_i if there has been a link status change in the last $2^{i-1} * t_e$ seconds.[6]

If the value of s_i is greater than the distance from this node to any other node in the network (which will cause the LSU to reach the entire network), the TTL field of the LSU is set to infinity (global LSU), and all the counters and timers are reset. In addition, as a soft state protection on low mobility environments, a periodic timer may be set to ensure that a global LSU is transmitted at least each t_b seconds. The latter timer has effect in low mobility scenarios only, since in high mobility ones, global LSUs are going to be transmitted with high probability.

Figure 3.3 shows an example of FSLS's LSU generation process when mobility is high and consequently LSUs are always generated every t_e seconds. Note that the sequence $s_1, s_2, \ldots$ is non-decreasing. For example consider what happens at time $4t_e$ (see figure 3.3). This time is a multiple of t_e (associated with s_1), also a multiple of $2t_e$ (associated with s_2) and $4t_e$ (associated with s_3). Note that if there has been a link status change in the past t_e or $2t_e$ seconds, then this implies that there has been a link change in the past $4t_e$ seconds. Thus, if we have to set the TTL field to at least s_1 (or s_2) we also have to increase it to s_3. Similarly, if there has not been a link status change in the past $4t_e$ seconds, then there has not been a link change in the past t_e or $2t_e$ seconds. Thus, if we do not send a LSU with TTL set to s_3, we do not send a LSU at all. Thus, at time $4t_e$ (as well at times $12t_e$, $20t_e$ any other time $4 * k * t_e$ where k is an odd number) the link state change activity during the past $4t_e$ seconds needs to be checked and, if there is any, then an LSU with TTL set to s_3 will be sent. Thus, in the highly mobile scenario assumed on figure 3.3, a LSU with

[6] Strictly speaking, the node will consider link changes since the last time a LSU with TTL greater or equal to s_i was considered (not necessarily transmited). This difference does not affect the algorithm's behavior in high mobility scenarios, so it will be ignored for the sake of simplicity.

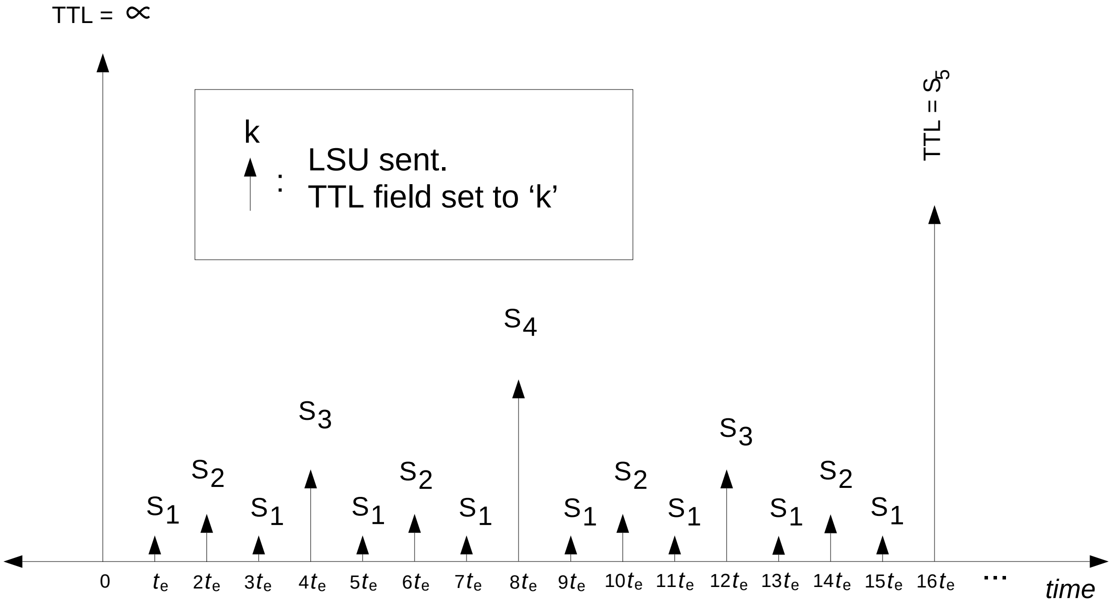


Figure 3.3: Example of FSLS's LSU generation process

TTL equal to s_3 is sent at times $4t_e$ and $12t_e$.

The above approach guarantees that nodes that are s_i hops away from a tagged node will learn about a link status change at most after $2^{i-1}t_e$ seconds. Thus, the maximum 'refresh' time ($T(r)$) as a function of distance (r) is as shown in Figure 3.4. The function $T(r)$ will determine the latency in the link state information, and therefore will determine the performance of the network under a FSLS algorithm.

Different approaches may be implemented by considering different $\{s_i\}$ sequences. Two novel (in this setting) but familiar cases: Discretized Link State (DLS) and Near Sighted Link State (NSLS) (see Figures 3.5 and 3.6) are discussed next.

DLS is obtained by setting $s_i = \infty$ for all i (see Figure 3.5 left). DLS is similar to the Standard Link State (SLS) algorithm and differs only in that under DLS a LSU is not sent immediately after a link status change is detected but only when the current t_e interval is completed. Thus, several link status changes may be collected in one LSU. DLS is a modification of SLS that attempts to scale better with respect to mobility. Under high mobility, DLS presents some similarities with Global State Routing (GSR)[116], another protocol that attempts to scale with mobility. In GSR, a node exchanges its version of the network topology table with its one-hop neighbors

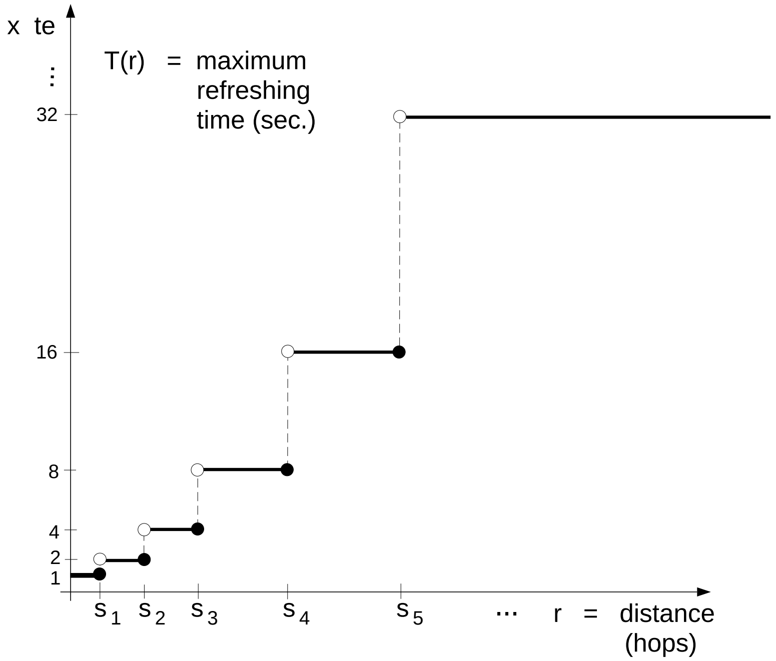


Figure 3.4: Maximum refresh time $T(r)$ as a function of distance from link event.

each t_{flood} seconds. This way, GSR limits the frequency of link state updates to be no greater that $\frac{1}{t_{flood}}$.

In highly mobile scenarios (where LSUs are sent every t_e seconds) DLS induces the same proactive overhead (in bits) as Global State Routing (GSR) (setting $t_e = t_{flood}$), since they both require control packets transmission of the equivalent of N times the average topology table size (in bits) each t_e (t_{flood}) seconds (N is the network size). However, DLS latency on the transmission of LSUs to nodes far away is fixed, i.e. $T(r) = t_e$ (see Figure 3.6 left), while GSR's increases linearly with distance (see Figure 3.7 left), i.e. $T(r) = t_e * r$ (since a link status update will have to wait at most t_e – and on average $\frac{t_e}{2}$ – seconds before it is propagated one more hop away from the node experiencing the link change). Thus, DLS is expected to outperform GSR, especially for large networks [7].

[7] GSR groups several LSUs in one packet. Thus, even if the same number of bits of overhead are transmitted, GSR transmits a smaller number of packets. In some scenarios, for example under a

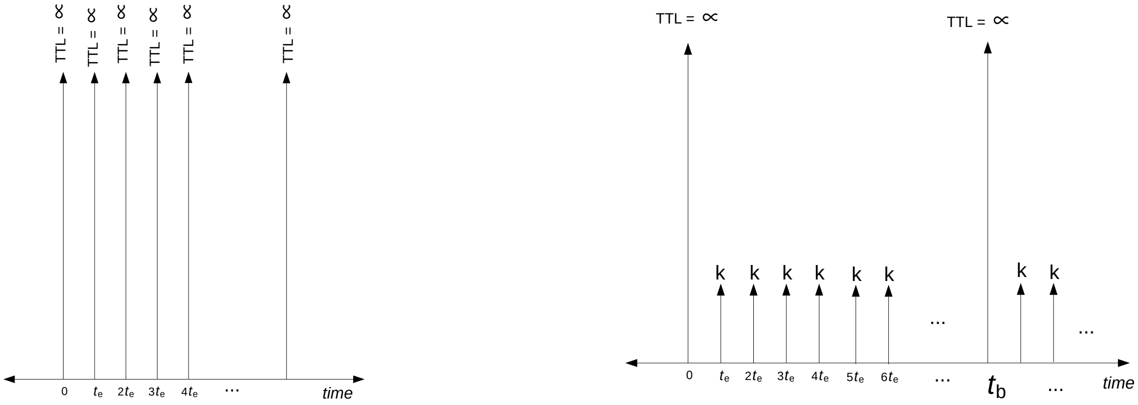


Figure 3.5: DLS's (left) and NSLS's (right) LSU generation process.

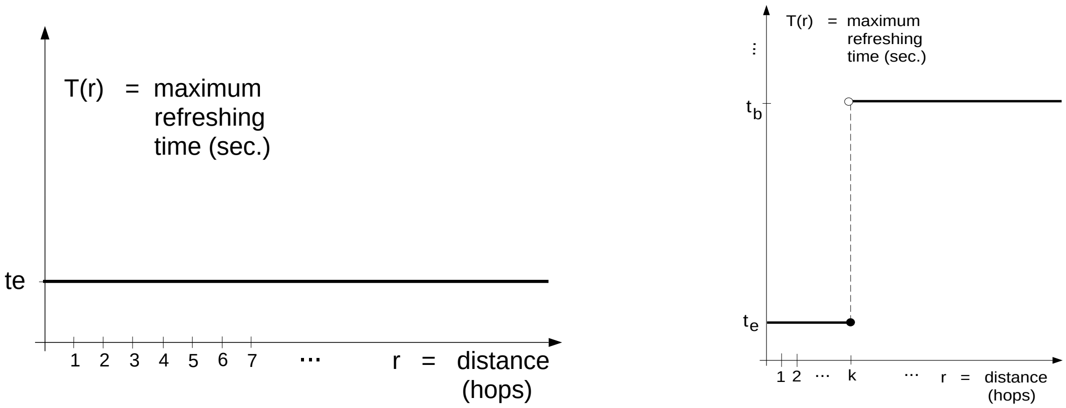


Figure 3.6: DLS's (left) and NSLS's (right) maximum refresh time $T(r)$ as a function of distance from link event.

Another member of the FSLS family is NSLS, obtained by setting $s_i = k$ for $i < p$ and $s_p = \infty$ (for some p integer), as shown in Figure 3.5 right. [8] In NSLS, a node receives information about changes in link status from nodes that are less than 'k' hops away (i.e. inside its sight area), but it is not refreshed with new link state updates from nodes out-of-sight. Suppose that initially, a node has knowledge

Request To Send (RTS)/Clear To Send (CTS)-based MAC with long channel acquisition and turn around times, the number of packets transmitted has a greater impact on the network capacity than the number of bits transmitted. In addition, GSR recovers faster than DLS from network partitions, especially under low mobility.

[8] In DLS and NSLS, since the values of s_i are the same for all i, based in the more precise rule mentioned before, a node checks for link changes for the past t_e seconds only.

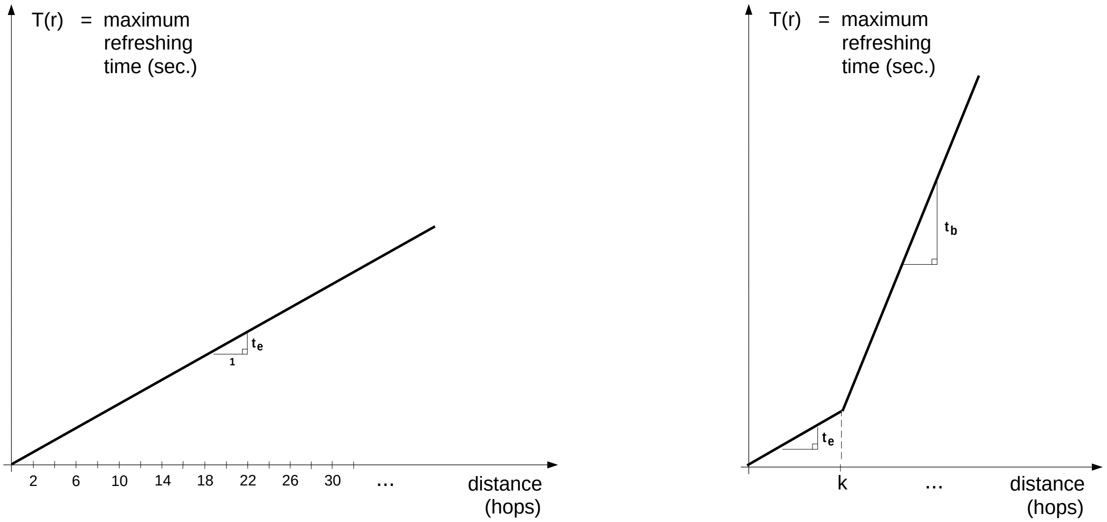


Figure 3.7: GSR's (left) and FSR's (right) maximum refresh time $T(r)$ as a function of distance from link event.

of routes to every destination. In NSLS, as time evolves and nodes move, the referred node will learn that the previously computed routes will fail due to links going down. However, the node will not learn of new routes becoming available because the out-of-sight information is not being updated. This problem is not unique to NSLS but it is common to every algorithm in the FSLS family. NSLS, however, represents its worst case scenario. To solve this problem, NSLS (and any algorithm in the FSLS family) uses the 'memory' of past links to forward packets in the direction it 'saw' the destination for the last time. As the packet gets to a node that is on the 'sight' of the destination, this node will know how to forward the packet to the destination. The above is achieved by building routes beginning from the destination and going backwards until getting to the source; without removing old entries that although inaccurate, allows tracing the destination.

NSLS has similarities with the Zone Routing Protocol (ZRP) [112]. ZRP is a hybrid approach, combining a proactive and a reactive part. ZRP tries to minimize the sum of the proactive and reactive overhead. In ZRP, a node propagates event-driven (Link State) updates to its k-hops neighbors (nodes at a distance, in hops, of k or less). Thus, each node has full knowledge of its k-hop neighborhood and may forward packets to any node on it. When a node needs to forward a packet outside its k-hop neighborhood, it sends a route request message (reactive part) to a subset of nodes (namely, 'border nodes'). The 'border' nodes have enough information about their k-hop neighborhood to decide whether to reply to the route request or to forward it to their own set of 'border' nodes. NSLS is similar to the proactive part of ZRP [112] without the reactive route search.

Also, there are similarities between NSLS and the Distance Routing Effect Algorithm for Mobility(DREAM) [109], with the difference that NSLS limits the LSU propagation based on the number of hops traversed, meanwhile DREAM limits the position update message's propagation based on the geographical distance to the source.

There are also similarities between NSLS and Fisheye State Routing (FSR) [9] [116]. FSR uses the same topology dissemination mechanism as GSR, but it does not transmit the whole topology information each t_{flood} seconds. Instead, only a short version including only the closest ('in scope') nodes entries is transmitted. A second, larger timer (t_{large}) is used to exchange information about out-of-scope nodes (the rest of the network). Setting $t_e = t_{flood}$ and $t_b = t_{large}$, and k such that all the nodes in-scope are k or less hops away, NSLS induces the same control overhead as FSR; however, the latency in updating link state information – as reflected in the function $T(r)$ – is greater in FSR than in NSLS. In NSLS, $T(r) = t_e$ for $r \leq k$, and $T(r) = t_b$ for $r > k$, as shown in figure 3.6 (right). In the other hand, in FSR, a LSU have to wait at most t_e seconds (in average $\frac{t_e}{2}$) to be propagated one more hop away from the node experiencing the link event while it is in scope ($r \leq k$), and wait t_b seconds when it is 'out-of-scope' (i.e. $r > k$). Thus, for FSR $T(r) = t_e * r$ for $r \leq k$, and $T(r) = k * t_e + (r - k) * t_b$ (see Figure 3.7 right), which is significantly larger than the values for NSLS.

Finally, the family of Fuzzy Sighted Link State algorithms is based on the observation that nodes that are far away do not need to have complete topological information in order to make a good next hop decision, thus propagating every link

[9] The same comments about the advantage of grouping LSUs in larger packets to reduce idle times during channel acquisition mentioned in GSR are applicable to FSR.

status change over the network may not be necessary. The sequence $\{s_i\}$ must be chosen as to minimize the total overhead (as defined in the previous section). The total overhead is greatly influenced by the traffic pattern and intensity. However, the choice of $\{s_i\}$ is solely determined by the traffic locality conditions. In the next sections, a uniform traffic distribution among all the nodes in the network is assumed and, as a consequence, the best values of $\{s_i\}$ were found to be equal to $\{s_i\} = \{2^i\}$, defining the Hazy Sighted Link State (HSLS) algorithm.

3.4 HSLS, the Optimal FSLS Approach

In this section, the best values of $\{s_i\}$ for the FSLS algorithm will be determined. These values will be the ones that minimize the total overhead. For this objective, an approximate expression for the total overhead induced by a tagged (typical) node will be derived. This expression will be derived by ignoring boundary effects, but the resulting $\{s_i\}$ will provide insight about the properties of the global solution, and will be applicable to the entire network.

In the next subsection (5.1) the network model and assumptions used on the analysis are introduced. Subsection 5.2 presents an approximate expression for the total overhead induced by a tagged node. Finally, the (likely) best sequence $\{s_i\}$ defining the Hazy Sighted Link State (HSLS) algorithm is derived in subsection 5.3.

3.4.1 Network Model

Let N be the number of nodes in the network, d be the average in-degree, L be the average path length over all source destination pairs, λ_{lc} be the expected number of link status changes that a node detects per second, λ_t be the average traffic rate that a node generates in a second (in bps), and λ_s be the average number of new sessions generated by a node in a second.

The following assumptions, motivated by geographical reasoning, define the kind of scenarios targetted on this work:

a.1 As the network size increases, the average in-degree d remains constant.

a.2 Let A be the area covered by the N nodes of the network, and $\sigma = N/A$ be the network average density. Then, the expected (average) number of nodes inside an area A_1 is approximately $\sigma * A_1$.

a.3 The number of nodes that are at distance of k or less hops away from a source node increases (on average) as $\Theta(d * k^2)$. The number of nodes exactly at k hops away increases as $\Theta(d * k)$.

a.4 The maximum and average path length (in hops) among nodes in a connected subset of n nodes both increase as $\Theta(\sqrt{n})$. In particular, the maximum path length across the whole network and the average path length across the network (L) increases as $\Theta(\sqrt{N})$.

a.5 The traffic that a node generates in a second (λ_t), is independent of the network size N (number of possible destinations). As the network size increases, the total amount of data transmitted/received by a single node will remain constant but the number of destinations will increase (the destinations diversity will increase).

a.6 For a given source node, all possible destinations ($N - 1$ nodes) are equiprobable and – as a consequence of a.5 – the traffic from one node to every destination decreases as $\Theta(1/N)$.

a.7 Link status changes are due to mobility. λ_{lc} is directly proportional to the relative node speed.

a.8 Mobility models : time scaling.

Let $f_{1/0}(x, y)$ be the probability distribution function of a node position at time 1 second, given that the node was at the origin $(0, 0)$ at time 0. Then, the probability distribution function of a node position at time t given that the node was at the position (x_{t_0}, y_{t_0}) at time t_0 is given by $f_{t/t_0}(x, y, x_{t_0}, y_{t_0}) = \frac{1}{(t-t_0)^2} f_{1/0}(\frac{x-x_{t_0}}{t-t_0}, \frac{y-y_{t_0}}{t-t_0})$.

Similarly, let $g_{0/1}(x, y)$ be the probability distribution function of a node position at time 0 second, given that it is known that the node position at time 1 will be $(0,0)$. Then, the probability distribution function of a node position at time $t < t_1$ given that the node will be at the position (x_{t_1}, y_{t_1}) at time t_1 is given by $g_{t/t_1}(x, y, x_{t_1}, y_{t_1}) = \frac{1}{(t_1-t)^2} g_{0/1}(\frac{x-x_{t_1}}{t_1-t}, \frac{y-y_{t_1}}{t_1-t})$.

Assumption a.1 follows since imposing a fixed degree in a network is desirable and achievable. It is desirable, because allowing the density to increase without bound jeopardizes the achievable network throughput. It is achievable, because there are effective power control mechanisms available [114]. In general, a topology control algorithm should attempt to make the density as small as possible without compromising (bi)connectivity.

Assumption a.2 is motivated by the observation that on large scales uniformity of node distribution is expected to increase. For example, it is expected that half the area covered by the network contains approximately one half of the nodes in the network. For a specific network topology this assumption may not hold; however, on average we expect this to be the case. This work focuses on expected (mean) behavior. Thus, although geographical reasoning may not define one hop connectivity (where multipath fading, obstacles, etc. are more important), it strongly influences connectivity as observed according to larger scales. One can talk about the 'geographical' and 'topological' regions. In the 'geographical' (large-scale) region, geographical-based reasoning shapes routing decisions. In the 'topological' region, it is the actual – and apparently arbitrary – link connectivity (topology) that drives the routing decisions, and geographical insights are less useful.

Assumptions a.3 and a.4 are based on assumption a.2. For example, consider a circular area centered at node S of radius R with n nodes in it. Doubling the area radius ($2R$) will quadruple the covered area, and therefore quadruple the number of nodes inside the area. On the other hand, the distance (in meters) from S to the farthest nodes will have only doubled, and assuming that the transmission range (after power control) of the nodes does not change, then the distance (in hops) will also double (on the average). Similarly, the 'boundary' area (where the nodes farthest away from S are) will increase linearly (as the circumference of a circle does) with the radius.

Assumption a.5 and a.6 are first order approximations motivated by observed behavior with existing networks; that is, as the network size increases the total amount of traffic generated by a single user typically diversifies rather than increases. For example, the availability of low-cost long distance service permits a user to speak with more family members and friends (wherever they are), but does not increase the total time the user has to spare for personal phone calls. Similarly, with the increase in size and content of the Internet, a user may find more web pages he/she would like to visit (destination set diversifies) but if the amount of bandwidth and time available for the user to connect is fixed, he/she will limit the total time (and traffic) spent on the Internet. Assumptions a.5 and a.6 are motivated by the behavior of human users, and some other networks may violate these assumptions. For example, in sensor networks each node may broadcast its information to all other nodes (causing λ_t to increase as $\Theta(N)$), or transmit to a central node (causing the destination set to consist of only 1 node, violating assumption a.6).

The traffic assumption is crucial to the analysis as it largely determines the effect of suboptimal routing on performance. For example, if traffic is limited to the locality of the source then hierarchical routing [105], ZRP [112], and NSLS will benefit. On the other hand, having a small set of destinations will favor algorithms such as DSR [107]. Uniform traffic tends to favor proactive approaches such as link state. In general, the effects of relatively equally distributed traffic tends to pose the most demanding requirements on a routing protocol. For this reason the analysis focuses on this case. Hence, assumption a.6 is not considered an unfair bias towards link state approaches. A protocol that is scalable (with respect to traffic) under assumption a.6, will also be scalable under any other traffic pattern. On the other hand, a protocol that is scalable under a localized traffic scenario, may fail when applied to a uniform traffic scenario.

Assumption a.7 stresses the importance of mobility. In particular, it is assumed that short-term variations in link quality can be offset by link control mechanisms, for example, by requiring a high fading margin before declaring a link up (so, small oscillations will not affect connectivity), or by waiting for several seconds before declaring a link down (so that short-lived link degradation will not trigger updates). The authors recognize that the wireless channel is quite unpredictable and long-lived link degradation is possible without mobility (e.g. in narrowband systems due to rapidly varying multipath fading caused by small displacement, obstructions, rain, etc.). Hence, mobility *will not always predominate.* However, the assumption is reasonable based on the previous justification and the assumed scenarios.

Assumption a.8 is motivated by mobility models where the velocity of a mobile over time is highly correlated. For example, this is the case if the unknown speed and direction are constant. This assumption does not hold for a random walk model; however, a random walk model would induce smaller node displacements over time ($\Theta(\sqrt{t})$, whete t is the elapsed time, since randomness tends to cancel out), and consequently they impose a less demanding scenario for routing protocols. Again, the objective is to focus on the most demanding scenario (that is, larger displacements) and assumes that the speed and direction are random processes with a slowly decaying autocorrelation function, which justifies assumption a.8.

3.4.2 Approximate Expression for the Total Overhead

The following expression for the total overhead induced by a tagged node S runnning a generic FSLS algorithm under high mobility has been derived in Appendix A, and is reproduced here for clarity :

$$\begin{aligned}
S_{pro} &= \frac{c\, size_{LSU}}{t_e}\left(\sum_{i=1}^{n-1}\frac{s_i^2}{2^i}+\frac{R^2}{2^{n-1}}\right) \\
S_{sub} &= \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma\ell}{4}\mathcal{M}R^2 t_e[2^{n-1}ln(R)-\sum_{i=1}^{n-1}2^{i-1}ln(s_i)] \\
S_{total} &= S_{pro}+S_{sub} \qquad (3.1)
\end{aligned}$$

where $\{s_i\}$, t_e, λ_t, σ, and N have been defined before. R is the network radius (distance, in hops, to the node farthest apart), $ln()$ is the natural logarithm function, c (β) is the constant relating the number of nodes at a distance k or less (exactly k) from node S with k^2 (k). $size_{LSU}$ is the average size (in bits) of a LSU packet. $\mathcal{M}$ is a constant that represents mobility, ℓ is related the transmission range of a node, α is the distance between S and its closest neighbors, γ is a constant whose value is in $< 1, 3 >$, and n is the smallest integer such that $2^n \geq R$.

The above equation was derived under the assumption that the tagged node S is located in the center of a network of radius R . This assumption allowed for a tractable model, although the resulting expressions prove to be dependent on the particular value of R and in general, on the boundary conditions. However, the posterior analysis of the nature of the solution for $\{s_i\}$ suggests that the solution found is still valid for non-typical nodes (nodes not in the center of the network), as will be seen in the next subsections.

3.4.3 Minimizing Total Overhead : The Hazy Sighted Link State (HSLS) Algorithm.

The selection of the best algorithm in the FSLS family reduces to minimizing equation 3.1 subject to the constraints that t_e be real positive, $\{s_i\}$ be a non-decreasing integer sequence, where $s_1 \geq 1$, and $s_{n-1} \leq R$. Note that n in equation 3.1 is not defined but it is also a variable. To solve the above problem, first a lower bound on the total overhead is obtained by relaxing the integer condition on s_i. Next, an integer (feasible) solution is proposed and compared to the lower bound. The proposed solution is less than 1% greater than the lower bound for $2 \leq R \leq 500$, and therefore it is considered the probably optimal solution to the integer problem.

A Relaxed Solution: Lower Bound

Assume that s_i may assume any real value greater than or equal to 1. Now, let's for a moment fix the value of n. Then using the lagrange multipliers method the following is obtained for s_i:

$$\frac{\partial}{\partial s_i} S_{total}(s_1, s_2, \ldots, s_{n-1}, t_e) \quad = \quad \frac{c\, size_{LSU}}{t_e} 2^{1-i} s_i - \frac{\lambda_t}{N} \frac{\alpha\beta\gamma\sigma\ell}{4} \mathcal{M} R^2 t_e \frac{2^{i-1}}{s_i}$$

thus, the condition $\frac{\partial}{\partial s_i} S_{total}(s_1, s_2, \ldots, s_{n-1}, t_e) = 0$ (for $i = 1, 2, \ldots, n-1$) implies $s_i = K * 2^{i-1}$, where

$$K \quad = \quad \sqrt{\frac{\lambda_t \alpha\beta\gamma\sigma\ell\mathcal{M}R^2}{4Nc\, size_{LSU}}}\, t_e \tag{3.2}$$

Also, it should be noted that if $K * 2^{i-1} < 1$ then $\frac{\partial}{\partial s_i} S_{total}$ is positive for all $s_i \geq 1$, and therefore the minimum is achieved for $s_i = 1$. Similarly, if $K * 2^{i-1} > R$, $\frac{\partial}{\partial s_i} S_{total}$ is negative for all $s_i \leq R$, and therefore the minimum is achieved for $s_i = R$. Finally, the optimality condition becomes :

$$s_i \;=\; \max\{\, 1,\ \min\{R,\ K * 2^{i-1}\}\,\} \tag{3.3}$$

In addition, the condition $\frac{\partial}{\partial t_e} S_{total} = 0$ implies $S_{pro} = S_{sub}$, which after regrouping terms becomes:

$$\begin{aligned} E_1 &= K^2 E_2 && (3.4) \\ & && (3.5) \end{aligned}$$

where

$$E_1 \;=\; \sum_{i=1}^{n-1} \frac{s_i^2}{2^i} + \frac{R^2}{2^{n-1}} \tag{3.6}$$

$$E_2 \;=\; 2^{n-1} ln(R) - \sum_{i=1}^{n-1} 2^{i-1} ln(s_i) \tag{3.7}$$

Note that equations 3.3, 3.4, 3.6, and 3.7 define a system of equations that can be solved numerically as long as the values of n and R are known. Finally, by using the relationship between t_e and K (equation 3.2) in the optimal overhead expression the following is obtained:

$$\begin{aligned} S_{total} &= 2\, S_{proactive} \\ &= \sqrt{\frac{\lambda_t \alpha\beta\gamma\sigma\ell\mathcal{M}R^2 c\; size_{LSU}}{N}}\; \frac{E_1}{K} && (3.8) \end{aligned}$$

The above set of equations (from 3.3 to 3.7) is solved numerically for $R = 2, 3, \ldots, 500$ and for increasing values of n up to the point where incrementing n does not reduce the total overhead. [10] Thus, for each R, the best ratio $\frac{E_1}{K}$ obtained is recorded. This value will be all that is needed to compare the lower bound on total

[10] What happens in those situations is that $s_i = R$ for all $i > n_0$ for some n_0.

overhead derived here and the actual value achieved by the integer (feasible) solution presented in the next subsection (HSLS).

It should be noted that special care is needed since there are several local minima close in numerical value. To understand this, consider two possible solutions with $(K', t'_e) = (1, t_1)$ and $(K'', t''_e) = (2, 2 * t_1)$. These solutions differ only in that the first solution is sending extra LSUs with TTL equal to 1 every other t_1 interval. LSUs with TTL equal to 1 will have a minimum impact on the total overhead expression, that is dominated by the LSUs sent/received from/to nodes far away. Note also that it is numerically more reliable to compute $\frac{E_1}{K}$ using the relationship $\frac{E_1}{K} = \sqrt{E_1 E_2}$, where K is chosen to minimize $\sqrt{E_1 E_2}$.

HSLS : An Integer (Feasible) Solution

While solving the LP relaxed problem, it has been noticed that the total overhead is somewhat insensitive to variations in K. What determines the goodness of the solution is the constant ratio of 2 between consecutive values of s_i. Typically, the values of K were between 1.5 and 3, so we explore the performance degradation (compared to the relaxed case) experienced when K is fixed and equal to 2.

By setting $s_i = 2^i$ for $i = 1, 2, \ldots, n-1$, where n is the lowest integer such that $2^n \geq R$, only the minimization with respect to t_e is needed:

$$\begin{aligned} S'_{total} &= \min_{t_e} \{ \frac{c\, size_{LSU}}{t_e} E'_1 + \frac{\lambda_t}{N} \frac{\alpha\beta\gamma\sigma\ell}{4} \mathcal{M} R^2 \, E'_2 \, t_e \} \\ &= \sqrt{\frac{\lambda_t \alpha\beta\gamma\sigma\ell \mathcal{M} R^2 c\, size_{LSU}}{N}} \sqrt{E'_1 E'_2} \qquad (3.9) \end{aligned}$$

where the prime suffix indicates a quantity associated with the integer (feasible) solution $s_i = 2^i$. E'_1, and E'_2 are computed according equations 3.6 and 3.7 respectively, but with the values of $s_i = 2^i$. Thus, these quantities become :

$$E'_1 = 2^n - 2 + \frac{R^2}{2^{n-1}} \qquad (3.10)$$

$$E'_2 = (2^{n-1} - 1) ln(2) + 2^{n-1} ln(\frac{R}{2^{n-1}}) \qquad (3.11)$$

and the value of t_e that achieves this minimum is :

$$t_e^{min} \quad = \quad \sqrt{\frac{4c\, size_{LSU} N}{\lambda_t \alpha\beta\gamma\sigma\ell\mathcal{M}R^2}\,\frac{E_1'}{E_2'}} \tag{3.12}$$

Finally, the relative difference between the lower bound (relaxed solution) and the feasible (integer) solution is equal to :

$$\delta \quad = \frac{S_{total}^{integer} - S_{total}^{relaxed}}{S_{total}^{relaxed}} \quad = \frac{\sqrt{E_1'E_2'} - \sqrt{E_1E_2}}{\sqrt{E_1E_2}}$$

In the interval $R \in [2, 500]$, the relative difference is oscillating with increasing R, but it is always less than 0.7018%. Thus, it may be stated that the solution $s_i = 2^i$ is nearly optimal in the sense that it is less that 0.7018% away from the lower bound derived in the previous subsection.

HSLS Algorithm Description and Non-central Nodes Discussion

In the previous subsections, it has been determined that choosing $s_i = 2^i$ will probably minimize the total overhead induced by a node. This assignment ($s_i = 2^i$) is referred to as the Hazy Sighted Link State (HSLS) algorithm. HSLS's generation process can be obtained by replacing $s_1, s_2, s_3, s_4, \ldots$ by $2, 4, 8, 16, \ldots$ respectively in Figure 3.3. HSLS's maximum 'refresh' time function is shown in Figure 3.8. It can be noted that there is an almost linear relationship between $T(r)$ and r. This linear relationship is responsible for HSLS's probable optimality for the central node studied in the previous subsection. This relationship reflects the fact that when forwarding packets to nodes far away, it is the angular displacement what really matters.

Thus, HSLS successfully balances refresh periods and distances, so that the probability of making a suboptimal (bad) next hop decision is roughly the same for every destination independently of the distance [11]. This balance is natural (avoiding 'hard' boundaries as in NSLS where a value has to be provided for k, the 'sight' area), and is typical when solving real life problems. It is the linear relationship between $T(r)$ and r what makes HSLS the winner algorithm regarding the centrally located

[11] Strictly speaking, the probability of a suboptimal (bad) next hop decision oscillates between the maximum and the minimum values as the distance to the destination increases.

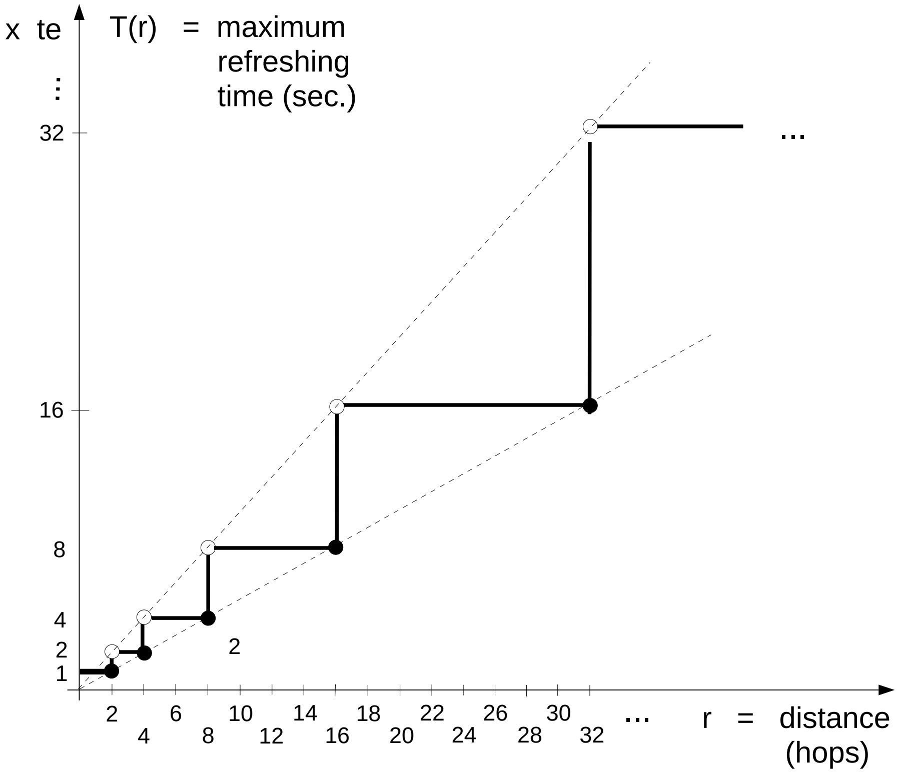


Figure 3.8: HSLS's maximum refresh time as a function of distance from link event.

node analyzed in the previous subsections. This property is kept when dealing with non-central nodes, so HSLS is expected to also be the winner FSLS algorithm when applied to a particular non-central node, and when considering the aggregation of all the nodes in the network. Then, the HSLS algorithm pseudo-code is provided in Figure 3.11. Note that the pseudo-code is slightly more complex than the discussion. It is because the discussion has focused on highly mobile scenarios. HSLS, however, adapts to slow varying scenarios, behaving like SLS when the rate of topological change is small (SLS mode in Figure 3.11). Also, the previous analysis – based on geographical reasoning – fails to capture the dynamics inside the 'topology region', that is, small scales. For practical implementations it was found through simulations that LSUs with small TTL do have a great impact on the algorithm performance. Level 1 LSUs do not induce much proactive overhead (just $\Theta(N)$) but they help to reduce loops and time to reaction to failures. So, every HSLS implementation should include them. [12] Further discussion about implementation issues is deferred to section 3.5.

3.4.4 HSLS Dependence on Size, Mobility and Traffic

Equations 3.10 and 3.11 can be rewritten in function of a factor $f = \frac{R}{2^{n-1}} \in (1, 2]$ as:

$$\begin{aligned} E'_1 &= (f + \tfrac{2}{f})R - 2 &&= \Theta(R) \\ E'_2 &= \tfrac{ln(2f)}{f}R - ln2 &&= \Theta(R) \end{aligned}$$

And applying the above expressions on equation 3.9 (after simplification due to the fact that $cR^2 = N$ and $\sigma \approx \frac{1}{\alpha^2}$) the following expression is obtained:

$$\begin{aligned} S'_{total} &= \sqrt{\gamma\,(\beta\frac{\ell}{\alpha}size_{LSU})\,\lambda_t\mathcal{M}}\,\sqrt{E'_1E'_2} \\ &= \Theta(\,(\frac{\ell}{\alpha})^{2.5}\,\sqrt{\lambda_t\mathcal{M}}\,R\,) \end{aligned}$$

where the last equality holds since β and $size_{LSU}$ increases linearly with the node degree d, and the node degree d increases as rapidly as $(\frac{\ell}{\alpha})^2$.

Thus, recalling that $R = \Theta(\sqrt{N})$ and adding up the overhead contribution from all the N nodes in the network, the following expression for HSLS total overhead is obtained :

$$HSLS_{total} \;=\; \Theta((\frac{\ell}{\alpha})^{2.5}\lambda_t^{0.5}\mathcal{M}^{0.5}N^{1.5}) \tag{3.13}$$

The above expression shows that HSLS present excellent scalability properties, since it not only scales as well as (or better than) HierLS with respect to the network size N, but also scales better than it with respect to mobility (HierLS total overhead is linear with mobility). It also shows good scalability with respect to traffic, since it is not linear (as DSR, flooding, and HierLS) but increases only as rapidly as $\sqrt{\lambda_t}$. A more detailed analysis may be found in Chapter 4.

It is also interesting to note the dependence of the total overhead from the ratio between the node transmission range and the actual minimum distance between

[12] In our implementation, HELLO messages exchanged between one hop neighbor (for neighbor/link discovery) played the role of LSUs with TTL equal to 1. Thus, no extra transmission of LSUs with TTL equal to 1 was necessary.

nodes. It may be noticed that as the transmission range increases (incrementing the node degree) the total overhead induced increases. This fact, combined with the fact that increasing the node degree reduces the efective throughput per node, points to the importance of limiting the nodes' transmission power to the minimum point where good connectivity is achieved.

Similarly, regarding the value of t_e that achieves the minimum overhead, it can be shown (from equation 3.12, and recalling that c also increases linearly with the node degree, i.e. c is also $\Theta((\frac{\ell}{\alpha})^2)$) that $t_e = \Theta(\sqrt{\frac{1}{\lambda_t \mathcal{M}}}(\frac{\ell}{\alpha})^{1.5})$. Thus, the optimal value of t_e is asymptotically independent of the network size depending only on the traffic, mobility, and transmission range. Thus, it is possible to set a value of t_e that works well independently of the network size.

3.5 HSLS's Implementation Issues

So far, several assumptions has been made during the analysis. In order for this assumptions to be valid, some additional mechanisms must be in place. These mechanisms are discussed next. Then, this section finishes with an algorithmic description of HSLS.

3.5.1 Loops and Level 1 LSUs

When a packet is far away from the destination, errors in the next hop decision are easily corrected without resending the packet backwards, because there are a lot of alternative minimum distance paths available. This is a reasonable assumption when employing min-hop routing in a network that is not pathologically sparse. The presence of alternate paths follows a 'geographical region' reasoning. We expect the packet to find its way as the water flow from the mountains to the sea. The packet, as the water, may change directions from time to time, but it always get closer to its destination.

Unfortunately, as the packet gets closer to the destination the above assumption no longer holds true. The packet is now in the 'topology' region, where arbitrary topologies determine the effectiveness of a routing approach. For example, sending the packet to a node that was a neighbor of the destination, but has recently lost its link to the destination, will be the equivalent to going into a dead-end street where the only choice is to go back. In a network, however, going back will result in a temporary loop where the packet will travel back and between two nodes.

Thus, it is important that nodes in the 'topology region' has proper topology information to avoid loops. But how large is the topology region?. While it is true that in sparse networks the topology region may include several hops, in our simulations, for average node degrees between 4 and 12, the topology region has typically included just 2 hops. That is, the most frequent cause of looping have been nodes that lose their links to the destination. We can see that these loops can be succesfully avoiding by providing quick reaction to link changes in the 2-hop neighborhood, which is easily achieved by propagating LSUs with TTL equal to 1 (level 1 LSUs) each time a link change is detected.

Morever, it was detected that one of the main causes for packets being dropped was the transmission over low-quality links. It should be noted that when minimum hop routing is employed, the routes chosen tend to include 'long' links (link where the extreme points – nodes – are far apart). As nodes move, the long links are the ones more likely to break. Thus, it is not surprising that for long paths at any given time there is high probability that some of the links in the path is no longer useful or have degraded quality (requiring retransmissions). Level 1 LSUs have the additional advantage of being able to quickly inform about link degradation, allowing intermediate nodes to switch the traffic to a different path. Thus, level 1 LSUs are an inexpensive way of solving both the short loops issue and the packet dropping due to low-quality links. It should be noted that a level 1 LSU requires only one transmission (no retransmission is necessary).

Thus, in HSLS level 1 LSUs are send frequently. In our particular implementation, however, there was no need for sending these LSUs explicitly, since the HELLO packets exchanged by the neighbor discovery module (link layer) provided these information in a timely manner. The HELLO packets contained the id of the node sending the beacon, as well as a list of the neighbors that node could listen to. For each neighbor, and indicator whether the link was unidirectional (incoming) or bidirectional (incoming and outgoing) was included, along with the number of beacons received from that neighbor during the last time window. A node receiving the HELLO message will not only determine if it has an incoming link from the node sending the beacon, but it may look for its own id in the node list of neighbors, and if present, determine that the link to/from that neighbor is bidirectional. The indication of the number of beacons received, when eavesdropped by the routing module, allows to quickly detect degrading links and to use alternative paths when available. Thus, quick reaction to local changes was achieved.

In general, for applications where the separation between layers (i.e. network and link layers) is enforced, the network layer must send level 1 LSUs at least every time a link change is detected, and in the best case they should be sent periodically, with a small time interval.

Currently, there is no mechanism in HSLS to avoid short-lived long loops (there are not long lived loops, since eventually al the nodes will recieve new link-state information). These loops were detected and removed in our simulations by means of link layer techniques (number of hops traversed, for unreliable MACs; and unique packet id for reliable MACs).

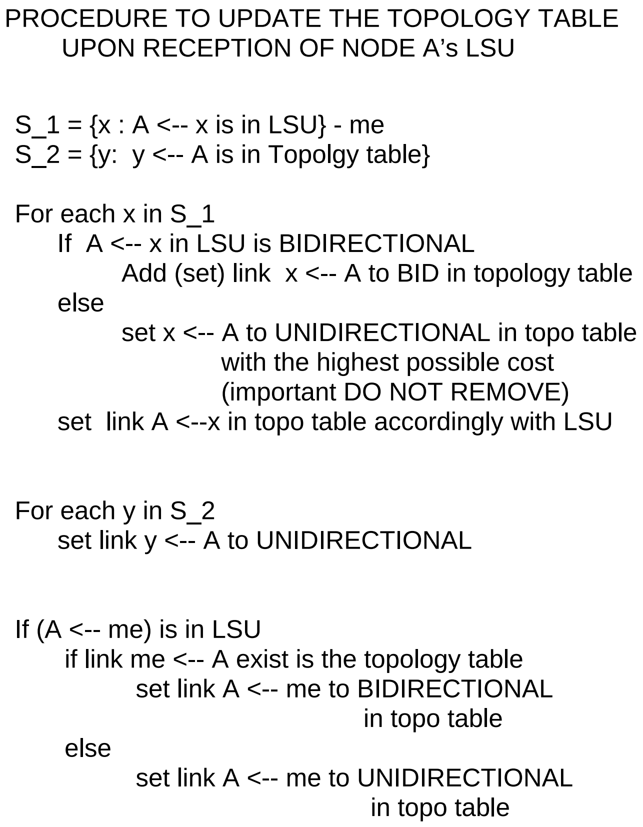
3.5.2 Topology Table Maintenance

Under SLS, each node learns about a link going down by means of 2 LSUs - one per each node at each extreme of the link. In HSLS, however, this is no longer the case. Due to the limited propagation of the LSU, a node may only receive the LSU sent for the closest node, or not at all (until a later time).

Thus, it is important that a node 'extracts' the most information from the LSU. For example, when the LSUs reports status of incoming links (i.e. when supporting unidirectional links), reception of a LSU from a node, say A declaring an incoming link from another node B may provide also information about incoming links to B. If node A's LSU declare the incoming link from node B as bidirectional (node A is aware of its one hop neighborhood topology), thus it must be inferred that node B

also has a bidirectional link from node A. Alternatively, if the link from node A is declared to be unidirectonal (incoming only), thus it is inferred that node B does no longer have an incoming link from node A, and if such an entry exists in the topology table's entry for node B, it should be removed. In this case, we are relying on the information conveyed by node A since it has more up-to-date information than us (because it is closer – one hop away – from node B, and receiving level 1 LSUs or beacons).

Figure 3.9 presents the pseudocode employed in our implementations for updating the topology table upon reception of a LSU from node A. It is based on LSUs conveying incoming link information in order to support unidirectional links (however, unidirectional links were assigned a larger cost than bidirectional ones, and were used only if there were no bidirectional path available). Two details should be observed. First, since the main principle is to relay on the information provided by the node with more up-to-date information, priority is given to the receiving node's local neighborhood information. Thus, any conflict between the information provided by node A, and the receiving node's local information (as provided by level 1 LSUs and beacons) is resolved in favor of the receiving node's local information, which is more up-to-date since the beacons mentioned in the previous subsection are interexchanged frequently. Note, however, that there is no conflicting information, as for example a report from node A about an unidirectional incoming link from the node executing the algorithm. The executing node's local information, coming from the beacons it received, does not provide info about that node's unidirectional outgoing links. The second detail is more subtle, but not for that less important. Note that there are situations where we learn that a link to a third node, say C, is no longer there and should be removed. However, since we may not receive an update from node C for a while we may not be able to find a new, alternative route to it. However, nodes closer to node C will have fresher information and may be able to find routes to it. Thus, the solution is to send the packets destined to node C in the direction node C used to be, and hope that along the way a node with current information deliver the packet. The above is accomplished by not removing the link from node A to node C but elevating its cost to the maximum allowed value, so that that link will not be used



```
PROCEDURE TO UPDATE THE TOPOLOGY TABLE
      UPON RECEPTION OF NODE A's LSU


 S_1 = {x : A <-- x is in LSU} - me
 S_2 = {y:  y <-- A is in Topolgy table}

 For each x in S_1
     If  A <-- x in LSU is BIDIRECTIONAL
            Add (set) link  x <-- A to BID in topology table
     else
            set x <-- A to UNIDIRECTIONAL in topo table
                       with the highest possible cost
                       (important DO NOT REMOVE)
     set  link A <--x in topo table accordingly with LSU


 For each y in S_2
     set link y <-- A to UNIDIRECTIONAL


 If (A <-- me) is in LSU
      if link me <-- A exist is the topology table
             set link A <-- me to BIDIRECTIONAL
                                    in topo table
      else
             set link A <-- me to UNIDIRECTIONAL
                                    in topo table
```

Figure 3.9: Pseudocode for topology table update upon reception of a LSU from node A.

if there is an alternative path. However, in the case where there are no alternative paths toward node C (the case we were woried about), the link from node A to node C will be used by route computation algorithm and determine that packets to node C be forwarded in the direction of nodes C and A, as desired. Thus, keeping those links (e.g. link A to C) allows to keep some memory and track a node shadow as it moves farther away from us.

3.5.3 Low Mobility Scenarios

So far, the discussion has been focused on highly mobile scenarios. It is beenm shown in equation 3.13 that HSLS total overhead increases as $\Theta(\sqrt{\mathcal{M}}) = \Theta(\sqrt{\lambda_{lc}})$, which is a significant improvement over SLS whose control (total) overhead increases linearly with λ_{lc}.

However, for small values of λ_{lc}, the square root function is not necessarily better than the linear function. For small values of λ_{lc}, linear is actually better (smaller) than $\Theta(\sqrt{\lambda_{lc}})$. The above is not a main concern since the main focus is in the protocol survivability under high stress conditions, meanwhile for low stringent conditions almost any protocol will suffice. However, it will be even nicer if a protocol may also achieve the best performance under not stressful scenarios.

To fulfill those requirements, Adaptive HSLS (A-HSLS) was created. A-HSLS design was motivated by the observation that under extreme low mobile scenarios, when link changes were unfrequent, HSLS will require to send a series of LSUs with TTL equal to 2, 4, 8, and so for until a global LSU was sent, for each link change. In those cases it would have been more efficient to just send a global LSU to begin with, had we know that no other link change would follow. Unfortunately, there is no way to know for certain whether more link state changes will follow a particular one or not.

However, it may possible to estimate the 'state' (low or high) mobility a network'is base on previous experience and to assume that behavior will be repeated in the immediate future. A-HSLS does just that. If the interval between the time the last global LSU was sent and the time of the last link status change exceeds a threshold, it is assumed that the node is in the low mobility mode and no more link changes will follow in the immediate future, therefore a global LSU is sent (as in SLS). In the other hand, if the time elapsed is smaller than the threshold, the LSU will be sent according to HSLS rules. The threshold is set to the inverse of the rate of link change that induce the same control (proactive) overhead under SLS and HSLS.

Under SLS, a node will generate LSUs at a rate of λ_{lc} LSUs per seconds, and each LSU will require N retransmissions. Thus, the node will induce a control overhead of $\lambda_{lc}N$ LSUs per second. Under HSLS, the control (proactive) overhead induced by a single node, say X, is roughly $\frac{N}{R_x t_e}(1+2f_x)$ LSUs per second (see subsection 4.8.1). Where R_x is the time index at which node X sends global LSUs, that is $R_x < MD_x \leq 2R_x$, where MD_x is the distance from node X to the node farthest apart according to node X's topology table. At time $R_x t_e$ node X will have to send a LSU with TTL equal to $2R_x$ and will notice that the LSU will reach all the nodes in the network, then node X overwrites the TTL value with a predefine value interpreted as infinity (i.e. not to be decreased) and re-initiate the algorithm. f_x is the ratio between the number of nodes that received a LSU with original TTL set to R_x (second largest TTL) and the number of nodes in the network (nodes that received the global LSU). f_x may vary from 0.25 to 1, but an average value of 0.5 will be assumed here (according to our observations this is the typical case).

Thus, equating both proactive overhead one obtain the following equation :

$$\begin{aligned}\lambda_{lc} N &= \frac{N}{R_x t_e}(1 + 2f_x)\\ \lambda_{lc} &= \frac{2}{R_x t_e}\end{aligned}$$

Thus, the rate of link change (λ_{lc}) at which HSLS and SLS induce the same proactive overhead is $\lambda_{lc} = \frac{2}{R_x t_e}$, and therefore the threshold used by A-HSLS is equal to $\frac{R_x t_e}{2}$.

Finally, in Appendix B it is shown that A-HSLS control overhead induced by a node is equal to equal to $\frac{1+2pf_x}{pR_x t_e + \frac{1}{\lambda_{lc}}} N$, where $p = 1 - \exp^{-\frac{\lambda_{lc} R_x t_e}{2}}$. Figure 3.10 compares this control overhead with SLS's and HSLS's for the case $f_x = 0.5$. It may be seen that SLS's and HSLS's control overhead coincide for $\lambda_{lc} = \frac{2}{R_x t_e}$. It can be noted that even for extremely low rate of link changes, A-HSLS induces the similar control overhead as SLS. For not-so-small rates of change, however, A-HSLS tends to converge to $\frac{2N}{r_x t_e}$, while SLS control overhead increases unbounded. In high mobility scenarios, A-HSLS proactive overhead will be equal to the value derived in subsection 4.8.1 assuming high mobility scenarios.

Finally, it should be noted that as network size increases (and individual nodes link change rate remains fixed) , the point $\frac{2}{R_x t_e}$ get smaller, and A-HSLS will tend to be in HSLS mode most of the time.

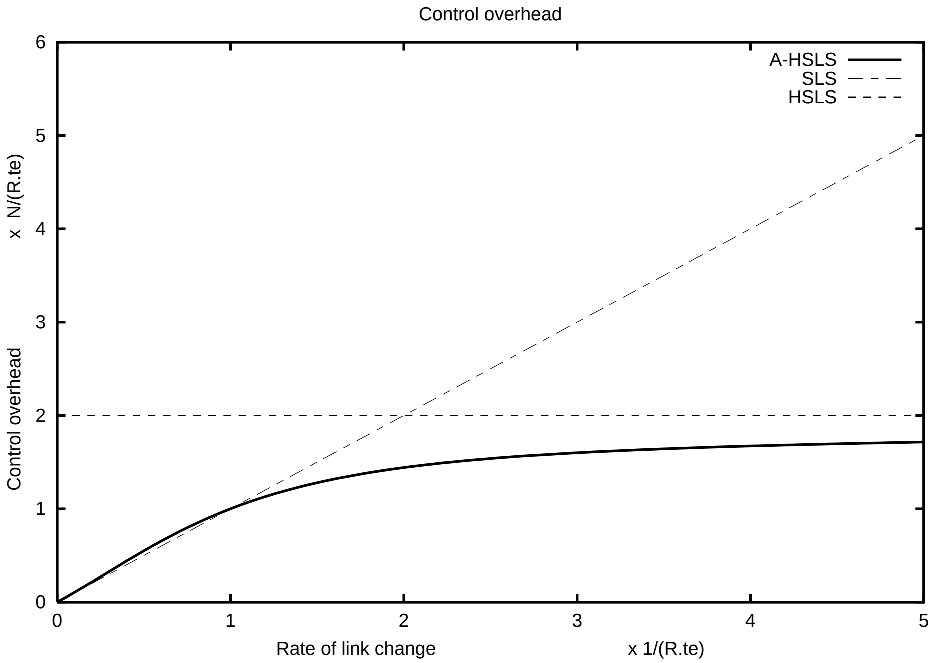


Figure 3.10: Control (proactive) overhead for A-HSLS, SLS, and HSLS

.

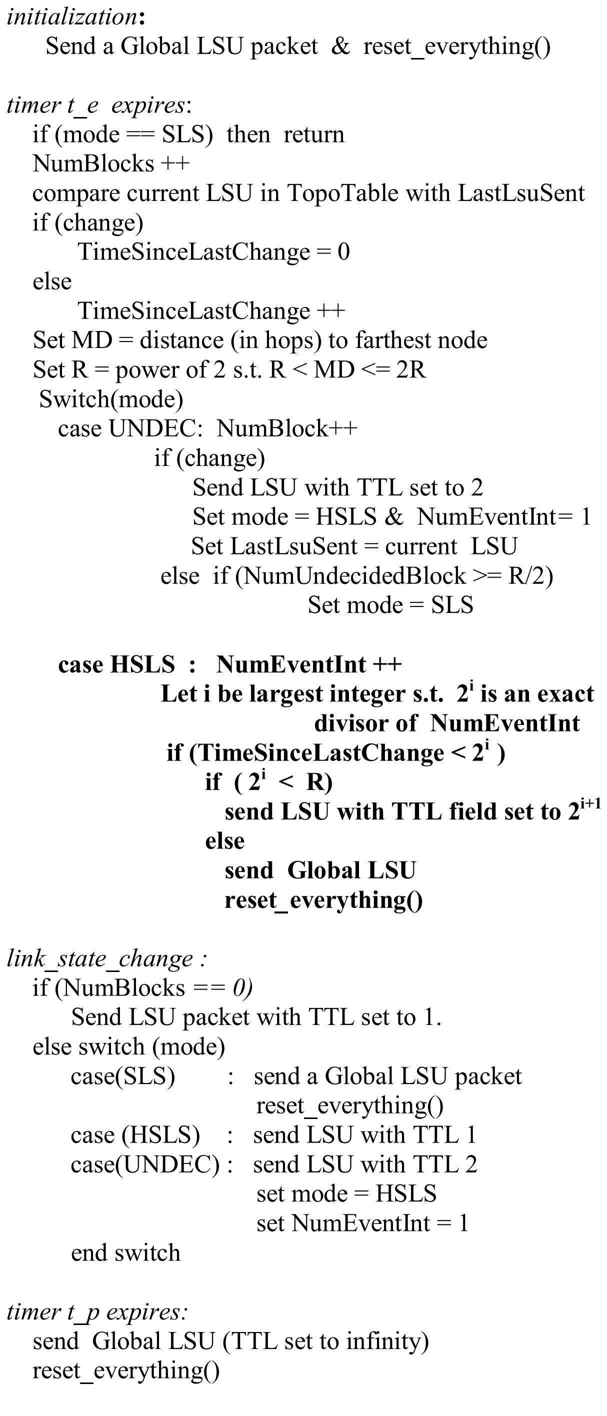
3.5.4 Protocol Description

After the last subsections discussion, we are finally in position to discuss HSLS (more precisely A-HSLS) pseudo-code shown in Figure 3.11.

HSLS comprises three main functions: LSU generation, LSU dissemination, and topology table maintenance.

Topology table maintenance pseudo-code (Figure 3.9 has been discussed in subsection 3.5.2, thus, further discussion is not necessary.

LSU dissemination consists in resent a received LSU after decreasing its TTL value by one. Of course, if TTL is equal to 0 (was received with a TTL equal to 1) it is not resent. Also, if the TTL value is equal to a predefined value meaning 'infinity", that value is not decreased. Duplicated LSUs (as recognized by their sequence number) are discarded.



```
initialization:
     Send a Global LSU packet  &  reset_everything()

timer t_e  expires:
   if (mode == SLS)  then  return
   NumBlocks ++
   compare current LSU in TopoTable with LastLsuSent
   if (change)
         TimeSinceLastChange = 0
   else
         TimeSinceLastChange ++
   Set MD = distance (in hops) to farthest node
   Set R = power of 2 s.t. R < MD <= 2R
    Switch(mode)
      case UNDEC:  NumBlock++
                   if (change)
                        Send LSU with TTL set to 2
                        Set mode = HSLS &  NumEventInt= 1
                        Set LastLsuSent = current  LSU
                    else  if (NumUndecidedBlock >= R/2)
                                   Set mode = SLS

      case HSLS  :   NumEventInt ++
                    Let i be largest integer s.t.  $2^i$ is an exact
                                   divisor of  NumEventInt
                     if (TimeSinceLastChange < $2^i$ )
                          if  ( $2^i$  <  R)
                            send LSU with TTL field set to $2^{i+1}$
                          else
                            send  Global LSU
                            reset_everything()

link_state_change :
   if (NumBlocks == 0)
        Send LSU packet with TTL set to 1.
   else switch (mode)
        case(SLS)        :   send a Global LSU packet
                             reset_everything()
        case (HSLS)    :   send LSU with TTL 1
        case(UNDEC) :   send LSU with TTL 2
                             set mode = HSLS
                             set NumEventInt = 1
        end switch

timer t_p expires:
   send  Global LSU (TTL set to infinity)
   reset_everything()
```

Figure 3.11: Pseudocode description of the Hazy Sighted Link State (HSLS) algorithm.

Thus, the remainder of this subsection discusses the LSU generation algorithm presented in Figure 3.11.

Upon initialization (sending of the first global LSU and resetting of all counters and timers), a node running HSLS (A-HSLS) is in the UNDEC (Undecided) mode. After information about link change rates have been obatined the node will transit to either SLS (for Standard Link State -like) or HSLS mode. If the node is in SLS it means that it regards its local topology as slow changing and the next time a link change is detected it will send a global LSU as in the SLS algorithm. If the node is in HSLS mode, it will send LSUs with values $2, 4$, and so on at time instants that are multiple of t_e. It will also send level 1 (TTL equal to 1) LSUs for changes that happens between those time instants.

For example, consider the case that after initialization there has not been a link change for a while. Every t_e seconds, a timer expires and increases the *NumBlocks* counter by one. Also, the node will generate a shortest path first table according to Dijkstra's algorithm and will determine the distance MD to the node furthest away from itself. Based on MD, the value of R, the highest power of 2 such that $R < MD \leq 2R$, is also found. If the accumulated value of the *NumBlocks* counter is greater or equal than $R/2$, then the node assumes that its local neighborhood is in a low mobility scenario and it switches to SLS mode. Under SLS mode, the next link change detected will induce a global LSU and the node will switch back to UNDEC (Undecided) mode, waiting to estimate the mobility state of its local neighborhood once again. All timers and counters are reset.

In the other hand, if a link change is detected before Rt_e seconds have elapsed since the last global LSU, the node enters HSLS mode by sending a LSU with TTL equal to 2 and setting the *NumEventInt* counter to 1. This counter (*NumEventInt*) will be incremented by one each time the timer expires (each t_e seconds) while in HSLS mode. In HSLS mode, the node will propagate changes so that nodes that are less than 2 hops away are refreshed each t_e seconds, nodes that are less than 4 hops away are refreshed each $2t_e$ seconds, nodes that are less than 8 hops away are refreshed every $4t_e$ seconds, and so for. To achieve the behavior just described, a node computes the maximum power of 2 that is an exact divisor of the *NumEventInt*, let *window_len* be this value, thus for example if $NumEventInt = 20$, then $window\_len = 4$, since 4 divides 20 and 8 does not. Upon determination of $window\_len$, the node determines that it should check for link changes in the past $window\_len\, t_e$ seconds, and if any, it should send a LSU with TTL equal to $2\, window\_len$. Also, if the TTL value ($2\, window\_len$) is greater or equal to $2R$ (i.e. reaching the entire network), the TTL value is overwritten with the infinity value, thus a global LSU is sent and the algorithm is reinitiated, i.e. switches back to UNDEC mode, and reset all timers and counters. Thus, in figure 3.11, the node checks whether the *TimeSinceLastChange* is greater than $window\_len\, t_e$ seconds, and if not, it sends a LSU as explained before.

Figure 3.12 shows an example of (A-)HSLS's LSU generation process. An 'x' marks the time when link status changes are detected. A vertical arrow signifies that a LSU with the specified TTL was sent. For clarity sake, level 1 LSUs (discussed) next) are not shown. The example xhown in this figure assumes that MD is somewhere between 9 and 16, so that R is equal to 8. We can see that the first link change after initialization (global LSU sent) happens after $R/2 = 4$ time intervals, i.e. when the node is in SLS mode. Thus, the node send a global LSU (TTL $= \infty$) and it is reset. The next link change happens before the node enters SLS mode. Since the time elapsed between the global LSU transmission and the link change detection is greater than t_e seconds, the node does not wait any time before sending a LSU with TTL equal to 2. The next time interval, th enode wakes up and update the value of *NumEventInt* to be equal to 2. Th enode then checks for link changes in the last $2t_e$ seconds, and since there have been link status changes, it sends a LSU with TTL equal to 4. The next time interval, the value of *NumEventInt* is equal to 3, so th enode check for link status changes in the past t_e seconds. Since there was no change, no LSU is transmitted. For the next time interval, *NumEventInt* is equal to 4 so the node check for changes in the past $4t_e$ seconds and sends a LSU with TTL equal to 8. For the next time interval, *NumEventInt* is equal to 5 so the node check for changes in the past t_e seconds, and since there was no change, no LSU is transmitted. Continuing for *NumEventInt* equal to 6, the node checks for changes in the past $2t_e$ seconds, and since there has been a link change, it sends a LSU. For *NumEventInt* equal to 7, there were no link change in the past t_e seconds so no LSU is sent. Finally,

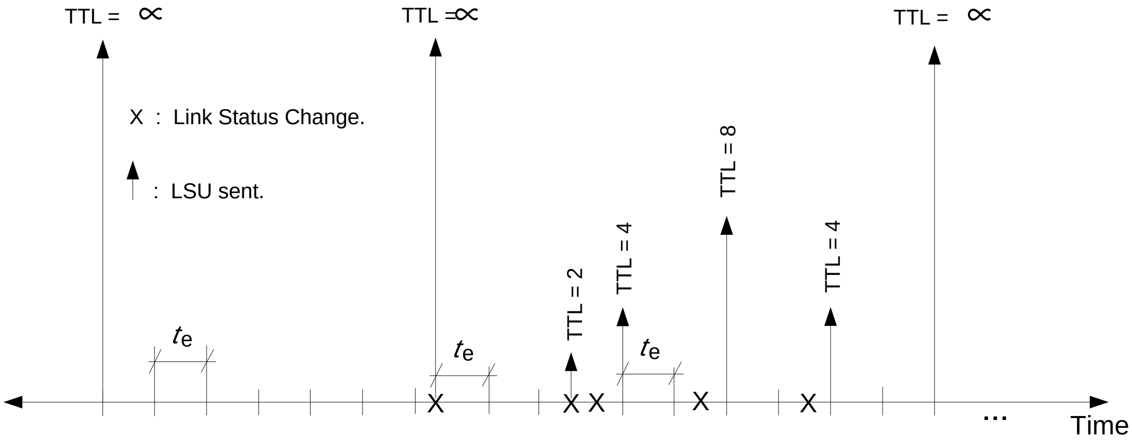


Figure 3.12: Example of A-HSLS's LSU generation process.

for *NumEventInt* equal to 8, the node checks for changes in the past $8t_e$ seconds, and since changes did happen it will send a LSU. However, since the corresponding TTL value 16 is greater than MD, the TTL field is overwritten with the value infinity and the algorithm is reinitiated in UNDEC mode.

Regarding the treatment of level 1 LSUs, while in HSLS mode each time the node detects a link change it will send a level 1 LSU. [13] The first link change detected (the node is still in the UNDEC mode) will receive special treatment. If the time elapsed between the global LSU transmission and the link change detection is greater than t_e, a LSU with TTL equal to 2 will be sent immediately and the node will switch to HSLS mode. However, if the elapsed time is lower than t_e (fast changing case), the node will wait until expiration of the t_e timer before sending the LSU with TTL equal to 2. A LSU with TTL equal to 1 will be send instead. This way, multiple changes will be collected before sending a more costly LSU.

Finally, a global LSU are sent if the time elapsed since the last global LSU transmitted exceeds a predefined threshold (not shown in Figure 3.12 for clarity sake. This is equivalent to the periodic LSUs in SLS.

[13] In our implementation, since the HELLO beacons were sent periodically, level 1 LSU were sent even with more often than just after link changes.

In the remainder of this dissertation the terms A-HSLS and HSLS will be used to referred to the algorithm (A-HSLS) just described. All our implementations refer to this algorithm.

3.6 Simulation Results

The relative performance of the HSLS algorithm compared to SLS, DLS, and NSLS [14] in an integrated system (including radio, channel, and traffic models) has been evaluated from high fidelity simulations conducted using the CPT++ protocol toolkit and OPNET. The performance metric of interest is the *throughput*, which is the percentage of packets successfully received. The *throughput* results reflect the dynamic interaction of several factors, among them the network load (data and total overhead), the suboptimality of routes (since packets traversing longer paths are more likely to experience a collision at some point along their route), link layer information latencies (e.g. having to wait t_e seconds to get information about a link gone down), routing inconsistencies due to different 'vision' of the network by different nodes, etc. Thus, it is of interest to assess the relative performance of HSLS and other algorithms under non-saturation scenarios. These results complements the previous theoretical analysis, where it was determined that HSLS induced a lower total overhead than other algorithms in the FSLS family and therefore will achieve a higher throughput (in number of bits) under saturation conditions.

The propagation model used in these simulations considered a power decay exponent of 4 with respect to distance (i.e. $received\_power = \Theta(\frac{1}{d^4})$, where d is the distance separating the receiver from the transmitter). The MAC layer used was CSMA (without RTS/CTS), which gave an unreliable link layer with low latencies and unidirectional link support. Thus, the throughput figures for large traffic loads tend to be small.

Simulations were conducted for networks up to 800 nodes. In all of them, nodes

[14] Unless stated otherwise, t_e was set to 10 seconds for all the algorithms (except SLS) and the sight radii for NSLS is set to $k = 2$. Periodic timers (inducing global LSUs) were adjusted as to induce comparable proactive overhead among NSLS and HSLS.

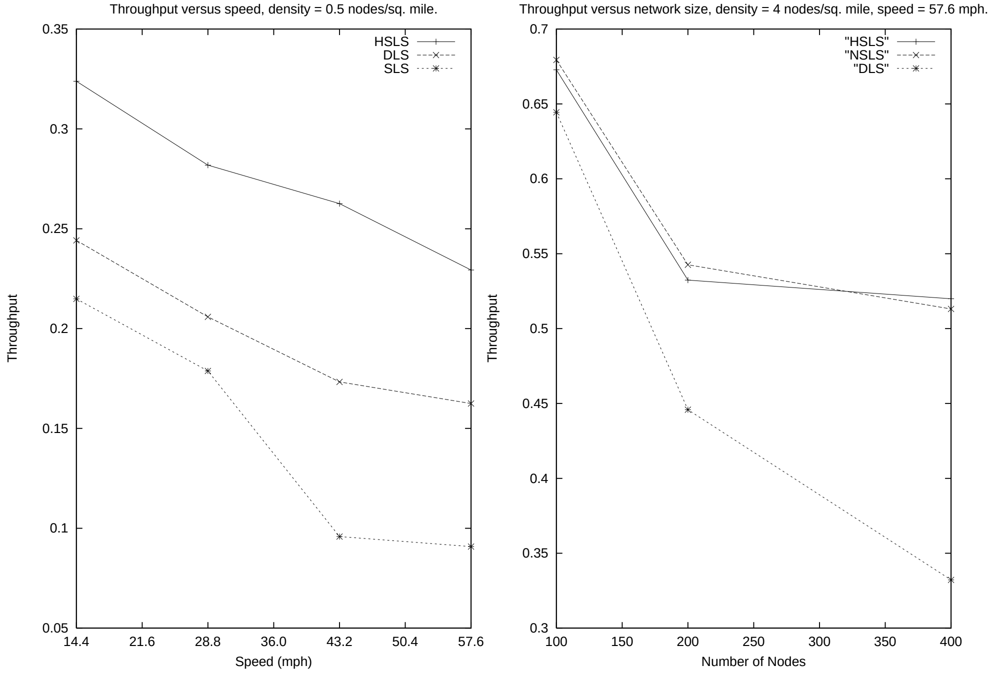


Figure 3.13: Throughput results for a 80-node network under different nodes' speed (left), and for different size networks (up to 400 nodes) (right).

were randomly located in a square area of varying size depending on the density parameter. Each node selected a random direction among 4 possible values and moved in that direction at maximum speed.

Figure 3.13 (left) shows simulation results obtained by CPT++ for a 80-node network with varying nodes' speed. The network density was set to 0.5 nodes per square mile. The radio link capacity was set to 300kbps, and there were 12 source-destination pairs chosen randomly. Each source generated 2048-bits packets with exponential interarrival time distributed around the mean of 1 packet per second (thus, there were 12 2Kbps streams). Figure 3.13 (left) compares DLS and HSLS with SLS. At this size (and for the given radio link capacity) the performance degradation of SLS – due to its scalability problems – is already noticeable. Thus, SLS was no longer considered for larger size simulations.

Next, the network size was increased up to 400 nodes in an OPNET simulation with 60 source-destination pairs (4 Kbps each). The radio link capacity was increased

Algorithm	Throughput
NSLS	0.3516
HSLS	0.4465

Table 3.1: Throughput for a 800-node network, density = 4 nodes/sq. mile, velocity = 57.6 mph

to 1.676 Mbps to match the Utilicom Longranger 2050 radio modem. [15] The density was increased to 4 nodes per square mile to get similar connectivity as before (transmission range decreases at higher frequencies). The OPNET results showed in Figure 3.13 (right) indicate that both NSLS and HSLS outperform DLS since they have better scalability properties. Also, at this network size and for this density there is not much difference between HSLS and NSLS and even there are cases where NSLS outperforms HSLS. This is not strange since at this network size, the network diameter is small and NSLS's and HSLS's LSU generation processes are almost the same (most nodes receive the LSUs with TTL equal to 2), so that their relative diference is subject to experimental error. Besides, our theoretical results hold for saturation conditions (where the remaining capacity is the more important factor) while the simulations are based on a lightly loaded scenario. However, as the size increases, HSLS's lead over NSLS increases and one expect's to see HSLS outperforming NSLS in the simulations. Further increasing the network size up to of 800 nodes produced the results shown on Table 3.1. It can be noticed that NSLS's performance degrades significantly while the HSLS performance is still within acceptable levels.

These results not only indicate that HSLS is the best approach in the family of FSLS algorithms, but considering the demanding scenario (60 4Kbps streams under unreliable CSMA) they also show the feasibility of HSLS as an extremely easy-to-implement solution (see HSLS pseudo-code in Figure 3.11) for scalability to networks of hundreds of nodes.

[15] The Utilicom Longranger 2050 is a 2.4 Ghz ISM Band, spread spectrum radio with programmable data rates up to 1.676 Mbps.

3.7 Conclusions

In this chapter, a class of approaches that attempt to scale link-state routing by limiting the scope of the update dissemination in space and time have been considered. This class opens a new design space, which is based on neither global nor local information, and represents a new way of thinking where each node may have a different view of the network. The first fundamental analysis of this generic approach – which is referred to as "Fuzzy sighted link-state routing" – has been presented.

Using a novel perspective on the "overhead" that combines in a single metric the overhead due to control messages and the overhead due to route suboptimality, an analytical model has been formulated that leads to the derivation of the best algorithm in this class, namely the HSLS algorithm. Although extremely easy-to-implement, it turns out from the comparative asymptotic study of key algorithms in the next chapter that this algorithm has nearly the best possible asymptotic overhead for any routing algorithm – proactive or reactive. The methodology introduced in this chapter for the computation of the "total overhead" may be also applied for the comprehensive study of different protocols in the literature, as it has been done in the next chapter.

In addition, this work presents a new paradigm for the design of routing protocols for mobile ad hoc networks, where it is the overall system performance which take precedence over any other design criteria, and the theoretical analysis precedes the protocol design.

Finally, although this work has been focused on link state routing, it can easily be extended to geographical routing approaches. For example, it was stated that DREAM [109] has similarities with NSLS. The results of this chapter suggest that DREAM may be improved by employing the same information dissemination algorithm as HSLS (instead of a NSLS-like approach).

But, above all these timely contributions, in the long term the main contribution of this chapter's work is the insight gained for the development of the LLS algorithm. Thus, it is only after the recognition of this chapter's work as an enabler for a multi-mode routing protocol synthesis that the true dimension of this chapter value can be appreciated.

Chapter 4

Asymptotic Behavior of Ad Hoc Routing Protocols with Respect to Traffic, Mobility and Size.

In the previous chapters, a framework for a multi-mode routing protocol was proposed. That framework relied upon, among other modules, in the limited information dissemination module. It was realized that prior to implementing such a limited information dissemination algortihm, a better understanding of the trade-off between control overhead reduction and route degradation was required. Chapter 3 provided this understanding, and determined the best algorithm (namely HSLS) among a family of link state variants (namely, FSLS).

Morever, the previous chapter results need to be expanded and compared with the achievable scalability properties of routing protocols in general. That is, we need to understand how the performance of the best FSLS algorithm relates to the performance achievable by any other protocol in the literature. For these assestment to be made, first we need to understand the fundamental meaning and limits of ad hoc network's scalability.

The central theme of this chapter focuses on the development of principles and methodologies for the analysis and design of scalable routing strategies for ad hoc networks. Analytical models are developed and results are presented that provide significant insight into the aforementioned dependency and the general performance characteristics of the most important classes of ad hoc network routing algorithms. The theoretical models developed establish the basis for an unbiased analysis and comparison of the relative scalability of several proposed routing protocols. The performance comparisons utilize the methodology developed in Chapter 3, which uses the novel concept of *total overhead* in order to allow a fair comparison of a wide variety of routing protocols based upon a well-defined and general set of assumptions.

Thus, this chapter provides a better understanding on the scalability of ad hoc networks in general. Along with a new perspective on a routing protocol's scalability that takes into account the network's own scalability properties, the first precise (asymptotic) expressions reflecting the impact of network size, traffic intensity and mobility on protocol performance for Plain Flooding (PF), Standard Link State (SLS), Dynamic Source Routing (DSR) [107], Hierarchical Link State (HierLS) [105], Zone Routing Protocol (ZRP) [112], and Hazy Sighted Link State (HSLS), are presented here. These protocols are a representative set of the routing protocols available in the literature.

As such, the results provide researchers with improved understanding of the limits and trade-offs inherent in ad hoc network routing. A significant result is that, under the assumptions of this work, HSLS—while being easier to implement – scales better than HierLS and ZRP with respect to network size. This analytical result is validated with simulation analysis comparing HSLS and HierLS. Thus, another important contribution of this work is to show that HSLS is an effective, more efficient alternative than hierarchical approaches for routing in large ad hoc networks.

The remainder of this chapter is organized as follows: Section-4.1 discusse sthe (scarse) previous work in this topic. Section 4.2 presents our network model, assumptions, and methodology; icluding a new perspective in a routing protocol's scalability. The following sections (Sections 4.3 - 4.8) present analysis of the asymptotic performance of PF, SLS, DSR, HierLS, ZRP, and HSLS respectively. A brief performance comparison of the protocol is presented in Section-4.9, focusing on HierLS and HSLS under large network size and including simulation validation of results. Finally, conclusions are presented in Section-4.10

4.1 Related Previous Work

Routing protocols for ad hoc networking have been the subject of extensive research over the past several years. A question that is often asked in this context is: *which routing protocol scales the best?*. The answer that is usually given is: *it depends*. Unfortunately, the networking community lacks a tenet for understanding the fundamental properties and limitations of ad hoc networks. Hence, a fundamental understanding of what scalability *is*, and *what* and *how* it depends on, is currently lacking.

One reason for this shortcoming is a lack of sufficient research aimed at general principles and analytical modeling. Scalability and other performance aspects of ad hoc routing have been studied predominantly via simulations (e.g. [79], [78], [80]), versus theoretical analyses. Simulation results, although extremely useful, are often limited in scope to specific scenarios. Thus, it often fails to produce results that provide the depth of understanding of the limitations of the protocols and their dependence on system parameters and environmental factors desired by researchers. The lack of much needed theoretical analysis in this area is due in part to the lack of a common platform to base theoretical comparisons on, and in part due to the abstruse nature of the problem.

Despite limited prior related theoretical work, there have been notable exceptions. In [81] analytical and simulation results are integrated in a study that provides valuable insight into comparative protocol performance. However, it fails to deliver a final analytical result, deferring instead to simulation. Thus, it is difficult to fully understand the interactions among system parameters. The present work closes this gap and provides an understanding of the dynamic interaction among network parameters.

The asymptotic capacity of a fixed wireless network was studied in [83]; however, it did not include routing overhead. In contrast, the present work *focuses* on total overhead, which includes routing overhead. The impact of mobility on network capacity was studied in [84]. There it is shown that given no restriction on memory size and arbitrarily large delays, mobility increases network capacity. This research, however, focuses on practical scenarios, wherein, delay cannot grow arbitrarily large. As discussed later, mobility is shown to reduce network capacity, thus degrading network performance.

4.2 Preliminaries

This section presents the assumptions, definitions, and methodology employed in our analysis.

4.2.1 Network Model

The network model used is the one presented in Subsection 3.4.1. That network model is general enough to include most of the typical networking scenarios.

4.2.2 Definition of Scalabilty

This work is aimed at the study of the scalability properties of routing protocols for ad hoc networks. However, currently there is not a clear definition of scalability. Indeed, scalability has a different meaning for different people. Thus, we need to define the exact meaning of this term.

Definition 4.1 Scalability *is the ability of a network to support the increase of its limiting parameters.* [1]

Thus, scalability is a property. In order to quantify this property, we use the concept of *minimum traffic load* introduced in Subsection 3.2.1 (see definition 3.1) to define the *network scalability factor* as follows:

Definition 4.2 *Let* $Tr(\lambda_1, \lambda_2, \ldots)$ *be the* minimum traffic load *experienced by a network under parameters* $\lambda_1, \lambda_2, \ldots$ *(e.g. network size, mobility rate, data generation*

[1] The *limiting parameters* of a network are those parameter – as for example mobility rate, traffic rate, and network size, etc. – whose increase causes the network performance to degrade. On the remainder of this work only limiting parameters will be considered, and therefore the term 'parameter' will be used in lieu of the term 'limiting parameter'.

rate, etc.). Then, the network scalability factor *of such a network, with respect to a parameter* λ_i *(* Ψ_{λ_i} *) is defined to be :*

$$\Psi_{\lambda_i} \stackrel{\text{def}}{=} \lim_{\lambda_i \to \infty} \frac{\log Tr(\lambda_1, \lambda_2, \ldots)}{\log \lambda_i}$$

The *network scalability factor* is a number that asymptotically relates the increase in network load to the different network parameters. For the class of mobile ad hoc networks defined by our network model assumptions (see Subsection 3.4.1), the *minimum traffic load* $Tr(\lambda_{lc}, \lambda_t, N)$ is $\Theta(\lambda_t N^{1.5})$, [2] and therefore $\Psi_{\lambda_{lc}} = 0$, $\Psi_{\lambda_t} = 1$, and $\Psi_N = 1.5$.

The *network scalability factor* may be used to compare the scalability properties of different networks (wireline, mobile ad hoc, etc.), and as a result of such comparisons we can say that one class of networks scales better than the other. However, if our desire is to assess whether a network is *scalable* (an adjective) with respect to a parameter λ_i, then the *network rate* dependency on such a parameter must be considered.

Definition 4.3 *The* network rate $R^{network}$ *of a network is the maximum number of bits that can be simultaneously transmitted in a unit of time. For the* network rate *(*$R^{network}$*) computation all successful link layer transmissions must be counted, regardless of whether the link layer recipient is the final network-layer destination or not.*

Definition 4.4 *A network is said to be* scalable *with respect to the parameter* λ_i *if and only if, as the parameter* λ_i *increases, the network's* minimum traffic load *does not increase faster than the* network rate *(*$R^{network}$*) can support. That is, if and only if:*

$$\Psi_{\lambda_i} \leq \lim_{\lambda_i \to \infty} \frac{\log R^{network}(\lambda_1, \lambda_2, \ldots)}{\log \lambda_i}$$

[2] Each node generate λ_t bits per seconds, that must be retransmitted (in average) L times (hops). Thus, each node induce a load of $\lambda_t L$, which after adding all the nodes results in a $Tr(\lambda_{lc}, \lambda_t, N) = \lambda_t N L$. Since, by assumption a.4 L is $\Theta(\sqrt{N})$, the above expression is obtained.

For example, it has been proved that in mobile ad hoc networks, at most $\Theta(N)$ successful transmissions can be scheduled simultaneously (see for example [83, 84]). The class of networks under study in this work (i.e. resulting from applying power control techniques) are precisely the class of networks that achieves that maximum *network rate*. Thus, in order for mobile ad hoc network to be regarded as scalable with respect to network size, we will need $\Psi_N \leq 1$. Unfortunately this is not the case, and as a consequence ad hoc networks (under assumption a.1 through a.8 in Subsection 3.4.1) are not *scalable* with respect to network size. [3] Wireline networks, in the other hand, if fully connected may have $\Psi_N = 1$, and therefore they are potentially scalable (in the bandwidth sense defined here) with respect to netwotrk size. Note however, that this scalability requires the nodes' degree to grow without bound, which may be prohibitely expensive.

Similarly, since the *network rate* does not increase with mobility or traffic load, then a network will be scalable w.r.t. mobility and traffic if and only if $\Psi_{\lambda_{lc}} = 0$ and $\Psi_{\lambda_t} = 0$, respectively. Thus, the networks under this study are *scalable* w.r.t. mobility, but are not *scalable* w.r.t. traffic.

Note that similar conclusions may be drawn for scalability w.r.t. additional parameters as for example network density, transmission range ℓ, etc. that are not being considered in our analysis. For example, as transmission range increases (and assuming a infinite size network with regular density) the spatial reuse decreases and as a consequence *network rate* decreases as rapidly as ℓ^2. Thus, Ψ_ℓ should be lower than -2 for the network to be deemed *scalable*. Since the *minimum traffic load* will only decrease linearly w.r.t. ℓ (paths are shortening), $\Psi_\ell = -1$, and therefore ad hoc networks are not scalable w.r.t. transmission range. [4]

[3] It has been shown in [84] that if the network applications can support infinitely long delays and the mobility pattern is completely random, then the average path length may be reduced to 2 ($\Theta(1)$) regardless of network size and, as a consequence, that network *scalability factor* with respect to network size Ψ_N is equal to 1. Thus, those ad hoc networks (random mobility and capable of accepting infinitely long delays) are the only class of ad hoc networks that are scalable with respect to network size. This work does not consider that class of networks since they have no practical relevance.

[4] This observation is the main reason behind our focusing on networks with power control, where the transmission range is kept in line so that the network degree is kept bounded.

Now, after noticing that mobile ad hoc networks are not *scalable* with respect to size and traffic, one may ask the meaning of regarding a routing protocol *scalable.* The remaining of this subsection will clarify this meaning.

Definition 4.5 Routing protocol's scalability *is the ability of a routing protocol to support the continuous increase of the network parameters without degrading network performance.*

Thus, from the above definition it is clear that the *routing protocol scalability* is dependent on the scalability properties of the network the protocol is run over. That is, the network own scalabilty properties provides the reference level as to what to expect of a routing protocol. Obviously, if the overhead induced by a routing protocol grows faster than the *network rate* but slower than the *minimum traffic load*, the routing protocol is not degrading network performance, which is being determined by the *minimum traffic load.*

To quantify a *routing protocol scalabilty*, the respective scalability factor is defined, based on the *total overhead* concept presented in Subsection 3.2.1 (see definition 3.2), as follows:

Definition 4.6 *Let $X_{ov}(\lambda_1, \lambda_2, \ldots)$ be the* total overhead *induced by routing protocol X, dependent on parameters $\lambda_1, \lambda_2, \ldots$ (e.g. network size, mobility rate, data generation rate, etc.). Then, the Protocol X's* routing protocol scalability factor *with respect to a parameter λ_i ($\rho^X_{\lambda_i}$) is defined to be :*

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

The *routing protocol scalability factor* provides a basis for comparison among different routing protocols. Finally, to assess whether a routing protocol is *scalable* the following definition is used: And, finally

Definition 4.7 *A routing protocol X is said to be* scalable *with respect to the parameter λ_i if and only if, as the parameter λ_i increases, the* total overhead *induced by such protocol (X_{ov}) does not increase faster than the network's* minimum traffic load. *That is, if and only if:*

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

Thus, for the class of network under study, a routing protocol X is *scalable* with respect to network size if and only if $\rho_N^X \leq 1.5$; it is *scalable* w.r.t. mobility rate if and only if $\rho_{\lambda_{lc}}^X \leq 0$; and it is *scalable* w.r.t. traffic if and only if $\rho_{\lambda_t}^X \leq 1$.

In the remainder of this chapter we will derive asymptotic expressions for the total overhead (and therefore the *routing protocol scalability factor*) induced by a representative set of routing protocols. Besides the trivial result that Plain Flooding (PF) is the only protocol that is *scalable* with respect to mobilty, and that most protocols are *scalable* with respect to traffic, the more interesting result that HSLS is *scalable* with respect to network size is found.

4.2.3 Methodology

This chapter will employ the same methodology used by the previous chapter for computing the *total overhead* induced by several protocols. This methodology consisted in computing each of the three components of *total overhead*, namely *proactive, reactive* and *suboptimal routing*, separatedly and then adding them up.

4.3 Plain Flooding (PF)

In PF, each packet is (re)transmitted by every node in the network (except the destination). Thus, $N-1$ transmissions are required for each data packet, when the optimal value (on average) should have been L. Since there are $\lambda_t N$ data packets generated each second, the additional bandwidth required for transmission of all these packets is *data* $(N-1-L)\lambda_t N$ bps. Since $L = \Theta(\sqrt{N})$, the PF's suboptimal routing overhead per second is equal to $\Theta(\lambda_t(N^2 - N^{1.5})) = \Theta(\lambda_t N^2)$.

PF does not try to find routes toward the destination, so it does not induce neither *reactive* nor *proactive* cost. Thus, PF *total overhead* per second is $\Theta(\lambda_t * N^2)$. In consequence $\rho_{\lambda_t}^{PF} = 1$, $\rho_{\lambda_{lc}}^{PF} = 0$, and $\rho_N^{PF} = 2$.

4.4 Standard Link State (SLS)

In SLS, a node sends a Link State Update (LSU) to the entire network each time it detects a link status change. A node also sends periodic, soft-state LSUs every T_p seconds. There is no *reactive* overhead associated with SLS, and since the paths determined are optimal, there is no *suboptimal routing* overhead associated with it either.

In SLS, each node generates a LSU at a rate of λ_{lc} per second, so in average there are $N\lambda_{lc}$ LSUs being generated at any given second. Each LSU is retransmitted at least once per each node (i.e. N times), inducing an overhead of $lsu\,N$ bits (where lsu is the size of the LSU packet). Then SLS *proactive* and *total-overhead* per second is $lsu\,\lambda_{lc}N^2$ bps, that is, $\Theta(\lambda_{lc}N^2)$; and $\rho_{\lambda_t}^{SLS} = 0$, $\rho_{\lambda_{lc}}^{SLS} = 1$, and $\rho_N^{SLS} = 2$.

4.5 Dynamic Source Routing (DSR)

In DSR no proactive information is exchanged. A node (source) reaches a destination by flooding the network with a route request (RREQ) message. When a RREQ message reaches the destination (or a node with a cached route towards the destination) a route reply message is sent back to the source, including the newly found route. The source attaches the new route to the header of all subsequent packets to that destination, and any intermediate node along the route uses this attached information to determine the next hop in the route. The present work focuses on DSR without the route cache option (DSR-noRC). A lower bound for DRS-noRC's *total overhead* is derived next.

4.5.1 DSR without Route Cache (DSR-noRC)

The DSR-noRC *reactive* overhead must account for RREQ messages generated by new session requests (at a rate λ_s per second per node) and the RREQ messages generated by failures in links that are part of a path currently in use. If we only consider the RREQ messages generated by new session requests, then a lower bound can be obtained.

Each route request message is flooded to the entire network, resulting in $N-1$ retransmissions (only the destination does not need to retransmit this message). Thus, each message induces an overhead of $size\_of\_RREQ(N-1)$ bits, and there are $\lambda_s N$ RREQ messages generated every second due to new session requests. Thus, the DSR-noRC reactive overhead per second is $\Omega(\lambda_s N^2)$.

For the DSR-noRC *suboptimal routing* overhead a lower bound will be obtained by considering only the extra bandwidth required for appending the source-route in each data packet. The number of bits appended in each data packet will be proportional to the length L_i of path i. Since this length is not shorter than L_i^{opt} (the optimal path length), using L_i^{opt} instead of L_i will result on a lower bound. The extra bandwidth consumed by a packet delivered using a path i (with at least L_i^{opt} retransmissions) will be at least $(\log_2 N)(L_i^{opt})^2$, where $\log_2 N$ is the minimum length of a node address. The average extra bandwidth per packet over all paths is $E\{(\log_2 N)(L_i^{opt})^2)\} \geq (\log_2 N)E\{L_i^{opt}\}^2 = (\log_2 N)L^2$ bits. Thus, for each packet sent from a source to a destination there is an average *suboptimal routing* overhead of at least $(\log_2 N)L^2$ bits. Since $\lambda_t N$ packets are transmitted per second, the *suboptimal routing* overhead induced over the entire network is at least $\lambda_t N(\log_2 N)L^2$ bps. Recalling that $L = \Theta(\sqrt{N})$ (assumption a.4), the DSR-noRC *suboptimal routing* overhead per second is found to be $\Omega(\lambda_t N^2 \log_2 N)$ bps.

Combining the previous results, DSR-noRC *total overhead* per second is $\Omega(\lambda_s N^2 + \lambda_t N^2 \log_2 N)$. Also, $\rho_{\lambda_t}^{DSR-noRC} = 1$, $0 < \rho_{\lambda_{lc}}^{DSR-noRC} <= 1$, [5] and $\rho_N^{DSR-noRC} > 2$.

4.6 Hierarchical Link State (HierLS)

In the *m-level* HierLS routing, network nodes are regarded as level 1 nodes, and level 0 clusters. Level i nodes are grouped into level i clusters, which become level $i+1$ nodes, until the number of highest level nodes is below a threshold and therefore

[5] DSR's *total overhead* does depend on mobility, since breakages of links forming existing routes will trigger route discovery procedures that will induce reactive overhead and/or cause route degradation. Similar to the lower bound derived in this section, an upper bound for DSR's *total overhead* may be derived by assuming that each link breakage trigger a global route discovery (regardless of the link being part of an active route or not). Such an upper bound would increase linearly with the mobility rate, and therefore we obtain the upper bound for $\rho_{\lambda_{lc}}^{DSR-noRC} <= 1$.

they can be grouped (conceptually) into a single level m. Thus, the value of m is determined dynamically based on the network size, topology, and threshold values.

Link state information inside a level i cluster is aggregated (limiting the rate of LSU generation) and transmitted only to other level i nodes belonging in the same level i cluster (limiting the scope of the LSU). Thus, a node link change may not be sent outside the level 1 cluster (if they do not cause a significant change to higher levels aggregated information), greatly reducing the proactive overhead.

HierLS relies on the Location Management service to inform a source node S of the address of the highest level cluster that contains the desired destination D and does not contain the source node S. For example, consider a 4-level network as shown in Figure 4.1. S and D are level 1 nodes; $X.1.1$, $X.1.2$, etc. are level 2 nodes (level 1 clusters); $X.1$, $X.2$, etc. are level 3 nodes (level 2 clusters); X, Y, V, and Z are level 4 nodes (level 3 clusters); the entire network forms the level 4 cluster. The Location Management (LM) service provides S with the address of the highest level cluster that contains D and does not contain S (e.g. the level 3 cluster Z in Figure 4.1). Node S can then construct a route toward the destination. This route will be formed by a set of links in node S level 1 cluster ($X.1.1$), a set of level 2 links in node S level 2 clusters ($X.1$), and so on. In Figure 4.1 the route found by node S is : $S - n_1 - n_2 - X.1.5 - X.1.3 - X.2 - X.3 - Y - Z - D$. When a node outside node S level 1 cluster receives the packet, the node will likely produce the same high-level route towards D, and will 'expand' the high-level links that traverse its cluster using lower level (more detailed) information. In Figure 4.1 this expansion is shown for the segment $Z - D$. The Location Management (LM) service can be implemented in different ways, whether proactive (location update messages), reactive (paging), or hybrid. Typical choices are:

LM1 Pure reactive. Whenever a node changes its level i clustering membership but remains in the same level $i + 1$ cluster, this node sends an update to all the nodes inside its level $i + 1$ cluster. For example, (see Figure 4.1) if node n_2 moves inside cluster $X.1.5$, i.e. it changes its level 1 cluster membership but does not change its level 2 cluster membership (cluster $X.1$), then node n_2 will send a location update to all the nodes inside cluster $X.1$. The remaining nodes

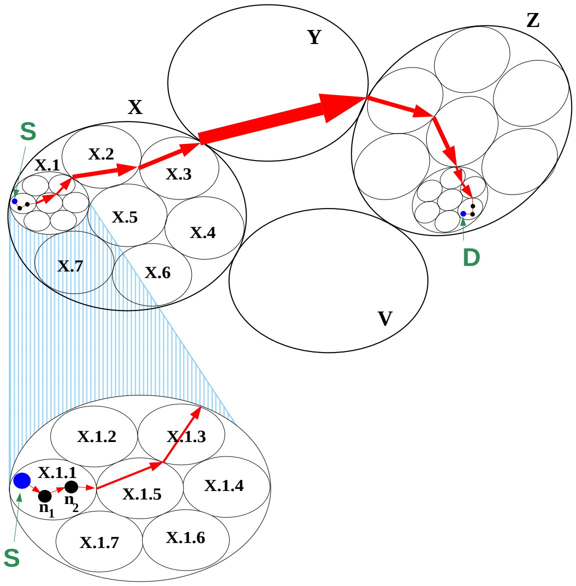


Figure 4.1: A Source (S) - Destination (D) path in HierLS.

will not be informed.

LM2 Local paging. In this LM technique, one node in each level 1 cluster assumes the role of a LM server. Also, one node among the level 1 LM servers inside the same level 2 cluster assumes the role of a level 2 LM server, and so on up to level m. The LM servers form a hierarchical tree. Location updates are only generated and transmitted between nodes in this tree (LM servers). When a node D changes its level i clustering membership, the LM server of its new level i cluster will send a location update message to the level $i+1$ LM server, which in turn will forward the update to all the level i LM servers inside this level $i+1$ cluster. Additionally, the level $i+1$ LM server checks if the node D is new in the level $i+1$ cluster, and if this is the case it will send a location update to its level $i+2$ LM server, and so on.

When a level i LM server receives a location update message regarding node D from its level $i+1$ LM server, it updates its local database with node D's new location information and forwards this information to all the level $i-1$ LM servers inside its level i cluster. Each of these level $i-1$ LM servers forwards the location update message to the level $i-2$ servers in its level $i-1$ cluster, and so on until all the level 1 LM servers (inside node D's level $i+1$ cluster) are informed of the new level i location information of node D. When a node needs location information about any node in the network, the node pages its level 1 LM server for this information.

LM3 Global paging. LM3 is similar to LM2. In LM3, however, when a level i LM server receives a location update from a higher level $i+1$ LM server, it does not forward this information to the lower level ($i-1$) LM servers. Thus, a lower level (say level $j < i$) LM server does not have location information for nodes outside its level j cluster. A mechanism for removing outdated location information about nodes that left a level i cluster need to be added to the level i clusters LM servers. Basically, a level 1 LM server that detects that a node left its level 1 cluster will remove the entry corresponding to this node from its own database, and will inform its level 2 LM server. The level 2 LM server will wait for a while for a location update from the new level 1 cluster (if inside the same level 2 cluster) and if no such an update is received it will remove the node entry and will inform its level 3 LM server, and so on until arriving to a LM server that already has information about the new location of the node.

When a node needs location information about any node in the network, the node pages its level 1 LM server for the information. If the level 1 LM does not have the required information, it (the level 1 LM server) pages its level 2 LM server, who in turn pages its level 3 LM server, and so on, until a LM server with location information about the desired destination is found.

In this work, a pure proactive LM technique (LM1) will be initially considered, since approach LM1 is easier to implement and analyze. The total overhead of such protocol, referred to as HierLS-LM1, will be analyzed in the next three subsections. Approach LM2 (referred to as HierLS-LM2) potentially reduces the bandwidth consumption (for reasonable values of λ_s) but at the expense of complexity (selection and maintenance of LM servers) and an increase in the latency for route establishment. However, the asymptotic characteristic of HierLS do not change whether we use approach LM1 or approach LM2, as will be shown in subsection 4.6.4. Approach

LM3 (referred to as HierLS-LM3) is the more complex to implement and analyze. It will induce a significant amount of reactive overhead (susceptible to traffic), but will reduce the amount of overhead induced by mobility. Approach LM3 will be analyzed in subsection 4.6.5.

4.6.1 HierLS-LM1 Proactive Overhead

A network organized in m level clusters, each of equal size k ($N = k^m$) is considered. Note that k is predefined while m increases with N.

Under assumption a.7, HierLS-LM1's *proactive* asymptotic overhead is dominated by the location management function, that induces an overhead that grows at least as fast as $sN^{1.5}$, where s is the node relative speed. In the other hand, most of the LSUs updates will correspond to level 1 links, and will be propagated inside the level 1 clusters only. Thus, LSU packets will induce a proactive overhead that will only grow as fast as $\lambda_{lc}\, k\, N$ (this is, of course, a lower bound).

HierLS-LM1 location management overhead expressions, can be obtained by considering that the time a node takes to change its level $m-1$ cluster is directly proportional to the diameter of this level $m-1$ cluster and inversely proportional to the node's relative speed s. Since the level $m-1$ cluster size is N/k, then the cluster diameter is $\Theta(\sqrt{N/k})$. Under approach LM1, the new location information will have to be forwarded to all the nodes inside the level m cluster (the entire network). Thus, every node will send a location update message to the entire network (N transmissions) each $\Theta(\sqrt{N/k}/s)$ seconds, inducing an overhead of $\Theta(\sqrt{k}\, s\sqrt{N})$ bits every second. Adding up all nodes contributions, the proactive overhead per second due to level $m-1$ clusters membership change is $\Theta(\sqrt{k}sN^{1.5})$. Regarding the location updates generated due to level $m-i$ membership change, it can be seen that a level $m-i$ cluster is k^{i-1} times smaller than a level $m-1$ cluster, and consequently a level $m-i$ cluster's diameter is $k^{\frac{i-1}{2}}$ times smaller than a level $m-1$ cluster's diameter. Thus, the generation rate of location updates due to level $m-i$ membership changes is $k^{\frac{i-1}{2}}$ times larger than the rate induced by level $m-1$ changes. Also, since the new location information will have to be transmitted to all the nodes inside the current level $m - i + 1$ cluster, then the number of transmissions required for each packet decreases by a factor of $k^{-(i-1)}$ with respect to the number of transmissions induced by level $m - 1$ changes, which results in a net reduction of $k^{-\frac{i-1}{2}}$. Then, the overhead due to all location updates is :

$$\begin{aligned} Loc\_Upd\_Cost &= \Theta(\sqrt{k} s N^{1.5})[1 + k^{-\frac{1}{2}} + k^{-1} + \cdots] \\ &= \Theta(\sqrt{k} s N^{1.5}) \frac{1}{1 - \sqrt{1/k}} \end{aligned}$$

Thus, the location management overhead is $\Theta(s\, N^{1.5})$ bps. Combining this value with the lower bound obtained for the LSU-induced overhead ($\Omega(\lambda_{lc} N)$), it is concluded that the HierLS-LM1 *proactive* overhead is $\Omega(s\, N^{1.5} + \lambda_{lc} N)$.

4.6.2 HierLS-LM1 Suboptimal Routing Overhead

To estimate the *suboptimal routing* overhead, it is assumed that each level i (beginning with level 2) increases the actual route length by a factor f_i (f_i depends on the value of k, the LSU triggering thresholds, and is typically close to 1, for example $f = 1.05$ means a 5% increase in the route length). Thus, if the optimal path length is l, then the actual path length will be $\Pi_{i=2}^{i=m} f_i\, l$. Let f be the geometric average of the set $\{f_i\}$, that is, $f = (\Pi_{i=2}^{m} f_i)^{\frac{1}{m-1}}$. Then, the *suboptimal routing* overhead induced by a packet transmission is $data\,[f^{m-1} - 1]\, l = data\,[k^{(\log_k f)(m-1)} - 1]\, l = data\,[\frac{N}{k}^{\delta} - 1]\, l$, where $\delta = \log_k f$. There are $\lambda_t N$ packets generated each second, thus the average *suboptimal routing* overhead per second is $data\,(\frac{N}{k}^{\delta} - 1)\, L\, \lambda_t N$. Since L is $\Theta(\sqrt{N})$, we finally get that the HierLS-LM1 [6] *suboptimal routing* overhead per second is $\Theta(\lambda_t N^{1.5+\delta})$.

4.6.3 HierLS-LM1 Total Overhead

Combining the lower bound obtained for the *proactive* cost and the tight bound obtained for the *suboptimal route* cost we finally obtain that the *total overhead* per second for HierLS-LM1 is $\Omega(s * N^{1.5} + \lambda_{lc} * N + \lambda_t N^{1.5+\delta})$. [7] Also, $\rho_{\lambda_t}^{HierLS-LM1} = 1$,

[6] This result is also valid for HierLS-LM2 and HierLS-LM3.

[7] This expression appears to be a tight bound and the total overhead induced by HierLS-LM1 seems to be (hipotesis) $\Theta(s * N^{1.5} + \lambda_{lc} * N + \lambda_t N^{1.5+\delta})$.

$\rho_{\lambda_{lc}}^{HierLS-LM1} = 1$, and $\rho_{N}^{HierLS-LM1} = 1.5 + \delta > 1.5$ (HierLS is *almost* scalable w.r.t. network size).

4.6.4 HierLS-LM2 Total Overhead

LM2 differs from LM1 in that:

- LM1 transmit location update to all the nodes inside a level i cluster. LM2 transmit this updates only to the level 1 LM servers.
- LM2 induces a reactive cost when paging the level 1 LM servers asking for a destination location information.

The first difference implies that LM2 reduction factor on location update cost with respect to LM1 is at most in the order of the ratio of the number of nodes (N) to the number of level 1 LM servers ($\Theta(N/k)$). This ratio is $\Theta(k)$; where k, the number of nodes in a level 1 cluster, is fixed (predetermined, bounded). Thus, HierLS-LM2 *proactive* cost asymptotic behavior is the same as HierLS-LM1's .

The second difference implies a reactive cost component in HierLS-LM2 *total overhead* expression. Indeed, each time a new session is established the source node has to page its level 1 location server. This paging message need to be retransmitted in average $\Theta(\sqrt{k})$ times (i.e. the radii of the level 1 cluster). The important observation is that this number is bounded as the maximum size of a level 1 cluster is predetermined independently of the size N (what changes with N is the number of levels m). Thus, each second $\lambda_s N$ paging messages will have to be retransmitted a constant number of times, therefore inducing a bandwidth consumption per second that is $\Theta(\lambda_s N)$. Thus, the *reactive* cost induced by HierLS-LM2 (upper bounded) is $O(\lambda_t N)$. This value is smaller than the *suboptimal route* cost ($\Theta(\lambda_t N^{1.5+\delta})$) and therefore has no impact in the total overhead expression. [8]

Thus, HierLS-LM2 *total overhead* is $\Omega(s\, N^{1.5} + \lambda_{lc} N + \lambda_t N^{1.5+\delta})$, and shows the same asymptotic properties and scalability factors as HierLS-LM1.

[8] Recall that λ_s and λ_t are directly proportional.

4.6.5 HierLS-LM3 Total Overhead

When the LM3 approach is used, a node change in its level i membership will be informed (by the new level i LM server) to the other level i LM servers in the same level $i+1$ cluster (k nodes on average). The number of transmission needed to reach each of these level i LM servers will be of the same order of magnitude that the diameter (D_{i+1}) of a level $i+1$ cluster. The generation rates of these level i location updates (for a given node) will roughly be $\Theta(s/D_i)$ where s is the node speed. Thus, the bandwidth consumption due to level i location updates induced by one node is $\Theta(k\,D_{i+1}) * \Theta(s/D_i) = \Theta(k\frac{D_{i+1}}{D_i}s) = \Theta(s)$. Where the last equality holds since in average $\frac{D_{i+1}}{D_i} = \sqrt{k}$. Thus, considering the location updates due to levels 1, 2, .., m; we get that the bandwidth consumption due to all the location updates induced by a node is $\Theta(s\,m) = \Theta(s\,log_k N)$. And, considering the bandwidth consumed by all the N nodes in the network we get that HierLS-LM3 location update cost is $\Theta(s\,N \log N)$.

For the paging (*reactive*) cost, we recall that the fraction of nodes in a source (say S) $m-1$ level cluster is (on average) $1/k$. Thus, most of the nodes belong to a different level $m-1$ cluster. Since all the nodes are equiprobable destinations (assumption a.6), we conclude that the majority of destinations will require long pages, that is, will require pages that will travel all the way to the level $m-1$ LM server. Thus, we may simplify the analysis by considering only the cost of paging for information to destination outside one node level $m-1$ cluster. Thus, each page will require at least $\Omega(\sqrt{N/k})$ transmissions (assuming that optimal routes to the level i LM server are available, and because of assumption a.4). Also, a page is generated at least every new session and the fraction of these pages that refer to destination outside the source (S) level $m-1$ cluster is $k-1/k \approx 1$. Thus, each second there are at least $\frac{k-1}{k} * \lambda_s * N$ pages being generated (in the entire network), inducing a *reactive* cost of at least $\Omega(\frac{k-1}{k} * \lambda_s * N\sqrt{N/k}) = \Omega(\lambda_s N^{1.5})$.

Note that an upper bound can also be found if we consider that all pages are far reaching, that a page is triggered for each data packet (at a rate of $\lambda_t * N$ packets per second) and that the average number of transmissions required for each page is $\Theta(N^{0.5+\delta})$ (see subsection 4.6.2 about suboptimal routes). Then, the paging cost obtained is $O(\lambda_t N^{1.5+\delta})$. Thus, the *reactive* cost (paging LM servers) can be absorbed by the *suboptimal route* cost expression ($\Theta(\lambda_t N^{1.5+\delta})$), and its inclusion in the *total overhead* expression will have no effect.

Finally, HierLS-LM3 *total overhead* per second is $\Omega(s\,N \log N + \lambda_{lc} N + \lambda_t N^{1.5+\delta})$, slightly different from the asymptotic expressions for HierLS-LM1 and HierLS-LM2 (although it has the same scalability factors).

4.7 Zone Routing Protocol (ZRP)

ZRP is a hybrid approach, combining a proactive and a reactive part, trying to minimize the sum of their respective overheads. In ZRP, a node disseminates event-driven LSUs to its k-hop neighbors (nodes at a distance, in hops, of k or less). Thus, each node has full knowledge of its k-hop neighborhood and may forward packets to any node within it. When a node needs to forward a packet outside its k-hop neighborhood, it sends a route request to a subset of the nodes in the network, namely the 'border nodes'. The 'border nodes' will have enough information about their k-hop neighborhoods to decide whether to reply to the route request or to forward it to its own set of 'border' nodes. The route formed will be described in terms of the 'border' nodes only, thus allowing 'border' nodes to locally recover from individual link failures, reducing the overhead induced by route maintenance procedures.

In Appendix C, a lower bound for for ZRP' *total overhead* (ZRP_{ov}) was found to be equal to $\Omega(n_k\,\lambda_{lc} N + \lambda_s N^2/\sqrt{n_k})$. Where n_k is the average number of nodes inside another node's 'zone' (i.e. 'k' hops neighborhood). The first term represents the proactive overhead, while the second term represents the reactive overhead. Minimizing this lower bound by properly choosing the value n_k, shows that the best asymptotic behavior of the bound is obtained when (if possible) $n_k = \Theta((\frac{\lambda_s N}{\lambda_{lc}})^{\frac{2}{3}})$, obtaining a total overhead that is $\Omega(\lambda_{lc}^{\frac{1}{3}} \lambda_s^{\frac{2}{3}}\, N^{\frac{5}{3}})$. [9]

When $\lambda_{lc} = \Theta(\lambda_s\, N)$, n_k must be $\Theta(1)$ and therefore the *total overhead* induced by ZRP becomes $\Omega(\lambda_{lc}\, N + \lambda_s N^2) = \Omega(N(\lambda_{lc} + \lambda_s N)) = \Omega(\lambda_s N^2)$. If λ_{lc} grows faster

[9] Note that in this case k, the zone radius, is $\Theta((\frac{\lambda_s N}{\lambda_{lc}})^{\frac{1}{3}})$. Thus, k should increases with traffic and decreases with mobility as suggested in [113]; but the functional form of the non linear dependency was not anticipated.

that $\Theta(\lambda_s N)$, values of n_k lower than 1 do not make sense. What happens is that ZRP behaves in a pure reactive mode (as DSR) and therefore the *total overhead* induced by ZRP in those cases is also $\Omega(\lambda_s N^2)$.

On the other hand, if $\lambda_{lc} = \Theta(\lambda_s/\sqrt{N})$, the best achievable value of n_k is $\Theta(N)$. Thus, ZRP's *total overhead* becomes $\Omega(\lambda_{lc} N^2 + \lambda_s N^{1.5}) = \Omega(N^2(\lambda_{lc} + \lambda_s/\sqrt{N})) = \Omega(\lambda_{lc} N^2)$. If $\lambda_s/\sqrt{N}$ grows faster than λ_{lc}, then n_k can not grow more than N and therefore ZRP behaves in a pure proactive mode (as SLS) and induces a total overhead of $\Omega(\lambda_{lc} * N^2)$.

Finally, ZRP's total overhead is:

$$ZRP_{ov} = \begin{cases} \Omega(\lambda_{lc} N^2) & \text{if } \lambda_{lc} = O(\lambda_s/\sqrt{N}) \\ \Omega(\lambda_{lc}^{\frac{1}{3}} \lambda_s^{\frac{2}{3}} N^{\frac{5}{3}}) & \text{if } \lambda_{lc} = \Omega(\lambda_s/\sqrt{N}) \\ & \text{and } \lambda_{lc} = O(\lambda_s N) \\ \Omega(\lambda_s N^2) & \text{if } \lambda_{lc} = \Omega(\lambda_s N) \end{cases}$$

Note that the asymptotic expression provides us with much more information about the parameters interactions than the scalability factors, which are computed assuming that just one parameter is increased while the others remain fixed. For ZRP, $\rho^{ZRP}_{\lambda_t} = 0$ (pure proactive mode), $0 < \rho^{ZRP}_{\lambda_{lc}} <= 1$ (pure reactive mode, similar to DSR's), and $\rho^{ZRP}_N \geq 1.66$. Note that the information provided by the *scalabilty factors* is incomplete, and it hinds the fact that the exponential rates of increase of ZRP's total overhead with respect to mobility and traffic always add up to at least 1, as can be seen from the *total overhead*'s asymptotic expressions.

4.8 Hazy Sighted Link State (HSLS)

HSLS was derived in the next chapter based on the observation that nodes that are far away do not need to have complete topological information in order to make a good next hop decision. Thus, propagating every link status change over the entire network may not be necessary. In a highly mobile environment, a node running HSLS will transmit - provided that there is a need to - a LSU only at particular time instants that are multiples of t_e seconds. Thus, potentially several link changes are 'collected' and transmitted every t_e seconds. The *Time To Live* (TTL) field of the LSU packet is set to a value (which specifies how far the LSU will be propagated) that is a function of the current time index as explained below. After one global LSU transmission – LSU that travels over the entire network, i.e. TTL field set to infinity, as for example during initialization – a node 'wakes up' every t_e seconds and sends a LSU with TTL set to 2 if there has been a link status change in the last t_e seconds. Also, the node wakes up every $2t_e$ seconds and transmits a LSU with TTL set to 4 if there has been a link status change in the last $2t_e$ seconds. In general, a node wakes up every $2^{i-1}t_e$ $(i = 1, 2, 3, ...)$ seconds and transmits a LSU with TTL set to 2^i if there has been a link status change in the last $2^{i-1}t_e$ seconds. If a packet TTL field value (2^i) is greater than the distance from this node to any other node in the network (which will cause the LSU to reach the entire network), the TTL field of the LSU is reset to infinity (global LSU), and the algorithm is re-initiated.

Nodes that are at most two hops away from a node, say X, will receive information about node X's link status change at most after t_e seconds. Nodes that are more than 2 but at most 4 hops away from X will receive information about any of X links change at most after $2t_e$ seconds. In general, nodes that are more than 2^{i-1} but at most 2^i hops away from X will receive information about any of X links change at most after $2^{i-1}t_e$ seconds. Figure 4.2 shows an example of HSLS's LSU generation process when mobility is high and in consequence LSUs are always generated. An arrow with a number over it indicates that at that time instant a LSU (with TTL field set to the indicated value) was generated and transmitted. Figure 4.2 assumes that the node executing HSLS computes its distance to the node farthest away to be between 17 and 32 hops, and therefore it replaces the TTL value of 32 with the value infinity, resetting the algorithm at time $16t_e$.

In the previous chapter, closed form expressions were derived for a tagged node assumed at the network center. In this chapter, the assumption of the node being located at the center of the network is relaxed, and asymptotic expression are derived for the *total overhead* induced by all the nodes in the network (centrally located or not).

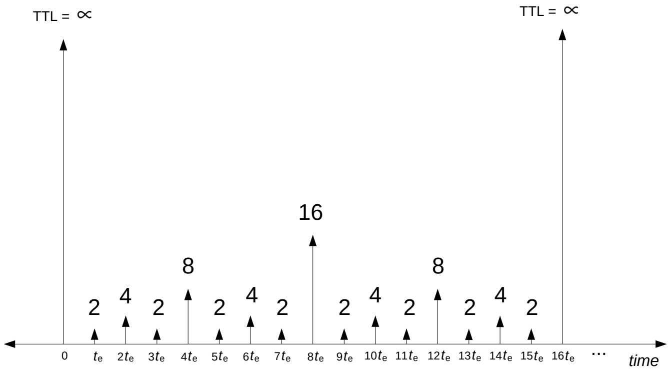


Figure 4.2: HSLS's LSU generation process (mobility is high).

4.8.1 HSLS Proactive Overhead

A highly mobile environment (i.e. a LSU is generated every time interval) is considered. All the different LSUs (re)transmissions due to LSUs generated by a node, say X, will be added and then averaged over time. The value obtained will be multiplied by the number of nodes in the network to get the *proactive* overhead. LSUs will be grouped based on their TTL value at the time they were generated, beginning with the LSUs with larger TTL values.

Let MD_x be the maximum distance from node X to any other node in the network. Let R_x be the power of 2 such that $R_x < MD_x \leq 2R_x$. For example, $R_x = 16$ in figure 4.2, where MD_x was assumed to be between 17 and 32. Under HSLS, node X computes MD_x each t_e seconds based on its own topology information, which is not necessarily up-to-date, so MD_x is a time-changing value that is not being timely updated. The above observation, however, will have little impact on the value of R_x, which may be assumed roughly constant over time.

Let's consider what happens at time $R_x t_e$ ($16t_e$ in figure 4.2). At this time node X sends a LSU to the entire network and the algorithm is re-initiated. Thus, every $R_x t_e$ seconds node X induces N transmissions, and therefore the bandwidth consumption due to these global LSUs is $\frac{lsu\,N}{R_x t_e}$, where lsu is the average length of a LSU packet.

The second larger TTL is R_x, and LSUs with this TTL are generated $\frac{R_x}{2} t_e$ seconds after a global LSU is sent (times $8t_e$ in figure 4.2). Recalling that the timers are reset at time $R_x t_e$, we notice that the interval between consecutive generation times is

$(R_x t_e - \frac{R_x}{2} t_e) + \frac{R_x}{2} t_e = R_x t_e$. Thus, the generation rate of LSUs with TTL equal to R_x is $\frac{1}{R_x t_e}$ (the same as the generation rate of global LSUs). These LSUs will not reach all the nodes in the network but only a fraction f_x. From assumption a.3, f_x should be around $(R_x/MD_x)^2$, i.e., $f_x \in [0.25, 1]$. In practical situations, due to boundary effects (i.e. the number of nodes at a maximum distance MD_x is small), we obtain that typically f_x is in the interval $[0.5, 1]$. Thus, the bandwidth consumption due to LSUs with TTL equal to R_x is $\frac{lsu\, f_x N}{R_x t_e}$.

For the remaining TTL values, 'boundary' conditions are no longer relevant. Thus, for TTL equal to $R_x/2$ the generation rate doubles (e.g. LSUs with TTL equal to 8 are sent at times $4t_e, 12t_e, \cdots$ in figure 4.2), and the number of transmissions induced per LSU is reduced by a factor of 4 (because of assumption a.3, and the fact that the TTL values are reduced to a half); thus the total effect is a reduction by a factor of 2 with respect the bandwidth consumption due to LSUs with TTL equal to R_x. The same argument applies for TTL equal to $R_x/4, R_x/8, ..., 2, 1$. [10] Finally, the total bandwidth consumption due to all the LSUs generated by node X is equal to :

$$X^{pro}_{HSLS} = \frac{lsu\, N}{R_x t_e} + \frac{lsu\, f_x N}{R_x t_e} + \frac{lsu\, f_x N}{2 R_x t_e} + \frac{lsu\, f_x N}{4 R_x t_e} + \dots$$
$$= \tfrac{lsu\, N}{R_x t_e}[1 + f_x(1 + \tfrac{1}{2} + \tfrac{1}{4} + \dots)] \;\approx \frac{lsu\, N}{R_x t_e}[1 + 2 f_x]$$

Since the size of a LSU depends only on the node density (bounded on average), f_x is bounded below 1, and R_x is $\Theta(\sqrt{N})$ (assumption a.4); the *proactive* overhead per second induced by one node is $\Theta(\frac{N^{0.5}}{t_e})$. Since there are N nodes, the *proactive* overhead per second induced by the entire network is $\Theta(\frac{N^{1.5}}{t_e})$.

4.8.2 HSLS Suboptimal Routing Overhead

Let t^{elap}_k be the maximum time elapsed since 'fresh' LSU information about a destination k hops away was last received. HSLS induces a quasi-linear relationship between t^{elap}_k and k. In general, $\frac{t_e}{2} \le \frac{t^{elap}_k}{k} \le t_e$. Thus, the ratio between the time elapsed

[10] Assumptions a.3 and a.4 are asymptotic conditions, and as such, are not applicable to small values of TTL. However, the contributions of LSUs with small TTL values in the proactive overhead of a large network is not significant and a more exact analysis can be safely omitted.

since fresh information was received and distance is *bounded* by t_e, independently of network size or distance to the destination. Based on the mobility model assumption a.8 (time scaling), this will cause the probability of a suboptimal next hop decision to be bounded[11], and the fraction of the increase of the suboptimal routes (with respect to the optimal ones) to also be bounded independently of network size. Then, it has been found in Appendix D that for a fixed value of t_e, HSLS *suboptimal routing* overhead will increase as $\Theta(\lambda_t N^{1.5})$ with respect to traffic and network size.

To investigate the dependence of the *suboptimal routing* overhead on the time t_e, a more precise mobility model need to be defined. Assuming a mobility model that induces an exponential residence time on a given area, HSLS *suboptimal routing* overhead was found (also in Appendix D) to be equal to : $\Theta((e^{\lambda_{lc} t_e K_4} - 1)\lambda_t N^{1.5})$, where k_4 is a constant.

4.8.3 HSLS Total Overhead

There is no *reactive* overhead associated with HSLS. Thus, the HSLS *total overhead* for the class of networks analyzed in the previous subsections is equal to :

$$HSLS_{ov} \;=\; N^{1.5}[K_5 \frac{1}{t_e} + K_3 (e^{\lambda_{lc} t_e K_4} - 1)\lambda_t]$$

The value of t_e should be tuned to optimize performance. For a moment, let's use the approximation $e^x - 1 \approx x$, where $x = \lambda_{lc} t_e K_4$. Thus:

$$HSLS_{ov} \;\approx\; N^{1.5}[\frac{K_5}{t_e} + K_6 \lambda_{lc} \lambda_t t_e]$$

Choosing the value of t_e that minimizes the above expression we get $t_e = \Theta(\frac{1}{\sqrt{\lambda_{lc}\lambda_t}})$, $x = \Theta(\frac{\sqrt{\lambda_{lc}}}{\sqrt{\lambda_t}})$, and $HSLS_{ov} = \Theta(\sqrt{\lambda_{lc}\lambda_t} N^{1.5})$. The previous expression would define the asymptotic behavior of HSLS's *total overhead* only if our approximation $e^x - 1 \approx x$ is valid. Indeed, if λ_t grows asymptotically faster than λ_{lc}, the value of x goes to zero and the approximation $e^x - 1 \approx x$ is valid. On the other hand, if λ_{lc} grows asymptotically faster than λ_t, the approximation will not be valid. In

[11] Since the ratio maximum displacement – speed times elapsed time – over distance is bounded, so is the 'angular' displacement of the destination. The 'angular' displacement will determine whether the node chosen as the next hop is the proper one or not.

this case, since the exponential function is the fastest growing, it is desirable to maintain the product $\lambda_{lc} t_e$ (and therefore the value of p) bounded and therefore we choose $t_e = \Theta(\frac{1}{\lambda_{lc}})$. Thus, the HSLS *total overhead* in this scenario becomes $\Theta(N^{1.5}(\lambda_{lc} + \lambda_t)) = \Theta(\lambda_{lc} N^{1.5})$, where the last equality holds due to our assumption that λ_{lc} grows asymptotically faster than λ_t and therefore λ_{lc} dominates the previous expression. Thus, the HSLS's *total overhead* is :

$$HSLS_{ov} = \begin{cases} \Theta(\sqrt{\lambda_{lc}\lambda_t} N^{1.5}) & \text{if } \lambda_{lc} = O(\lambda_t) \\ \Theta(\lambda_{lc} N^{1.5}) & \text{if } \lambda_{lc} = \Omega(\lambda_t) \end{cases}$$

Also, it can be noted that $\rho_{\lambda_t}^{HSLS} = 0.5$, $\rho_{\lambda_{lc}}^{HSLS} = 1$, and $\rho_N^{HSLS} = 1.5$. Thus, HSLS is the only protocol that is *scalable* with respect to network size.

4.9 Comparative Study

In the previous sections the *scalability factors* of several representative routing protocols have been derived. From those results we concluded that PF is the only protocol known to be *scalable* w.r.t. mobility ($\rho_{\lambda_{lc}}^{PF} = 0$), while all of the protocols were *scalable* w.r.t. traffic. More interesting was to find that HSLS is the only protocol *scalable* with respect to network size ($\rho_N^{HSLS} = 1.5$). However, much more information about the protocol parameter's interactions may be derived from the asymptotic *total overhead* expressions.

The *total overhead* results derived in th eprevious sections are summarized in Tables 4.1 and 4.2. Table 4.1 present the results for each overhead source (*proactive*, *reactive*, and *suboptimal routing*). [12] Table 4.2 presents the results for the *total overhead* when the tunable parameters are selected to optimize performance (or at least, optimize the lower bounds derived before).

These results increase our understanding of the limits and provide valuable insight about the behavior of several representative routing protocols. The better understanding of these limits will help network designers to better identify the class of protocols to engage depending on their operating scenario. For example, if the

[12] Unless otherwise stated, the HierLS results correspond to HierLS-LM1.

Protocol	Proactive	Reactive	Suboptimal
PF	–	–	$\Theta(\lambda_t N^2)$
SLS	$\Theta(\lambda_{lc} N^2)$	–	–
DSR-noRC	–	$\Omega(\lambda_s N^2)$ $O((\lambda_s + \lambda_{lc}) N^2)$	$\Omega(\lambda_t N^2 \log_2 N)$
HierLS	$\Omega(sN^{1.5} + \lambda_{lc} N)$	–	$\Theta(\lambda_t N^{1.5+\delta})$
ZRP	$\Theta(n_k \lambda_{lc} N)$	$\Omega(\lambda_s N^2/\sqrt{n_k})$	$O(\lambda_t N^2/\sqrt{n_k})$
HSLS	$\Theta(N^{1.5}/t_e)$	–	$\Theta((e^{\lambda_{lc} t_e K_4} - 1)\lambda_t N^{1.5})$

Table 4.1: Asymptotic results for several routing protocol for mobile ad hoc networks.

designer's main concern is network size, it can be noted that HierLS and HSLS scale better than the others. Similarly, if traffic intensity is the most demanding requirement, then SLS, and ZRP are to be preferred since they scale better with respect to traffic (*total overhead* is independent of λ_t); HSLS follows as it scales as $\Theta(\sqrt{\lambda_t})$, and PF, DSR, and HierLS are the last since their *total overhead* increases linearly with traffic. [13]

Similarly with respect to the rate of topological change, we observe that PF may be preferred (if size and traffic are small and the rate of topological change increases too rapidly), since its *total overhead* is independent of the rate of topological change. Provably next will be ZRP and DSR since their lower bounds are independent of the rate of topological changes. The bounds are not necessarily tight, and ZRP's and DSR's behavior should depend somewhat of the rate of topological change. Finally, for SLS, HierLS, and HSLS we know (as opposed to DSR and ZRP where we suppose) that their *total overhead* increase linearly with the rate of topological change.

It is interesting to note that when only the traffic or the mobility is increased (but not both), ZRP can achieve almost the best performance in each case.[14] However,

[13] It is interesting to note that HSLS scales better with traffic intensities than HierLS (the only other protocol that scales well with size). This result may have an intuitive explanation in the fact that HierLS never attempts to find optimal routes towards the destination, even under slowly changing conditions. HSLS on the other hand, may eventually obtain full topology information – and therefore optimal routes – if the rate of topological changes is small with respect to $1/t_e$, as is the case when λ_t grows faster than λ_{lc}.

[14] Almost, because ZRP can not achieve the independence of *total overhead* from mobility. PF

Protocol	Total overhead (best)	Cases
PF	$\Theta(\lambda_t N^2)$	Always
SLS	$\Theta(\lambda_{lc} N^2)$	Always
DSR-noRC	$\Omega(\lambda_s N^2 + \lambda_t N^2 \log_2 N)$	Always
HierLS	$\Omega(sN^{1.5} + \lambda_{lc}N + \lambda_t N^{1.5+\delta})$	LM1 or LM2 approach used
	$\Omega(s * N \log N + \lambda_{lc} * N + \lambda_t N^{1.5+\delta})$	LM3 approach is used
ZRP	$\Omega(\lambda_{lc} N^2)$	if $\lambda_{lc} = O(\lambda_s/\sqrt{N})$
	$\Omega(\lambda_{lc}^{\frac{1}{3}} \lambda_s^{\frac{2}{3}} N^{\frac{5}{3}})$	if $\lambda_{lc} = \Omega(\lambda_s/\sqrt{N})$ and $\lambda_{lc} = O(\lambda_s N)$
	$\Omega(\lambda_s N^2)$	if $\lambda_{lc} = \Omega(\lambda_s N)$
HSLS	$\Theta(\sqrt{\lambda_{lc}\lambda_t} N^{1.5})$	if $\lambda_{lc} = O(\lambda_t)$
	$\Theta(\lambda_{lc} N^{1.5})$	if $\lambda_{lc} = \Omega(\lambda_t)$

Table 4.2: Best possible total overhead bounds for mobile ad hoc networks protocols.

if mobility and traffic increase at the same rate; that is, $\lambda_{lc} = \Theta(\lambda)$ and $\lambda_t = \Theta(\lambda)$ (for some parameter λ), then ZRP's *total overhead* ($\Omega(\lambda N^{1.66})$) will present the same scalability properties as HSLS's ($\Theta(\lambda N^{1.5})$) and HierLS's ($\Theta(\lambda N^{1.5+\delta})$) with respect to λ, with the difference that ZRP does not scale as well as the other two with respect to size.

These and more complex analyses can be derived from the expression presented, when different parameters are modified simultaneously accordingly with the scenario the designer is interested in.

HSLS has better asymptotic properties than HierLS, which means that as size increases HSLS eventually outperform HierLS. The idea of HSLS – being much more simple to implement – outperforming HierLS is counter-intuitive. A first reaction to this result will likely be to assume that the constants involved in the asymptotic analysis may be too large, preventing HSLS from outperform HierLS under 'reasonable' scenario. Thus, the authors relied on a couple of simulation experiment to validate if, in effect, HSLS may outperform HierLS even under moderate network size and traffic load.

does.

4.9.1 A Simulation Experiment: HSLS vs. HierLS-LM1

Table 4.3 shows the simulation results obtained by OPNET for a 400-node network where nodes are randomly located on a square of area equal to 320 square miles (i.e. density is 1.25 nodes per square mile). Each node choose a random direction among 4 possible values, and move on that direction at 28.8 mph. Upon reaching the area boundaries, a node bounces back. The radio link capacity was 1.676 Mbps. Simulation were run for 350 seconds, leaving the first 50 seconds for protocol initialization, and transmitting packets (60 8kbps streams) for the remaining 300 seconds. The HierLS approach simulated was of the type HierLS-LM1, and corresponds to the DAWN project [115] modification of the MMWN clustering protocol [105]. The minimum and maximum cluster size was set to 9 and 35 respectively.

The metric of interest is the throughput (i.e. fraction of packets successfully delivered). The simulation results presented are not a comprehensive study of the relative performance of HierLS versus HSLS under all possible scenarios. They just presents and example of a real-life situation where HSLS outperform HierLS, and complement our theoretical analysis. The theoretical analysis focuses on asymptotically large network, heavy traffic load, and saturation conditions where the remaining capacity determines the protocol performance. The simulation results, in the other hand, refer to medium size networks with moderate loads, where depending on the MAC employed, other factors may have more weight over the protocols performance.

The results in Table 4.3 show that HSLS may outperform HierLS in medium size, more realistic, scenarios. However, both protocols performance is quite poor. The above is a consequence of the MAC protocol being employed, which was unreliable CSMA. For the network load being induced, the non-negligible probability of collision reduced the chance of packets reaching destination more than a few hops away. Thus, these results are for comparison sake only, since they suggest to use more elaborated MAC algorithms as the use of the RTS/CTS handshake. Two main reasons contributed to HSLS outperforming HierLS for such a wide margin:

- Since min-hop routing was used, the routing protocols tends to choose paths with 'longer' links (i.e. greater distance between the 2 nodes at each extreme of the link). As nodes move, these links deteriorate faster than the 'shorter' ones, and as a consequence packets are being loss (unreliable MAC). HierLS has to wait until a 'degraded' link is declared DOWN before switching the packet transmission to different ones. HSLS, in the other hand, is benefited from quick feedback about a node one-hop neighborhood by eavesdropping the HELLO messages exchanged by the neighbor discovery modules. Note that HierLS design philosophy prevents it from using such information, since it relies on all nodes inside the same cluster having the same view of the intra-cluster topology. HSLS, in the other hand, was designing under the assumption that nodes that are closer should be updated more frequently, so that having HELLOs messages interpreted as LSUs with TTL equal to 1 falls naturally into HSLS framework.

However, since some of HierLS shortcoming can be alleviated by techniques such as alternate path routing or by including 'stability' as a factor in the route selection, the authors tried to remove some of the bias towards HSLS. For this reason, HSLS-2 was also simulated. In HSLS-2, the routing protocol is prevented of eavesdropping the HELLO messages, and no level 1 LSU is transmitted. This was done for comparison sake only, and it is not the intention of the authors to propose such an approach. LSUs with TTL equal to 1 , being inexpensive, improve significantly the protocol performance – as can be seen from the difference between HSLS and HSLS-2 in Table 4.3 – so they should always be transmitted. Instead, the author would propose to improve HierLS to address the previous issues and improve performance. Unfortunately, due to time-constraint, the aforementioned approach had to be implemented (i.e downgrading HSLS instead of upgrading HierLS). Even with the above modifications, HSLS-2 outperformed HierLS.

- HierLS provided longer routes that made extremely difficult to the packets to reach their destination without colliding. HSLS also suffered from collisions, but the paths that HSLS provided for destination close by, tend to be smaller than the ones provided by HierLS. For example, when HSLS provided a 4 hop path, HierLS would provide a 6 hop path (for a destination in a neighboring

Protocol	Throughput	Delay
HSLS	0.2454	0.0163
HSLS-2	0.1556	0.0141
HierLS-LM1	0.0668	0.0134

Table 4.3: Throughput of a 400-node network.

cluster). The extra path length (2 hops) may seem negligible, but in a scenario where after 6 hops was almost certain that a packet would collide, it make a great difference. Since we were moderately loading the network, the probability of collision was high, and packet are not traveling more than 6 hops in average.

It can be seen that the previous results are highly influenced for another factors such as the MAC protocol being used, the quality of the links that neighbor discovery declares up, the latency on detecting link failures, etc. So, whether HSLS or HierLS should be preferred for a particular scenario, depends on the particular constraints (for example, if memory or processing time is an issue, HierLS may be preferred since it require to store/process an smaller topology table). The present work, however, provides some guidelines, suggesting that as traffic, network size, and data rate increases, and a better MAC is employed (allowing to achieve the full channel capacity), HSLS should tend to be preferred.

4.10 Conclusions

Prior to this work, the community lacked a basic tenet for understanding the fundamental limitations and invariants associated with ad hoc networks. Simulation studies, while valuable, provided results that were limited in scope, and traditional discrete-event simulation methods have failed to produce practical simulators for very large or highly mobile networks.

This chapter addressed these shortcomings by presenting a novel methodolgy – along with a new perspective on scalability – that allows for an analytical comparison, and deeper understanding of the characteristics and tradeoffs associated with various classes of routing protocols for mobile networks. This methodology, first introduced in chapter 3 to analyze a family of link state protocol variants, was fully developed in this chapter and applied to study a variety of protocols that have been proposed and analyzed via simulation methodologies in the literature.

The analytical methods developed in this work and the resulting asymptotic *total overhead*'s expressions provide an important contribution to the field that promises to shed new light on the fundamental limitations and underlying characteristics of mobile networks in general, and in the studied protocols in particular. Results for HSLS – a novel, easy-to-implement link state variant – suggest that the implementation of a complex hierarchy is not mandatory for scalability. A more focused comparison between HierLS and HSLS was undertaken, and as a result, HSLS was established as a competitive alternative to HierLS.

Finally, this work is only a first step. Greater understanding of cross-layer interactions and the impact of more general mobility models and traffic workloads is required. In particular, it was observed that for random mobilty, HSLS and HierLS (approximately) scaled as $\Theta(N^{1.5})$ with respect to network size. That value is a consequence that both proactive and suboptimal routing overheads increases as rapidly as $\Theta(N^{1.5})$, and therefore is unlikely to be improved. However, if mobility does present well defined patterns (group mobility) and the cluster selection is determined by those patterns, then the overhead induced by location management techniques (the main contributor to HierLS proactive overhead) may be significantly reduced and the proactive overhead of such a HierLS variant will reduce to $\Theta(N)$. This, in turn, open the posibility of improving the protocol performance by trading off some increase in the proactive overhead ($\Theta(N)$) in exchange for a decrease in the suboptimal routing overhead ($\Theta(N^{1.5+\delta})$, obtaining an overall decrease in the total overhead asymptotic behavior. Thus, it may be worthy to consider group mobility (or other pattern) if present when designing an effective routing protocol, as suggested in the multi-mode routing protocol framework presented in Chapter 2.

The multi-mode framework proposed not only uses limited information disseminations techniques, that as we have seen in this chapter enables a routing protocol to be *scalable* with respect to network size, but also includes a self-organizing module that tries to learn the current network mobility and traffic patterns in an attempt to adapt to them (and improve the routig protocol scalability w.r.t. mobility, specially in th epresence of well defined mobility patterns). The next chapter introduces a novel algorithm (SOAP) that represents an instantiation of the self-organizing module required by the multi-mode routing framework. Further research should focus on approaches that (like SOAP) tries to learn/exploit different mobility pattern for effective routing, and that enable a multi-mode routing protocol.

Chapter 5

SOAP : a Self-Organizing, Adaptive Protocol

This chapter completes our study of the elements of the framework for a multi-mode routing protocol presented in Chapter 2. The previous 2 chapters have discussed variants of the LLS algorithm (instantiation of the algorithm executed at the limited information dissemination module) and its impact on the scalability of ad hoc routing protocols. This chapter switches the attention to the algorithm proposed to be executed at the other learning/engaging module, the self-organizing module, namely the self-organizing (SO) algorithm.

In this chapter, the SO algorithm introduced in Chapter 2, which attempts to learn and exploit the instantaneous network structure, mobility and traffic patterns, is studied by providing a detailed protocol description, which provide us with a better understanding of its complexity and feasibility. Some extensions of SOAP to address heterogeneous scenarios (e.g. fixed-mobile integration) are also discussed.

SOAP, while including a limited dissemination technique (based on NSLS or similarly, the proactive element of ZRP), may also be interpreted as representing a particular instance of a multi-mode routing protocol. That is, under certain mobility and traffic patterns, a multi-mode routing protocol may elect to behave exactly as SOAP. However, a multi-mode routing protocol may change its behavior (mode) based on changes on the mobility and traffic patterns (i.e. the state) of the network.

This chapter attempts to provide an understanding of the complexities associated with the implementation of a self-organizing algorithm (first) and a multi-mode routing protocol (later). This chapter presents a different, more implementation oriented discussion compared to that in the previous ones.

The work in this chapter intends to be an initial step only. This chapter's focus is to provide a flexible protocol specification. Such a specification will provide an insight into SOAP's implementation complexity. Future work should be directed toward analyzing the impact of the different choices of the parameters of the algorithm (as for example the cost function) on the overall system performance. Such an assessment is expected to facilitate the development of a multi-mode routing protocol specification.

5.1 Preliminaries

In a mobile network there are two main issues that need to be considered. One is the cost of choosing a suboptimal path at a particular point in time (bandwidth waste), the other is the cost of tracking destinations due to mobility (i.e. link state or distance vector updates, cluster formation/maintenance and location management, etc.). Traditional protocols that address the first issue in a static network with moderate success (e.g. SLS, SDV) have a poor performance in a highly dynamic environment. This is due not only to the instability typically associated with them, but also to the increased bandwidth overhead required to operate correctly under increased mobility rates. For highly dynamic networks the bandwidth overhead present in typical Link State or Distance Vector algorithms may consume the greater part of the available bandwidth. It is clear that in such cases the availability of optimal routes is not of primary importance if the protocol itself congests the network. An alternative protocol that reduces the network congestion due to flooding of routing information is preferred, even if the routes obtained are not optimal. The HSLS algorithm, presented in the previous chapters is an example of such an algorithm. HSLS, however, does not take advantage of mobility patterns. It was pointed out in Chapter 4 that potentially better performance may be achieved if group mobility is taken into account during cluster membership selection in hierarchical approaches. Such a consideration may significantly reduce the location management overhead. Similarly, improvement may be obtained if traffic patterns are considered when proactively building routes toward destinations so that network resources are not wasted maintaining routes that are unlikely to be used.

An efficient routing protocol must take into account the spatial-temporal correlations inherent in any network and should also consider the network behavior and structure, trying to take advantage of them. In this chapter a Self-Organizing Adaptive Protocol (SOAP) that addresses these requirements, reducing the cost associated with route discovery is presented.

From an implementation point of view, SOAP combines features (and concepts) from Mobile IP [117], TORA [63], and ZRP [56]. Thus, a brief description of each is provided below.

Mobile IP [117] has been employed to handle mobility in cellular and other wireless networks. Mobile IP assumes a hub-based architecture with nodes organized around fixed access points that execute all the routing functions. Traditional Mobile IP relies on a Home Agent (HA) that knows the location of the Mobile Host (MH) at any point in time. Packets with a MH as their destination are sent first to its HA which, in turn, forwards them to the MH. Mobile IP may result in great waste of bandwidth since even if the destination MH is close to the source, all the packets have to go through the HA. On the other hand, mobile IP does not require the source to modify its IP kernel nor to be mobile-aware. More recently, a route optimization mechanism has been proposed [118] to improve the efficiency of mobile IP. Such route optimization is achieved by exploiting the binding between a MH and its Care-of-Address. The Care-of-Address of a MH is a router in the foreign network (where the MH is currently located) that agrees to provide service to the mobile host. The association between a MH and its Care-of-Address is called mobility binding or simply binding. In the earlier Mobile IP implementation, the HA intercepts all the packets addressed to a mobile host and then encapsulates and tunnels them to the Care-of-Address. An encapsulated packet will have the address of the Care-of-Address as the IP destination address and the address of the actual destination inside the IP payload. The route optimization extension in [118] allows the current Care-of-Address information to be passed along to the source node using special binding update and binding warning messages. Once the source node has the information of the destination's current Care-of-Address it will encapsulate and tunnel the packet itself, allowing the packet to go through the most cost-effective path toward the destination. The price paid for this performance improvement is that the source (usually a fixed host) should become mobile-aware.

TORA belongs to the family of 'link reversal' [19] algorithms and provides a node with information about the next hop toward a desired destination. That is, once a route discovery is invoked, the entire network becomes a Directed Acyclic Graph (DAG) rooted at the destination. TORA is a tracking protocol where each node has an unique height associated with each of the possible destinations. In TORA, the packets' flow direction is always from a higher to a lower node (downstream links). A node reacts to topological changes (lost of its last 'downstream' link) by increasing its own height, leading to incoming link reversal. In TORA the effect of topology changes are kept local, there are no loops, and several alternate routes toward a destination are provided. Simulation results in [78] show that for medium to high mobility networks TORA outperforms SLS (Standard Link State). On the other hand, TORA provides information about only the neighboring links not allowing for the computation of a path (succession of links toward a destination) metric or the identification of a best-cost route. TORA overhead and memory requirements increase linearly with the number of nodes in the network.

ZRP has been discussed in the previous chapters. A brief description follows for clarity and completeness sake. In ZRP, each node is the center of its own *zone*, which is composed of all its *k-neighbors* (nodes that are at a distance of k hops or less from it). The nodes that are exactly at a distance of k hops from the center of a *zone* are called *border nodes* of that *zone*. All nodes propagate changes in their link states (or their distance vector) within their *zone* according to the IntrAzone Routing Protocol (IARP), keeping changes local. Note that if ZRP's IARP chooses to propagate link state information (instead that distance vectors) it will be similar to the NSLS algorithm introduced in Chapter 3. When a source needs to send a packet to a destination, it first checks whether the destination is within its *zone* or not. If the destination is within the *zone* of the source node, the latter will forward the packet to the destination using the information provided by the IARP protocol. If the destination is not within the source's *zone* then the IntErzone Routing Protocol (IERP) is invoked. The IERP relies on a procedure called *bordercast* in which unicast packets are sent to the border nodes only. A source node will *bordercast* a REQUEST to its border nodes. The border nodes will know (thanks to their own IARP protocol) whether the destination is within their *zones*. If a border node has the destination within its *zones*, it will forward the packet to the destination. If the destination is not within a border node's *zone* the border node will propagate the REQUEST to its own border nodes and so on. Some mechanisms are included to prevent looping back or revisiting regions of the network that have already been queried. When a REQUEST reaches a destination, the succession of border nodes traversed is recorded and returned to the source node which stores it in its (IERP) routing table. If changes occur, which left the current routes unavailable, the source has to run a new route discovery procedure. In large mobile networks, it is highly likely that changes along the path will occur during a session, therefore ZRP without a route recovery procedure is not well suited for a large, highly mobile environment.

At this point it should be noted that SOAP presents great similarities to another two protocols: Landmark based routing [42] and LANMAR [120]. Those similarities, as well as fundamental differences are pointed out in the subsequent sections.

5.2 Protocol Overview

SOAP is developed based on the SO algorithm presented in Section 2.4. SOAP comprises three components: *clustering*, *tracking*, and *location management* algorithms.

In the proposed Self-Organizing Adaptive Protocol (SOAP) each node is running a modified version of the ZRP protocol – as will be explained in the next sections – and therefore each node is the center of its own *zone*. The *clustering* mechanism (presented in the next section) will determine that some nodes become reference nodes (RN) that will be used as 'beacons' or 'landmarks' for routing packets toward nodes in the reference areas (RA) around them. [1]

The SOAP clustering algorithm will determine the status for each node; possible values are: *assigned, around, free.* A node is *assigned* if it is inside a reference area. A node is *around* if it was inside a reference area in the past but currently is outside the reference area but less that 2k hops away from that reference area's reference node; i.e. at least one of the reference node's border nodes is a k-hop neighbor (that is, it is k or less hops away) of the node with status *around* and therefore it has up-to-date routing information about it. A node is *free* if it is not inside or around (see above) any reference area. Figure 5.1 shows an example of a network that at a given point in time has organized itself into three reference node/areas. The value of k has been set to 2. There are four nodes with *around* status. These nodes were inside one reference area a while ago. There are also 5 *free* nodes; these are nodes which it is not cost-effective to group into a new reference area. It is also shown that the reference areas may be overlapping and that one node determines its reference area not only based on distance but also taking into account the immediate past. It should be pointed out that although each node is the center of its own *zone*, there are only three reference areas, each associated with one reference node.

SOAP's *tracking* algorithm keeps downstream links toward every reference node. SOAP's *tracking* algorithm keeps a *destination table* in each node with the addresses of all the reference nodes. Thus, for each reference node in the network, the *tracking* algorithm running in a node will mark each of this node's links as 'upstream' or 'downstream' with respect to the aforementioned reference nodes. In Figure 5.2 the links toward the reference node R are shown. The downstream links form a directed acyclic graph (DAG), with node R as the root. TORA provides a similar DAG for each reference node.

SOAP's *location management* algorithm keeps track of each node's status and the reference area they belong to, if any. This information is stored in each node's *location table*, which has one entry for each possible destination in the network. [2] Each entry in the location table will have the address of the reference node of the last

[1] A reference area is the zone – set of k-neighbors – of a reference node RN.

[2] The mechanisms to initialize and maintain the *location table* are discussed in section 5.5.

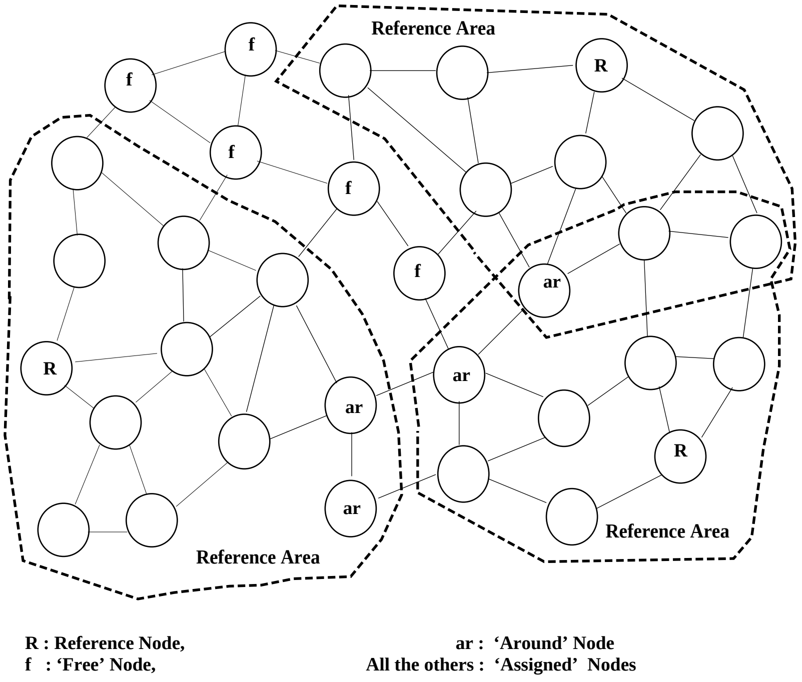


Figure 5.1: A self-organized network with three reference areas; k is equal to 2.

recorded position (reference area) of the node associated with that entry. Different nodes will have different values in the same location table entry, since some nodes may have more up to date information that others. It is also assumed that when a node moves from a reference area to another, the moving node will inform its past reference node (and only it) of its new reference node address. In this way if a packet arrives to a node that has been previously a reference node for the destination node, the reference node will forward the packet to the reference node in the next visited area. This procedure will continue until the packet reaches the current reference area of the node. The previous procedure allows for the sources not to have an up-to-date information on their *location tables* but still be able to reach their destinations. It should be noted that it is also possible that a node moves away from a reference area and does not enter another reference area. In such a case the last reference node will regard itself as the last reference node visited by the destination node.

If all the information is available, the proposed protocol will work as follows when forwarding packets: when a node requires to send a packet toward a destination it

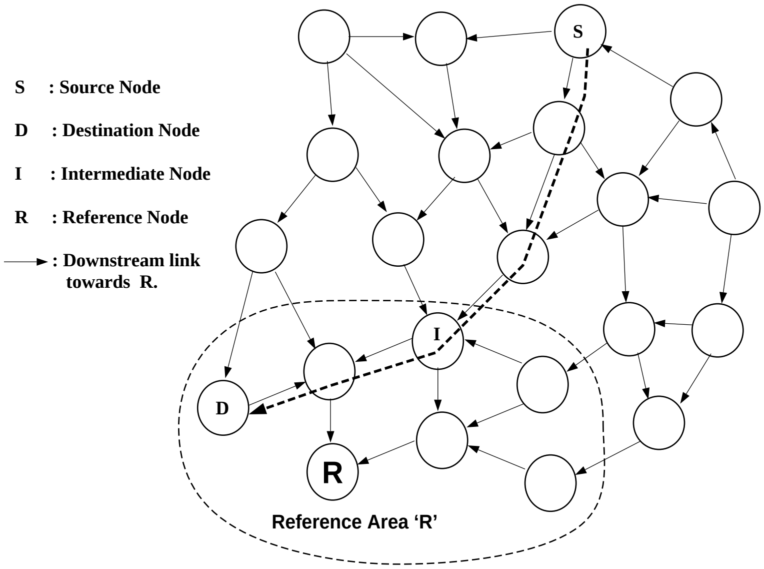


Figure 5.2: Packet routing using SOAP when location (reference area) information is up-to-date.

checks first if the node is within its own zone (IARP). If not it looks at its location table for the entry associated with the destination node. If the entry is empty, the source executes the IERP of ZRP, bordercasting a REQUEST through the network. If the location table entry is not empty, the source node sets the D_ENCAPSULATION flag on and encapsulates the packet putting the reference node address (Care-Of-Address) as the destination address and putting the actual destination address inside the IP payload (mobile IP). Also, the source puts its own reference node address as the IP source address and puts its actual address in the IP payload. This way the destination node (and all the nodes along the way) may update their location table entries, making it simple to return packets to the source node. The source node then sends the packet through the network using one of its downstream links associated with the destination address (as in the TORA protocol).

Each intermediate node in the network will look at the set D_ENCAPSULATION flag and will recover the actual destination address from the payload. If the destination is within the intermediate node's zone then the latter will forward the packet directly to the destination, but without decapsulating it. If the destination packet is not within the intermediate node's zone it will forward the packet through one of its downstream links associated with the Care-of-Address (TORA protocol). If the Care-of-Address is not a reference node, it must be within the intermediate node zone (as is the case for the *around* nodes, as explained later); in such cases the intermediate node forwards the packet in the direction of the Care-of-Address as if it were using IARP.

When a reference node receives the packet, it recovers the destination address and checks if the node is within its zone. If the node is, then it decapsulates the packet, set the D_ENCAPSULATION flag off and forwards the packet directly to the destination. If the destination node is not within the reference node's zone it checks if the node is reachable (status *around*, i.e., less than 2k-hops away). If the packet is reachable it re-encapsulates the packet putting as Care-of-Address the border node that is closer to the destination node (note that in this case the Care-of-Address is not a reference node). If the destination node is not *around* the reference node, the latter will look at its location table and will forward the packet to the reference node pointed to in the location table, putting the new reference node address as the care-of-address and forwarding the packet to one downstream link. If the reference node pointed in location table is this node itself, this node (reference node) will assume that the destination node is "unreachable" and will send a packet to the source node which will update its location table and will bordercast a REQUEST for the destination node.

Once the packet is received by the destination node, it will recover the source's reference node's address and put it in its location table. The destination node will use the new reference node address as Care-of-Address for any subsequent packet sent to the source node. Also any subsequent packet sent by the destination to the source will have the current destination's reference node's address and therefore the source may update its location table (Route Optimization procedure). Therefore, after the initial packets have been interchanged both source and destination will have knowledge of the most up to date location information of each other.

Figure 5.2 shows an example of the path followed by a packet when up-to-date location information is available (the middle of a session). The source node (S) sends the packet in the direction of reference node R. As the packet reaches an intermediate node I with knowledge of routes toward the actual destination (node D), it re-direct the packet to D. For this case, it may be noted a high degree of similarity between SOAP and a 2-level Landmark Routing [42] scheme. Reference nodes will be equivalent to global (level 1) Landmark nodes, and every destination will be regarded as level 0 Landmark. Besides these similarities there are fundamental differences between SOAP and Landmark routing, the more important being that Landmark was designed for static networks and makes no attempt to extract/learn the network nodes mobility pattern. In addition, Landmark mandates the use of distance vector for keeping track of level 0 Landmarks. SOAP however, requires extended link-state information (as will be seen later) not only for reaching level 0 Landmark nodes, but also in determining the best candidates for reference nodes. Further details presented in the remainder of this chapter will make some others, more-subtle differences evident.

Figure 5.3 shows the path followed by a packet in a case in which the location information is out-of-date. The packet will visit the past destination's reference nodes until reaching the current reference area. This long path is expected to be used only at the beginning of a session, since the destination node may include its current location information in any packet (e.g. acknowledgments) it sends back to the source, allowing subsequent packets to follow a more direct path.

From the above discussion, it may be seen that a bordercast is no longer necessary for the *assigned* and *around* nodes, therefore significantly reducing the number of bordercasts in the network.

It should also be pointed out that during normal communication, a packet addressed to an *assigned* node does not necessarily have to go through the destination's reference node, but any node in the destination zone may forward it. Since it is expected that the majority of the nodes in the network will have status *assigned* it can be said that the proposed protocol attempts to distribute the load within the network. However, it should be noted that packets addressed to *around* nodes are more likely to pass through the destination's reference node; and also packets with outdated Care-of-Address will require to pass through a sequence of reference nodes before reaching destination. These two operations will result in an increase in the

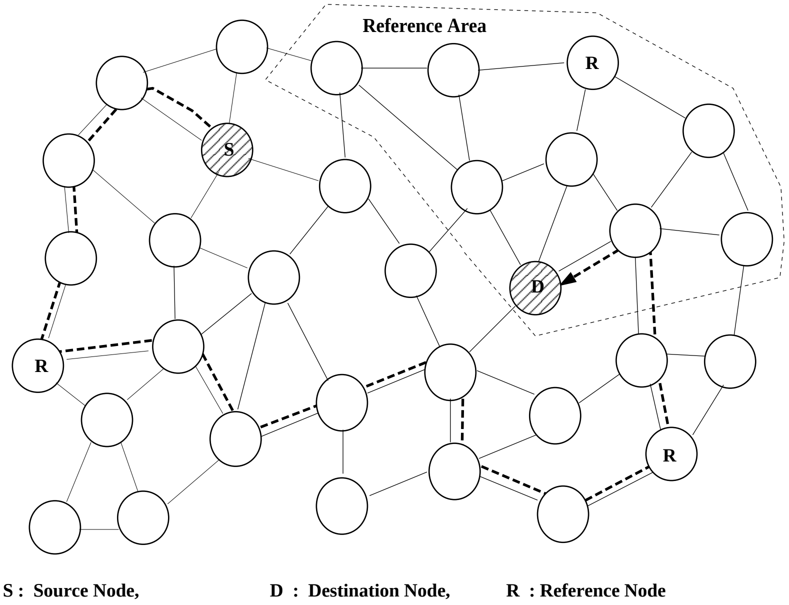


Figure 5.3: Packet routing using SOAP when location (reference area) information is out-of-date.

traffic going through the reference node's links somehow creating an unfair traffic load (burden) on the reference nodes. Nevertheless, this effect may be reduced (or eliminated) if non-reference nodes forwarding packets to *assigned* nodes within their zones (the most of the cases) are instructed to choose paths that avoid interfering with reference node's links, whenever possible. In this way, the reference nodes are 'relieved' of some traffic burden and the network maintains some good balance.

The three elements of SOAP are described in the next sections. The *clustering* algorithm is discussed in Section 5.3, the *tracking* algorithm is presented in Section 5.4, and the *location management* procedure is explained in Section 5.5. The protocol description is presented in Section 5.6 and performance expectations are discussed in Section 5.7. Some applications and modifications for particular situations are discussed in Section 5.8, and finally some conclusion and new research directions are treated in Section 5.9.

5.3 Clustering

The Clustering algorithm is responsible for choosing the reference nodes and for assigning to every node some of the following values {*assigned, around, free*} with respect to a reference node. A node is *assigned* if it is inside a reference node's zone, i.e. it belongs to a reference area. A node is *around* a reference node if it was *assigned* to the reference node but now it is outside the reference area but less than $2k$ hops away, so that at least one of the reference node's border node is inside the node's zone. A node is *free* if it is not associated with any reference node, probably because it is not worthy tracking this node location. In Figure 5.1 an example of a clustering is shown.

As an initial point for this discussion let's consider a network that is running the ZRP protocol. From the discussion above (Section 5.2), it is clear that there is a benefit in grouping all the nodes *assigned* with a reference node in a reference area, mainly because the number of bordercasts needed is significantly reduced with respect to ZRP since bordercasts are needed to reach *free* nodes only. The objective of the clustering algorithm is to choose the nodes which are the best candidates for being a reference node in the sense that they maximize a 'Gain' metric. This Gain will be compared with a Cost metric (extra bandwidth overhead with respect to ZRP) of maintaining a reference area.

In the following subsections the Gain and Cost functions are presented together with an heuristic for (suboptimal) clustering of the network.

5.3.1 The Gain Function

The gain function of a node A at time t and at level L was defined in Subsection 2.4.3 as (equation 2.1, being reproduced here for clarity) :

$$G_L(A,t) \quad = \quad \sum_{i \in V(A,t)} S_i(A,t,L) R_i(t,L)$$

Where $S_i(A,t,L)$ is the estimated percentage of the next L seconds that node i will stay inside node A's zone, i.e. within a radius of k from A. $R_i(t)$ is the total expected number of new sessions over the next L seconds, and $V(A,t)$ is the set *free* or *around* nodes that are in node A's zone (i.e. k or less hops away).

Subsection 2.4.3 presented a rationale for the choice of estimator of $S_i(A,t,L)$. Following the same rationale and trading off the estimation quality and implementation complexity, the estimator of $S_i(A,t,L)$ was chosen to be :

$$\hat{S}_i(A,t) \;=\; (1-\rho)\sum_{j=0}^{\infty} \rho^j \, Asso(A,i,t) \tag{5.1}$$

Where the L index was dropped. $Asso(A,i,t)$ is a function representing the instant association between nodes A and i at time t, and $0 < \rho < 1$ is the forgetting factor. The forgetting factor determines the extend of the 'memory' of the estimator. Larger values of ρ will imply long memory and therefore slower reaction to instantaneous variations. Smaller values of ρ will reduce the memory length and result in a faster reaction to changes.

It should be noted that although t represents time, in equation 5.1 it has a different interpretation. Since $\hat{S}_i(A,t)$ is computed periodically, t only takes discrete values and can be though of a time index. Thus, $\hat{S}_i(A,t)$ can be easily computed by means of the following recursion:

$$\hat{S}_i(A,t+1) \;=\; \rho\,\hat{S}_i(A,t) + (1-\rho)\,Asso(A,i,t+1) \tag{5.2}$$

Implementing equation 5.2 (initialized to $\hat{S}_i(A,0) = 0$) has a much lower computational cost than implementing equation 5.1. It limits to (for each node in the network) to keep the past value of $\hat{S}_i(A,t)$ and the current value of $Asso(A,i,t+1)$. Thus, we can drop the time argument and from now on refer to $\hat{S}_i(A)$ and $Asso(A,i)$.

The association function $Asso(A,i)$ is chosen to be a function of the distance between nodes A and i $(d(A,i))$. If node i is not reachable using node A's topology table entries it is assumed that $d(A,i) = \infty$. this distance is infinite. Thus, an obvious choice for $Asso(A,i)$ is:

$$Asso(A,i) \;=\; \begin{cases} 1 & \text{if } d(A,i) \le k \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

However, to add some flexibility, two different association functions ($Asso_1(A,i)$ and $Asso_2(A,i)$) are considered:

$$\begin{aligned} Asso_1(A,i) &= \begin{cases} bonus & \text{if } d(A,i) \leq k \\ -penalty & \text{otherwise} \end{cases} \\ Asso_2(A,i) &= \begin{cases} [k - d(A,i)]\, bonus & \text{if } d(A,i) \leq k \\ -[d(A,i) - k]\, bonus & \text{if } k < d(A,i) \leq 2k \\ -penalty & \text{otherwise} \end{cases} \end{aligned}$$

Where *bonus* and *penalty* are protocol parameters. Equation 5.4 is a straightforward generalization of equation 5.3, that tries to penalize nodes that temporarily leave the reference node's zone. Equation 5.4 is a more elaborate scheme where the nodes closer to the reference node (candidate) are assumed to have a higher association with it. Also, nodes that are at a distance more than k but less than $2k$ hops from the candidate for reference node will see its penalty increase linearly with distance. Nodes that are more than $2k$ hops away will be regarded unreachable and they will be given the maximum penalty. Note that care should be taken so that $\hat{S}_i(A,t)$ should ever e negative. So, if at the current iteration $\hat{S}_i(A,t)$ is less than zero according to equation 5.2, then this value should be overwritten to 0.

Finally, for the $R_i(t,L)$ estimation, feedback from the upper layer should be employed. In our current implementation (simulations) we will use two modes: *traffic feedback disabled* ($\hat{R}_i(t,L) = 1$ always); and *traffic feedback enabled* ($\hat{R}_i(t,L) = 1$ if i is the destination of a traffic stream, and 0 otherwise). In real life scenarios, the value of $\hat{R}_i(t,L)$ should be provided by a higher layer traffic estimator.

5.3.2 Cost Function

The rationale for the choosing of the SO algorithm cost function has been discussed in Subsection 2.4.4.

In the current implementation of SOAP, the cost function is not adaptive, but it is input at the beginning of the simulation.

5.3.3 Reference Area Formation

A reference area should be created only when it is worthy. In the case of a potential reference node A, if there were not a reference area centered in A then the initial packets of a session sent to A (or any node inside node A's zone) should be routed using ZRP (bordercasted). The gain of not bordercasting has to be compared with the cost associated with the creation of a reference zone in order to determine if it is worthy to create the reference zone. Therefore, the cost function defined in Subsections 2.4.4 and 5.3.2) is a threshold against which the gain of the better suited candidate to become a reference node (defined in Subsections 2.4.3 and 5.3.1) has to be compared.

ZRP considers that each node informs its k-neighbors when a change in link state occurs. SOAP considers that a node i sends extra information to its k-neighbors along with the ZRP's information. The information is sent after a change in link status or after a timer expires (similar to ZRP and NSLS). The extra fields are:

$$\Big(\; status \quad , \quad \hat{R}_i(t) \quad , \quad G_L(i,t) \quad , \quad RF_p \quad , \quad RF_1 \quad , \quad \ldots \quad RF_n \; \Big)$$

Where *status* has three possible values : *free, around, assigned* . $\hat{R}_i(t)$ and $G_L(i,t)$ were introduced before. RF_p is the reference node to which node i is assigned (if any) and $RF_1, \ldots, RF_n$ are , in order of distance, all the other reference nodes in whose area node i belongs as well (are node i's k-neighbors).

Additionally, a node has to send at least one update packet every T seconds. Therefore, if for the last T seconds there have not been any link status change the node will have to send an update packet. The value of T should be chosen so that the updates triggered by this time out are infrequent. Therefore T should be chosen to be greater than the mean failure time of a link. In a highly dynamic environment where the changes in link status happen frequently and for moderate values of T, most updates will be triggered by link status changes.

Node A uses the information presented above to constantly update its gain $G_L(A,t)$ considering only the *non-assigned* nodes (i.e. with status *free* or *around*) in equation (2.1). If $G_L(A,t)$ is not greater than the gain of all the other *non-assigned* nodes inside its zone, then no further action is taken. If $G_L(A, t)$ is greater than the gains of all node A's *non-assigned* k-neighbors, then node A is a candidate to be a reference node. In the case that two or more nodes have the same gain, the node with (for instance) the higher address will be the candidate to be a reference node.

If node A is a candidate to be a reference node then it compares its gain $G(A)$ to the cost threshold presented in subsection 5.3.2. If the gain is greater, then node A becomes a reference node and sends a broadcast packet all over the network. There is a field in the broadcasted packet that contains the cumulative height with respect to the reference node and that is initialized to the current time plus zero, according to the TORA protocol. Node A also includes additional information in this broadcast as will be explained later.

When a node receives the broadcast of A it reads the cumulative height and increases it by one. If this height is lower than its last value (initially in NULL, and non-NULL only after the first reception of the broadcast packet) the node updates its height associated with the reference node and forwards the packets to their neighbors. After all the nodes in the network have received the broadcast from node A they all have at least one downstream link toward node A and the entire network is a directed acyclic graph (DAG) rooted at A.

After the DAG has been formed, the network will continue tracking node A using the route maintenance mechanism of TORA. The main difference with the current TORA implementation is that routes that have been deleted (because of network partitions) are being reestablished as soon as the connectivity is recovered. The links are going to be definitely erased (change to undirected state) only when node A broadcasts another packet telling that it is no longer a reference node, as explained below.

All the nodes within the zone of A change their status to *assigned* and record A as their reference node.

It should be noted that there is at least one node – the one with the global maximum gain – which has the maximum gain among its neighbors. Therefore, if the threshold (cost) is sufficiently low (favoring the creation of reference areas) then at least one reference area is created and the nodes within this area change their status to assigned. Therefore, the set of *non-assigned* nodes is reduced and a different node has the new (smaller) global maximum gain. The latter node becomes a new reference node and so on; the procedure continues until a large portion of the network is grouped in reference areas. The number of nodes left *free* depends of the value of the threshold.

It should also be noted that a node is not selected to be a reference node unless it is willing. If a node does not send the broadcast packet announcing itself as a reference node, then it will not become a reference node even if its gain is greater that its k-neighbors'. Such a node should put a value of zero instead of its actual gain in their next transmission.

The resulting set of reference areas would represent the structure (hierarchy) of the network at a particular moment in time. After this initial setup, reference areas may split or may combine and new reference areas may appear. Every time a node has a gain greater than its neighbor's and also greater than its threshold and it is willing, it will become a reference node. Therefore, it may be interpreted that every time a large set of nodes is left outside the existing reference areas these nodes will tend to form a new reference area.

5.3.4 Reference Area Deletion

All the possible changes in reference areas : creation, splitting, combination, and deletion are the result of two basic procedures, namely creation and deletion. The algorithm employed to create a new reference area has already been explained and is running permanently at each node. In this subsection the algorithm used to delete existing reference area is explained.

The deletion algorithm is based in the computation of what is called the *minimal gain*. The *minimal gain* of the reference node A is the gain of A computed considering only the nodes within the zone of A that are not within the zone of any other reference node. We can see the minimal gain as the actual contribution of node A since the other nodes could be taken care of by the other reference nodes.

For a reference node A to disappear two conditions have to be met : the *minimal* *gain* of node A has to be lower than the minimal gain of all its neighboring reference nodes[3] and the *minimal gain* has to be lower than the value *threshold_del*. A high value of *threshold_del* will result in frequent reference area deletion/creation and increase bandwidth overhead. Thus, *threshold_del* should be set to be a small fraction of the cost function discussed in subsection 5.3.2 but taking into consideration that a value so small that results in infrequent changes in reference area topology may reduce the algorithm ability to adapt (quickly) to varying network conditions.

In order to guarantee that the two conditions are met, a reference node that detects that its minimal gain is lower than *threshold_del* sends a packet to every neighboring reference node informing them that it is going to be deleted. The node will be deleted only after all the neighboring reference nodes acknowledge its packet and agree with it. A neighboring reference node that receives a message from a reference node that is going to be deleted compares that node's minimal gain to its own and acknowledges agreeing only if its minimal gain is greater. In such a case (reference node is going to be deleted), the neighboring reference node attaches to the message its current gain along with a list with its distance in hops to the nodes it wishes to take care of (including any node with status *around*).

Before deleting itself, the reference node decides (based on the distance) which will be the new reference node (if any) for each of its *assigned* or *around* nodes. In case there is no node willing to take care of a particular node, that node will change its status to *free*. After that, it sends a broadcast message to the entire network including a list of its previously associated nodes with their new reference node's addresses. The links toward the deleting node are changed to undirected and the nodes previously associated with the deleted node record the identity of their new reference node (from the broadcast).

[3] Two reference nodes are said to be neighbors if there is at least one other node that is within the zone (i.e. k hops away) of both reference nodes. Reference node A knows all its neighboring reference nodes based on the information being sent by all its k-neighbors as explained in previous subsection.

5.3.5 Handoff

Node i originally assigned to reference node A is said to have executed a handoff from reference node A to reference node B if node i decides to have node B as its new reference node.

When a node i originally in node A's reference area moves more than k hops away (outside node's A reference area) it changes its status to *around*, but it is still associated to node A. It will remain associated with node A as long as its distance in hops is less than $2k$, that is, at least one node inside node A reference area is inside node's i area. If the distances in hops between nodes A and i is equal to or exceeds $2k$ then node i status become *free* until it is associated with (less than k-hops away from) another reference node. In the latter case, node i will change its status to *assigned* again and will have to notify node A of its new reference area as required by the location management mechanism explained in the next section.

Nodes in *around* status have to send a packet to their associated reference node – in addition to their k-level broadcast (as required by the ZRP protocol) – indicating their closest k-neighbor that is inside its associated reference node area (i.e. a border node of the reference area). Based on this information, the reference node will always be able to forward packets to the *around* nodes.

If at any moment a node that has status *around* detects that it is within another reference node zone (i.e. less than k-hops away) and that the distance to its previous associated reference node is greater than the distance to the new reference node plus *extra_hops* then this node decides to execute a handoff to the new reference area. The node will change its status to assigned and will inform its previous reference node of its new reference area as required by the location management mechanism explained in the next section. The value of *extra_hops* is initially set to 2, and is intended to provide with some 'capture' effect to prevent a (border) node from being constantly oscillating between neighboring reference areas. Also, no *assigned* node executes a handoff even if the distance to another reference node is smaller than its distance to its current reference node. Thus the overhead caused by unnecessary handoffs is reduced.

5.4 Tracking Algorithm

SOAP will employ a tracking algorithm based on the TORA protocol. Section 5.6 discusses the differences between SOAP's tracking and TORA. Mainly the fact that in SOAP downstream links toward each reference node are constructed upon reception of the first bordercast sent by a newly formed reference node, and it is constantly updated (refreshed) upon reception of such updates.

The main reason by choosing TORA for SOAP's tracking mechanism was TORA's ability to handle highly mobile scenarios and its ability to recover from partitions.

However, a variant of Distance Vector could also have been chosen. It is not clear which one is the best alternative, and most likely each will be a better alternative under different scenarios. Finally, geographical based routing could also be chosen as SOAP tracking mechanism. This possibility may be explored in the future.

5.5 Location Management

The objective of the *location management* algorithm is to provide each intermediate node with the last known reference node a destination is (was) associated with, if any. To this end, the *location management* algorithm running at each node keeps a table, referred to as *location table*, associating any possible destinations with its last known reference node (if any).

The location management algorithm functionality may be split into 2 procedures: *initialization* and *maintenance*.

5.5.1 Initialization

Each node has a location table with as many entries as known destinations. Each destination has initially associated the value NULL meaning that no information is available about the location of that destination.

When a new reference area is formed the new reference node sends a broadcast all over the network. In this broadcast the reference node indicates the nodes that belong to its reference area. All the nodes receiving this broadcast will update their location tables putting the new reference node address as the current location of the nodes mentioned in the broadcasted packet.

For example, if node A decides to be a reference node for the nodes B,C,D, and E then node A will send a broadcast packet indicating that nodes A,B,C,D, and E belong to its reference area. If node i receives this broadcast, node i will update the location table entries of nodes A,B,C,D, and E with the address of node A.

After several reference areas have been formed, a number of entries of the location tables of the network nodes will be filled. NULL entries will still be present in the location tables corresponding to *free* nodes.

5.5.2 Maintenance

There are the following maintenance procedures:

- When a node changes its reference node (handoff) it will inform the previous reference node - only - of its new location (reference node). The previous reference node will update the corresponding location table entry and this node will serve as a pointer toward this destination.

- In any packet sent, the source IP address will correspond to the current reference area (if any) and the actual source node address will be in the payload. A node that receives the packet will extract the source node's reference area information and will update its location table accordingly.

- After a number M of handoffs a node will send a new broadcast informing all the nodes on the network of its current position. The optimal value of M will depend on the network conditions and will be specified later. A small value of M will imply too many updates being propagated (bandwidth waste) and a too large a value of M may result in out-of-date information producing inefficient initial source-destination paths congesting the areas close to the reference nodes and degrading the network performance.

- When a reference node is deleted it sends a broadcast informing the network not only of its deletion but the new reference areas of the nodes previously inside its reference area.

5.6 Protocol Description

The proposed SOAP protocol uses a combination of the mechanisms of the Mobile IP, TORA, and ZRP protocols. SOAP recognizes and uses the REQUEST, REPLY, and FAILURE packets of the ZRP protocol and the QRY (query), UPD (update) and CLR (clear) packets of TORA. Also a *binding update* packet is used to update the nodes' current location when needed. Instead of explaining all the details contained in these protocols, only the differences with the proposed protocol are presented.
Major differences between related functionalities in SOAP and Mobile IP

- The binding database in Mobile IP is replaced by the location table in SOAP.
- Every node in the network has a location table, as opposed to only the Home Agent being equipped with the binding database in Mobile IP.
- In SOAP it is assumed that every packet has a D_ENCAPSULATION and a S_ENCAPSULATION bit flags. The D_ENCAPSULATION (S_ENCAPSULATION) flag is set on if the destination (source) IP address is not the actual address but the address of one reference node. The actual destination (source) address is inside the payload.
- In SOAP, when a node receives a FAILURE packet, it sets the location table entry of the destination referred to by the FAILURE packet to NULL. FAILURE packets were previously used only by the ZRP protocol.

Major differences between related functionalities in SOAP and TORA

In SOAP only the reference nodes are TORA destinations so that the two terms are used equivalently (reference nodes and TORA destinations). The nodes keep a table of the current destinations (*destination table*). When a new physical link is established, the new neighboring nodes exchange among other information their height with respect to the current destinations, therefore assigning a value (upstream or downstream) to the physical link. The following modifications to TORA need to be employed, to account for TORA's role in SOAP.

- In TORA the routes toward a destination are generated after a query from the source (source initiated - on demand) while in the present protocol the initial routes toward a destination (reference node) will be constructed based on the broadcast sent by the reference node at the time of the reference area's creation. When a node first receives the mentioned broadcast it records the original sender's address and the time the reference area was created in its destination table. It will set its height to the value retrieved from the broadcast message. Finally, the receiver node increases by one the height field received in the broadcast packet and resends it. If the node receives the same broadcast again no action is taken unless the height is lower than the one recorded. In such case, the node proceeds as before updating the height and resending the broadcast. It is clear that the route creation is less costly in SOAP than in the original TORA.

- When a new node enters the network it will receive – from its neighbors – the information about their height with respect to the current reference nodes. This way the new node will be informed of the reference nodes in the network and will set its height (for each reference node) slightly higher than the highest of its neighbors.

- Similarly, when a node that has its height toward a destination with the value NULL (possible after a network partition) establishes connectivity with a node that has a non-null height it will set its height slightly higher than its neighbor's. This means that the network will try to continue tracking a reference node as soon as some connectivity is reestablished following a network partition.

- The nodes stop tracking a destination only if the destination ceases to be a reference node. The broadcast packet sent by the deleted reference node serves as the CLR (clear) packet in the original TORA protocol. Not only the links are set to undirected (changing their height to NULL) but the reference node is removed from the list of destinations; that is, there is no height associated to that destination and the nodes stop broadcasting information related to this destination when new physical links are established. The time the reference node was deleted is recorded.

- When network partitions occur it may be possible that different nodes have different destination tables. When two nodes realize they have different destination tables they send each other their last recorded creation/deletion time. The more recent event takes precedence. The nodes update their destination tables and heights and propagate the information to their neighbors as necessary.
- TORA does not have a mechanism to choose best-cost routes. In SOAP a greedy approach is taken. In SOAP, when a node needs to forward a packet to a destination it will choose the lowest cost link among its downstream links. In this context, the cost of a link is equal to the number of nodes that shared the link over the (current) available bandwidth of the link (the same cost function presented in [99]).
- In SOAP, if a node has to forward a packet to a reference node that is unreachable (maybe because of a network partition) then the node has to send a FAILURE packet to the source node of that packet. TORA does not provide such a mechanism.
- Infrequent updates (broadcasts) need to be considered to compensate for route degradation over time. The original version of TORA leaves the possibility open but does not require it or specify any procedure.

Major differences between related functionalities in SOAP and ZRP

- The amount of information that a node has to send to its k-neighbors is larger in SOAP than in ZRP. The new fields were introduced in subsection 5.3.3. These fields provide the nodes with the information necessary to decide to create/delete reference areas.

- In SOAP, the nodes send updates not only when there is a change in their connectivity but also after a period of T seconds. T should be chosen in order to provide for infrequent updates which do not lead to an excessive traffic burden.
- IN SOAP, nodes whose status is *around* have to send the updates not only to their k-neighbors but also to their reference node.
- In SOAP, when a node receives a FAILURE packet, it sets the location table entry associated to the destination to NULL.
- In SOAP, when forwarding packets using IARP the nodes will try to avoid paths that interfere with a reference node's links. ZRP does not consider such a mechanism.

With the modifications outlined above to Mobile IP, TORA, and ZRP, the following actions have to be executed by the nodes involved in the routing of a packet sent by node i to node j in order to induce the protocol behavior presented in Section 5.2.

5.6.1 Node i (Source Node)

- If node j is within node i's zone then the latter forwards the packet using IARP-ZRP (no encapsulation is necessary and the D_ENCAPSULATION and S_ENCAPSULATION flags are set OFF).
- If node j is not within node i's zone, node i looks at its location table for the node j's reference node address (if any) :
  - If an entry is found. Let node A be the recorded reference node for node j. Node i encapsulates the packet putting node A's address as the IP destination address and its own reference node's address (if any) as the IP source address. Inside the payload node i will put node j and its own (node i) addresses. The D_ENCAPSULATION and S_ENCAPSULATION flags are set ON. If node i does not have a reference node, the S_ENCAPSULATION flag is set OFF and the source IP address will be equal to node i address.

The packet will be forwarded across the network using the 'downstream' links established by the TORA algorithm.

- If no entry is found. Node i uses the IERP-ZRP protocol. It looks at its IERP table for the route toward node j and sends the packet to the next border node in that route. The destination IP address is equal to node j address and the D_ENCAPSULATION flag is set OFF. Also, the S_ENCAPSULATION flag and the source IP address field will be set depending on whether or not node i has a reference node as in the previous case. If there is no known route toward node j, node i bordercasts a REQUEST. The rest of the (border) nodes receiving a REQUEST react to the REQUEST as specified in the ZRP protocol.

5.6.2 Node k, (Intermediate Node)

If an intermediate node receives a REQUEST packet, it will react as specified in the ZRP protocol. Therefore, only the case where a data packet is received is considered here.

When node k receives a data packet (i.e. a packet that is not a REQUEST), it reads the S_ENCAPSULATION flag and updates its location table entry corresponding to the source node. Also, the node reads the D_ENCAPSULATION flag and recovers the destination address. Node k compares the destination address to its own. If they are equal it is implied that node k is the destination, therefore it executes the procedure explained in subsection 5.6.3. If node k is not the destination node, it checks the D_ENCAPSULATION flag again.

- If the D_ENCAPSULATION flag is OFF, the node tries to forward the packet using the IARP-ZRP protocol. If this is not possible the node sends a FAILURE packet to the source node.

- If the D_ENCAPSULATION flag is ON, the intermediate node recovers the actual destination address (node j in the previous example) and the presumed (maybe out-of-date) reference node's address.

- If node j is within the intermediate node's zone. The intermediate node forwards the packet directly to node j, using the IARP of ZRP – either Link State (LS) or Distance Vector (DV) algorithm – and bypassing the reference node if possible. The packet is still D_ENCAPSULATED.
- If node j is not within the intermediate node's zone. Node k compares the presumed reference node's address (IP destination address in the packet) with its own address.
  - If they are different, node k forwards the packet D_ENCAPSULATED along its less costly 'downstream' link.
  - If they are the same. Node k is the presumed reference node. If node k is not a reference node then it sends a FAILURE packet to the source. The other possibility is that node k is a reference node but node j is not within its zone. It may be because the source location table is out-of-date or because the node j is *around*. Node k checks its location table for node j's reference node address looking for a more up-to-date reference node information and the node status.
    - If a more up-to-date reference area is found, node k forwards the packet to the new reference node D_ENCAPSULATED using TORA.
    - If node k does not have a more up-to-date reference node and node j status is *assigned*: An error occurred. Node k sends a FAILURE packet to the source.
    - If node k does not have a more up-to-date reference node and node j status is *around*. Node k knows which border node is closer to node j. Node k sends the packet to node j using ZRP. The D_ENCAPSULATED flag is OFF and the accumulated route toward j consists of only two terms : the border node closer to node j, and node j itself.
    - If node k does not have a more up-to-date reference node and node j status is *free*. Node k generates and sends a limited-depth

bordercast (D_ENCAPSULATED OFF) trying to reach node j's current position. After receiving the location update or after a timer expires, node k sends a packet to node i updating node j's location table entry to the new found reference node address (if any), or to NULL (a FAILURE packet). Node i will use the new reference area (or NULL) value for successive packets.

· If node k does not have a more up-to-date reference node and node j status is *assigned*: An error occurred. Node k sends a FAILURE packet to the source.

5.6.3 Node j (Destination Node)

When the destination (node j) receives the packet it updates its location table with node i's reference node address (if any), removes the overhead and passes the data to the upper layers.

5.7 Performance Results

Given the common scenario in which a node's mobility follows group patterns (i.e. group mobility), proper choice of the reference area will dramatically improve performance. This has recently been shown/confirmed by the results obtained by the LANMAR protocol [120]. In [120], it is shown that even for moderate size networks (100 nodes) Landmark based routing outperforms AODV, DSR, and FSR by over 100% with respect to throughput under high mobility and traffic loading.

LANMAR is an extension of the original work in Landmark Routing [42] to mobile scenarios. LANMAR presents certain similarities with SOAP and was published in the time between SOAP's initial development and publication [121, 122] and the time of the preparation of this dissertation [4]. LANMAR is similar to SOAP in that both try to exploit group mobility, with similarities in the tracking of destinations (level 0 landmarks) by means of link state information dissemination. However, SOAP

[4] Thus, SOAP was developed independently and published *prior* to LANMAR.

uses NSLS (or, equivalently ZRP's IARP with link state updates) as the underlining limited information dissemination mechanism, LANMAR uses FSR. Furthermore, SOAP uses TORA to track reference nodes (global landmark nodes in LANMAR terminology), LANMAR utilizes distance vector routing. At this point, it not clear which choice is the best (TORA versus distance vector), and it is likely that the answer depends on the particular scenario.

There is, however, a fundamental difference between LANMAR and SOAP. In LANMAR, the landmark nodes (reference nodes) are pre-assigned. LANMAR assumes the mobility pattern (i.e. the group mobility) is known beforehand. SOAP, on the other hand, is an adaptive algorithm that represents an instantiation of the self-organizing (SO) algorithm, which is a fundamental part of the multi-mode routing framework introduced Chapter 2. SOAP uses the algorithm presented in section 5.3 to determine the best candidates to reference nodes, and in doing so it tries to learn/extract the network structure and use it for effective routing. SOAP understands the network structure to be defined not only for topology and mobility patterns (e.g. group mobility, or other mobility pattern) but also by the traffic patterns, as reflected in the gain computations. This is a significantly better approach than that of LANMAR. In the *worst* case when there is little or no group mobility, or the pattern is not extracted accurately, SOAP can perform *as well as* LANMAR. However, it is clear based on the design alone, that with high probability SOAP will significantly outperform LANMAR.

Thus, under group mobility scenario SOAP will work well. Similarly, SOAP can exploit non-uniform traffic distributions to improve its performance relative to LANMAR. However, the most crucial observation is that SOAP is not limited to a predefined scenario. SOAP is designed to adapt, hence, it will be effective under a wide range of scenarios. For example, if only some nodes that are close by are traffic destinations, SOAP may choose to create only one reference area that includes all the traffic destinations. Thus, only one reference node will be tracked, optimizing performance. Other protocols lack this adaptation capability. LANMAR, for example, will still track each of the pre-assigned group leaders.

Finally, from a conceptual point of view, SOAP may be considered to be a simplified implementation of a multi-mode routing protocol by itself—built according to the framework proposed in Chapter 2. The *limited information dissemination* module consists of SOAP's link state propagation algorithm based on ZRP's IARP, that corresponds to the NSLS algorithm presented in Chapter 3. The *clustering*, *tracking*, and *location management* modules constitute a simplified version of the *self-organizing* module. And finally, the higher level protocol decisions to either send the packet along a downstream link toward a reference node, use routes computed using the topology database (built with link state updates), or generate a route discovery query based on ZRP's IERP's bordercast procedure constitute a reduced set of the decisions that a *multi-mode routing engine* module has to make. Thus, conceptually and in its performance SOAP is substantially different from, and will outperform LANMAR.

Even though SOAP's strength comes from its adaptation to different scenarios, it is also of interest to asses SOAP performance under a particular one. In this section we explore the performance of SOAP under group mobility, mainly because group mobility scenarios are highly likely patterns in human communication and mobility. Furthermore, keeping the previous discussion in mind, this approach allows the results to build upon recent performance results for LANMAR.

LANMAR results [120] showed that using landmarks (reference nodes) to 'summarize' routes to several destinations significantly improves the routing protocol performance in presence of group mobility if the mobility leaders were known beforehand. This can be taken as a forgone conclusion—or as our starting point. In practical situations we may not know the identity of the group leaders) or even may want to adapt to time varying patterns), so we wonder whether the performance improvements promised by LANMAR can be achieved even if no *a priori* information about the group leaders is available. It is obvious that if LANMAR is not fed with proper inforamtion about the group leaders, it will not be able to deliver his positive performace results. But, if SOAP is employed, mobility information can be learnt (estimated) on-the-fly by using the algorithm presented in this chapter. The important questions are *how much performance degradation is experienced when the mobility leader's information is not supplied but estimated on-the-fly?*, and *what is the additional performance*

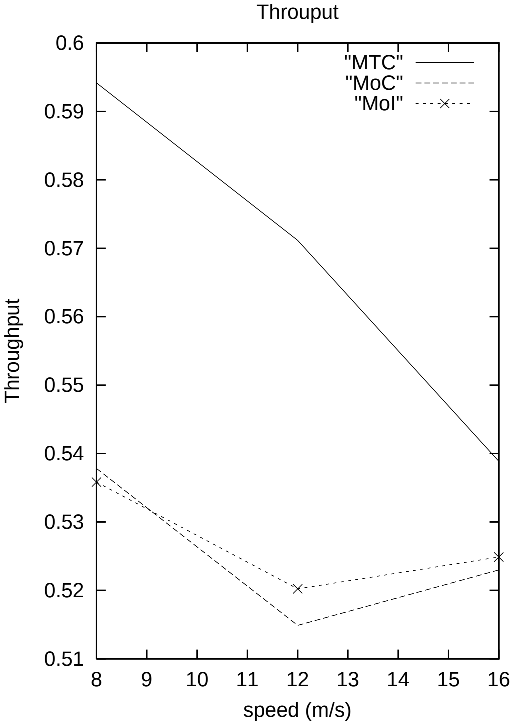


Figure 5.4: Throughput obtained by several SOAP's variants running in a 100 node network under different speeds.

improvement that can be achieved by considering the traffic patterns in addition to the mobility pattern?.

To analyze these questions both SOAP and a modified version of SOAP were implemented in the OPNET simulator. The modification was made such that the advertised gain function was not computed on-the-fly based on the estimators defined in Subsection 5.3.1, but was input as a fixed parameter for each node.

Figure 5.4 shows the results obtained for network throughput in a 100 node network subject to differing node velocity. The simulation testbed was identical to the one used in the previous chapters (simulation tool was OPNET, MAC and radio models were the same as before, etc.). The main difference is that the mobility pattern is no longer random but nodes move in groups. Each group consists of 20 nodes, hence, there are 5 groups.

MoI (Mobility only, Ideal) refers to the SOAP variation in which the nodes' gain functions are input as parameters, and the larger values of gain are assigned to the mobility leaders, forcing them to be the reference nodes (emulating LANMAR behavior). Since this is the best performance that a mobility-learning algorithm could achieve it models the ideal case, although it does not mean that this performance (or better) can not be achieved by an adaptive algorithm (if reference node selection is effective) or even surpassed (if traffic requirements are also taken into account).

MoC (Mobility only, Computed) models the best SOAP results under the *traffic feedback disabled* mode, that is, $\hat{R}_i(t, L) = 1$ for every i. Since there are 2 choices for the association function $Asso(A, i)$, only the best performance is reported. MoC is choosing the referene nodes based on mobility patterns only. The results show that SOAP successfully learn mobility information and delivers similar results as LANMAR without knowing the identity of the group leaders beforehand. We can see that even in one case (8 m/s) MoC outperforms MoI. This may seem counterintuitive, but it is explained by the fact that the nodes move randomly around the group leader, who is expected to be the 'center of gravity' of the group and therefore the best candidate to be reference node. MoI uses this knowledge of the expected behavior of the nodes for selecting the reference nodes. MoC, in the other hand, computes each node's gain based on the actual node movements, and chooses the node that is actually at the center of the group during the simulation lifespan. Thus, sometimes there are nodes that are better choices of reference node than the group leaders for a moderate amount of time. In the other hand, nodes chosen by MoC to be reference nodes may become less representative of the group as time elapses, forcing MoC to reselect the reference node. The simulation results for 8 m/s shows that for the 900 seconds of simulation time, there were nodes more representative of the group than the (predefined) group leaders.

MTC (Mobility and Traffic, Computed) models the best SOAP results under the *traffic feedback enabled* mode, that is, $\hat{R}_i(t, L) = 1$ if i is a destination and 0 elsewhere. Once again, only the result obtained under the best performing association function $Asso(A, i)$ is reported. This result shows that by considering the traffic patterns the routing protocol performance can be improved. We expect the amount of improvement to increase even more for larger networks. Note that MTC's performance is an upper bound to the performance that may be obtained by considering the traffic pattern, since it is assumed to be a perfect estimation. In practical situations this may not be case. However, MTC's results suggest that substantial performance gains can be obtained by providing the routing module with traffic estimation information from the upper layer.This information may range from *a priori* estimation of incoming sessions, based for example on the user role in the network (e.g. database server, mission commander, etc,), to on-the -fly estimations based on recent history and future application layer requirements.

5.8 Extensions

The present protocol was initially conceived to work in a horizontal highly mobile network. In this section, the protocol behavior in different scenarios is investigated.

5.8.1 Superclusters

In an ad-hoc network, the number of destinations increases with the network size. Beyond a certain point it may be too costly to keep track of all the clusters (reference nodes). Under such conditions it would be effective to group several clusters into superclusters. The nodes would then keep track of all the superclusters in the network as well as the clusters that belong in the same supercluster with that of a particular node. A destination node should be referred to by using its supercluster (reference) node's address, its cluster (reference) node's address, and finally its own address.

5.8.2 Two-Level Network

In [99] a two-level network has been considered. One level is formed by mobile land stations with limited coverage, and the second level (vertical network) is formed by a network of airplanes with both land-to-air and air-to-air interfaces. An airplane's land-to-air radio interface has a greater radio coverage and bandwidth and can serve to directly connect two land stations. The number of users inside an airplane radio coverage area defines the interface's footprint size. The concept of bandwidth pool is introduced to represent the bandwidth available to all the users sharing a common access medium (land-to-land interface or land-to-air interface). The cost of a link is defined to be inversely proportional to the remaining bandwidth in the bandwidth pool and directly proportional to the number of users accessing this pool, or in other words, the number of users that are deprived of accessing the bandwidth pool when a packet is transmitted. The routing protocol at each node has to decide which bandwidth pool to access when forwarding a particular packet.

In this scenario, the horizontal network (land mobiles) implementation of the SOAP protocol may enable a highly efficient method for exploiting the land-to-air interface. Assuming that the horizontal network has implemented the SOAP protocol, choosing reference nodes and forming clusters, a two level protocol may be implemented as follows:

When a source node has initially a packet to transmit (beginning of a session), it will check its location table for the destination node's reference node. The time the last entry (in the location table) was entered is also recovered. The next step is to assign a cost for the packet's use of each interface. For the land-to-air interface the cost is equal to the ratio between the footprint size of the air interface and the available Bandwidth in the bandwidth pool. For the land-to-land interface the packet is assigned a cost that is directly proportional to the time elapsed since the last update was entered. This cost will reflect the fact that older data may be out-of-date and may origin a long path toward the destination. The land-to-land interface cost is also proportional to the average number of neighbors a user has and inversely proportional to the network land-to-land interface available bandwidth. If no reference node is found the cost is set to infinity. This means that instead of bordercasting a REQUEST over the horizontal network, the requests are going to be carried out over the land-to-air interface. The resulting behavior in the network will be to minimize the number of bordercasting over the horizontal network. Also, long inefficient paths toward a destination are most likely eliminated, being replaced by broadcasts using the air network support (land-to-air and air-to-air interfaces). The horizontal network will tend to use only up-to-date paths, where the node is almost sure to be inside the recorded reference area.

After the beginning of the session and once the source node has up-to-date information about the destination node's reference area (if any), the source node may update the route's *accumulated cost*. For this purpose, each packet traveling over the horizontal network will have both an *accumulated cost* and a *remaining cost* field. The *accumulated cost* field will be initially set to zero, and each node that forwards the packet will add their link's cost computed as the ratio between the number of neighbors sharing the link over the available (remaining) link bandwidth. When the packet arrives at the destination, the *accumulated cost* field will have the current cost associated with the path followed over the horizontal network. This cost is path-dependent, so different packets following different path will collect different values of this cost. However, the observed *accumulated cost* is good enough to provide a rough estimate of the distance between the source and the destination nodes as well as the congestion state of the subset of the network between them.

When a node already in a session has a packet to send, it uses the most up-to-date *accumulated cost* information available as the land-to-land interface cost and compares it to the land-to-air interface cost to decide over which interface to forward the packet. When the land-to-land interface's cost is lower and as a consequence the packet is forwarded over the horizontal network, each intermediate node in the packet's path network will use the packet's *remaining cost* field as its land-to-land interface cost for this packet. The *remaining cost* field is initialized by the source node to the value of the last known *accumulated cost*, and each node along the path will decrement it by an amount equal to the cost of transmitting the packet over its links. It should be noted that the *remaining cost* may be negative or may reach the destination with a nonzero value. Nevertheless, it gives an idea of the approximate cost of the remainder of the route. Additionally, in networks with bidirectional links the *accumulated cost* of a source-destination path may be assumed to be equal to the value in the *accumulated cost* field of the packets received in the reverse direction (destination-source). For asymmetric networks, however, the *accumulated cost* in the forward direction has to be piggybacked from the destination back to the source.

Finally, the SOAP protocol with the mentioned modifications will allow for an efficient use of the land-air interface. The horizontal network will be relieved of broadcast, bordercast, and other congestion-causing packets. Packets that need to traverse long paths will be forwarded using the land-air interface whereas packets traveling short portions of the network will be transmitted over the horizontal network.

5.8.3 Integrated Fixed-Mobile Network

A mixed environment with picocells interconnected with fixed network nodes, and mobile nodes that may want to communicate with other mobiles as well as fixed network nodes is considered here. Picocells will have two interfaces. One will communicate them with the fixed network, and through it, with other picocells. Mobile nodes will also have two interfaces. One will communicate them with their closest picocell, and the other will serve them to communicate them directly to each other. The transmission range of this second interface will be smaller than the first, due to power and processing constraints of the nodes with respect to the picocells. In this scenario, the mobile nodes form an ad hoc network and may communicate with each other in a multi-hop fashion. The mobile nodes may rely on the picocells to forward their packets, or given that it is likely that mobile nodes are out of the picocell range, they may need to forward packets to each other in a multihop fashion all th away to the destination or at least up to the closest mobile node that can reach a picocell. Such an scenario may be possible in buildings having fixed antennas in predefined positions but that do not cover all the possibles areas; as for example halls, entrances, some rooms, etc.

In this scenario, it is expected that the SOAP protocol will identify the picocells as reference nodes and will keep track of them [5]. If the mobile nodes are always k hops or less away from a picocell then the picocells will be the only reference nodes. The picocells will always have routes to each other – after the initial broadcast announcing a picocell as reference node – so they will not require to execute the TORA protocol never again (will never get without a ‘downstream link’). In this scenario the protocol

[5] This behavior may even be forced by the picocells setting their *Gain* field to the maximum value and advertising themselves as new reference nodes

will present the required behavior as far as no mobile node is chosen as reference node.

But, if the mobile nodes are more than k hops away from a picocell it is likely that some of them will organize in a reference area. In such a case, a picocell would be forced to track down a mobile node using the TORA protocol resulting in eventual performance degradation as shown by the next example. Consider that a mobile node is chosen as reference node, over time it may be possible that the height level of all the mobile nodes with respect to a mobile reference node have been updated to the current time causing the mobile nodes' heights to be greater than the height of their closer picocell. In this situation a picocell would be forced to increase its own height and propagate the new height level all over the fixed network. This action is neither desired nor efficient.

The SOAP need to be revised and modified in this situations. The self-organizing algorithm discussed in Section 5.3 is still valid, but some alternative to TORA needs to be developed.

An ad-hoc solution may be to run a different algorithm in the picocells, which will behave as gateways as explained below. Routes between picocells and fixed network nodes will be computed in the traditional fashion, as for example, using link state updates (LSU). Mobile nodes will run the SOAP protocol described above. The picocells will run SOAP in their mobile network interface, and will keep track of the (mobile) reference nodes. If a picocell detects that a reference node is reachable [6] it may send an LSU update including a (virtual) link between the reference node and the picocell with a cost equal to the cost of the path between the picocell and the reference node[7]. Similarly, if the picocell detects a network partition that causes a reference node to be unreachable, it will send an LSU notifying the rest of the picocells that the link reference node-picocell is down. Also, when a picocell receives a broadcast from a new reference node, it will send an LSU update to the fixed network advertising a link to the new reference node [8]. Any picocell that receives such an LSU

[6] TORA detects if a node is unreachable (network partition) with one bit in the height field.

[7] Since the path between the picocell and the reference node is variable, so is their cost. The cost of the (virtual) reference node-picocell link should reflect the expected value of the path cost in the near future but not necessarily be directly proportional to it.

[8] Additionally, the picocell has to transmit, in the same or other packet, the location information of the nodes associated with the reference node. Both packets (LSU and location update) packets

will convert it into a broadcast similar to the one originated by the reference node (preserving the hop count)[9]. With the above modifications to SOAP, when a mobile node want to send a packet to a node that is unreachable (or too far away) in the mobile network, it would rely on the closest picocell which will use Link State (or other) protocol to forward the packet to the destination (if a fixed node), or to the closest picocell (if a mobile node). In the latter case, the picocell will forward the packet to the destination node's reference node as indicated by the SOAP protocol. In case the destination node is not associated with any reference node, then flooding will have to be used. The above protocol behavior assumes that location information is available. The actual implementation of the location management function depends on the choice between a hierarchical network (the picocells will storage the location management information for nodes that are not reachable if using only the mobile network segment) or flat network (the location management information for nodes unreachable unless using the fixed network segment, may be distributed across the network). The decision rules for such a choice need to be further investigated.

5.9 Conclusions

In this chapter SOAP, a protocol allowing a network to self-organize, has been introduced. This protocol, by employing a self-organizing algorithm together with a limited information dissemination and location management modules, efficiently extract the mobility and traffic patterns of the network. By exploiting these patterns, SOAP present excellent performance in the presence of strong pattern as for example is the case for group mobility (instead of random mobility) or when the set of destination is small and well defined (as opposed to be random, uniformly distributed). A complete protocol specification is presented, and some modifications to other scenarios are also covered. It should be noted that SOAP is a contribution in its own

should not leave the picocell network domain.

[9] Alternatively, the picocells may form a higher level hierarchy recording the reference node info without passing it to its neighboring mobile nodes. In such a hierarchical network, when a mobile source node needs to send a packet to a node that is not reachable or for which no location information is available, the source node will forward the packets the closest picocell, which will behave as its default gateway.

right.

Moreover, SOAP's self-organizing (SO) algorithm is a fundamental piece in the framework for a multi-mode routing protocol presented in Chapter 2. The work in this chapter provide us with an understanding of the complexity required in implementing such self-organizing algorithm, and such a multi-mode routing protocol. Indeed, SOAP may even be interpreted as as a simplified implementation of a multi-mode routing protocol built according to the proposed framework: the *limited information dissemination* module would consist of SOAP's link state propagation algorithm based on ZRP's IARP, that corresponds to the NSLS algorithm presented in Chapter 3, the *clustering*, *tracking*, and *location management* modules would constitute a simplified version of the *self-organizing* module, and finally, the higher level protocol decisions or either send the packet along a downstream link toward a reference node, use routes computed using the topology database (built with link state updates), or generate a route discovery based on ZRP's IERP's bordercast procedure, would constitute a reduced set of the decisions that a *multi-mode routing engine* module has to make. Thus, this chapter work proves the feasibility of implementing – at least simplified versions of – a multi-mode routing protocol.

However, in SOAP the gain and cost function determinations have been made in an ad hoc manner. Further research should focus on a theoretical framework that provides and understanding on the impact of the different choices of the SO algorithm parameters in the overall performance, allowing to identify the best set of parameters; as it was done in the previous chapters for the link state variants candidates to run in the *limited information dissemination* module. Later on, combining these results with the ones in Chapters 3 and 4, a multi-mode routing protocol specification may be developed.

Chapter 6

Conclusions

In the past routing protocols for wireless ad hoc networks have typically been designed with a particular, limited environment in mind. The results of this research have shown that due to the wide diversity of the conditions that may be encountered in an ad hoc network this approach is neither sufficient, nor is it necessary. Routing in environments subject to widely varying characteristics by engaging a single routing strategy is not effective.

This dissertation presents the first fully analytical approach to studying the effects of total-overhead, a new metric that captures the interrelated effects of routing overhead and path degradation. Hence, the analysis provides new insight into the actual tradeoffs involving routing overhead and path optimality for each of the classes of ad hoc network routing protocol. Results demonstrate that a novel protocol referred to as Hazy Sighted Link State provides a low cost alternative to complex hierarchical techniques, and provides optimal balance between overhead and routing optimality among the class of limited link-state algorithms.

The analysis provides the basis for a routing framework that is based on the concept of multi-mode routing. The idea is to apply the appropriate routing strategy, or “mode”, that is determined to be the most efficient under a given set of conditions. Thus, it adapts the strategy to both temporal and spatial network dynamics. The proposed framework presents a significant shift in current approaches to ad hoc routing that is shown to be effective and an improvement over the state-of-the-art in routing. The multi-mode approach includes techniques that allow for scalability and for extraction of and adaptation to the network mobility patterns.

In summary, prior to this research there was a lack of sufficient understanding of the network dependence on the value of the different protocol and system parameters. This shortcoming appeared to be due largely to a lack of theoretical understanding. However, such an understanding was necessary in order to develop the desired multi-mode framework. Hence, this dissertation focused on the study of these dependencies. The results of the research provide the theory and enabling mechanisms for the the development of a multi-mode routing protocol specification. Furthermore, analytical assessment of routing protocol asymptotic dependency on network parameters was introduced; based on this methodology the first theoretical results on the performance of the most representative routing protocols in the literature was derived; a new perspective on scalability was introduced; the first link state variant (HSLS) that is proven to be scalable was found; and the first protocol (SOAP) that extracts mobility and traffic patterns together was specified. Thus, the research presented in this dissertation makes several significant contributions that have advanced the current state-of-the-art on the topic of routing and ad hoc networks.

6.1 Future Work

The multi-mode framework represents an important step towards the development of practical ad hoc routing that is scalable and adaptive to a wide range of environments. However, since the SO and LLS algorithms were effectively studied as stand alone approaches, further work regarding their synergistic interaction is needed. Thus, a complete multi-mode routing protocol specification represents the next step in this work. The majority of the effort required involves quantifying the interactions of different parameters of the LLS and SO algorithms on overall system performance. In particular, the quantitative effect of group mobility on the gain and cost metrics involved in the SO algorithm, and the effect of the hierarchy imposed by the SO algorithm over the information dissemination technique of the LLS algorithm require further exploration.

Finally, the present work focuses in best-effort, min-hop routing. The objective of any wireless communication system is to eventually support the same suite of applications—including real-time, multimedia—that users can benefit from when tethered to a wired Internet. Thus, the entire scope of issues related to QoS constrained routing, coupled with adaptive application control and media access remain as exciting and important extensions to this work.

Appendix A

Approximate Expression for FSLS's Total Overhead

The *total overhead* induced by a tagged node running a generic FSLS algorithm under high mobility is composed of the proactive (control) overhead and the suboptimal routing overhead. These overhead types are analyzed separately in the next sections.

For both sources of overhead, it is assumed that the tagged node S is located in the center of a network of radius R, since for a large network each node may be viewed as being at the center ignoring boundary effects. This assumption allows for a tractable model, although the resulting expressions prove to be dependent on the particular value of R and in general, on the boundary conditions. However, the posterior analysis of the nature of the solution for $\{s_i\}$ suggests that the solution found is still valid for non-typical nodes (nodes not in the center of the network).

A.1 FSLS's Proactive Overhead

Consider Figure 3.3 and a highly mobile environment so that an LSU is generated every time interval. Let $s_1 \leq s_2 \leq s_3 \leq \ldots \leq s_n = R$. Thus, every $2^{n-1} * t_e$ seconds the algorithm is reset.

Consider grouping the LSU (re)transmission induced by S according to their initial TTL value (upon generation). Let type j LSUs ($j = 1, 2, \ldots, n$) be the LSUs transmitted and retransmitted as a consequence of a LSU that was generated by S with a TTL value set to s_j.

For some value $i < n$, LSUs with TTL set to s_i are generated by S at times $2^{i-1} * k * t_e$, where k is odd ($k = 1, 3, 5, \ldots$). These LSUs are generated every $2^i * t_e$ seconds and are retransmitted to $\Theta(s_i^2)$ nodes (assumption a.3). On the average, the expected number of retransmissions may be approximated by $c_i * s_i^2$, for some real c_i. Thus, the network forwards $c_i * s_i^2$ type i LSUs each $2^i * t_e$ seconds. In conclusion, node S induces a control overhead of $\frac{c_i * s_i^2}{2^i * t_e}$ type i LSUs per second for $i = 1, 2, \ldots, n-1$.

Regarding type n LSUs, they are generated at times $2^{n-1} * k * t_e$, with k being any integer ($k = 1, 2, 3\ldots$). Thus, type n LSUs are generated each $2^{n-1} * t_e$ seconds (the same as type $n-1$ LSUs), causing $\Theta(R^2)$ retransmissions. Thus, the network forwards $c_n * R^2$ type n LSUs each $2^{n-1} * t_e$ seconds. In conclusion, node S induces a control overhead of $\frac{c_n * R^2}{2^{n-1} * t_e}$ type n LSUs per second.

Adding together the types 1 to n LSUs, letting $size_{LSU}$ be the average size of a LSU packet, and assuming $c_i \approx c$, the proactive overhead (S_{pro}) induced by a node S running a generic FSLS algorithm is found to be:

$$\begin{aligned} S_{pro} &= \frac{size_{LSU}}{t_e} \left(\sum_{i=1}^{n-1} \frac{c_i s_i^2}{2^i} + \frac{c_n R^2}{2^{n-1}}\right) \text{ bps} \\ &\approx \frac{c * size_{LSU}}{t_e} \left(\sum_{i=1}^{n-1} \frac{s_i^2}{2^i} + \frac{R^2}{2^{n-1}}\right) \text{ bps} \end{aligned}$$

A.2 FSLS's Suboptimal Routing Overhead

In this section, a closed form expression for the suboptimal routing overhead induced by a tagged node S running the FSLS algorithm is derived.

Node S induces suboptimal routing overhead each time it forwards a data packet (whether it is the originating source or not) to a node that is not along the shortest path to the data packet's destination. For example, let's assume that node S receives a data packet destined to node D, which is r hops away. If node S forwards the packet to a node $r-1$ hops away from D then no suboptimal routing overhead will be induced. However, if node S forwards the packet to a node that is also r hops away from D, then a suboptimal routing overhead equivalent to the size of the data packet will be induced. In general, if node S forwards the packet to a node that is $r + j$ hops away from D [1], then a suboptimal routing overhead equivalent to $j + 1$ times the size of the data packet will be induced.

To analyze the suboptimal routing overhead induced by node S, consider Figure A.1 where a snapshot of the network, as seen by node S, is presented. It is assumed that all the nodes are within R hops from S. The network shown in Figure A.1 presents a great deal of uniformity but it is not intended to limit the analysis to just such a regular network. The regularity observed in Figure A.1 simply represents the *average* or *expected* values based on assumptions a.1 through a.4 (subsection 3.4.1). Thus, the network depicted in Figure A.1 reflects the expectation for average values for networks with the same density σ (i.e., $c_1 * r^2$ nodes at a distance of r hops or less from S, and $c_2 * r$ nodes exactly at a distance of r hops).

Let $x \rightarrow y$ be a flow from node x (the source) to node y; and let $\mathcal{F}_t(S)$ be the set of all such flows passing through node S at a given time t. Let $\lambda_f^{x \rightarrow y}$ be the average traffic going through flow $x \rightarrow y$. Then, by assumption a.5, $\lambda_f^{x \rightarrow y} = \frac{\lambda_t}{N-1} \approx \frac{\lambda_t}{N}$.

To compute the average suboptimal routing overhead induced by S at time t, consider that every time node S receives a packet destined to a node, say D, that is r hops away, it may or may not make a proper next hop decision. Let $p(S, x, y, t)$ be the probability that node S's next hop decision for flow $x \rightarrow y$ at time t is non-optimal. It is further assumed that if a non-optimal next hop decision is made, the packet is forwarded to a node that is at the same distance from D as node S, and that the flows are loop free. The first assumption is based on the observation that sending a data packet to a node that is $r + 1$ or more hops away from the destination will be equivalent to sending the packet in a direction totally opposite to the destination. While the above event is possible, it is highly unlikely during normal protocol operation (i.e., before the protocol breaks) for dense networks, especially when the destination is far away and geographical constraints dominate the routing decisions (i.e. the 'geographical' region discussed in subsection 3.4.1). Thus, ignoring these unlikely events will lead to a first order approximation on the network performance. The second assumption

[1] Values of j greater than 1 are only possible in the presence of unidirectional links

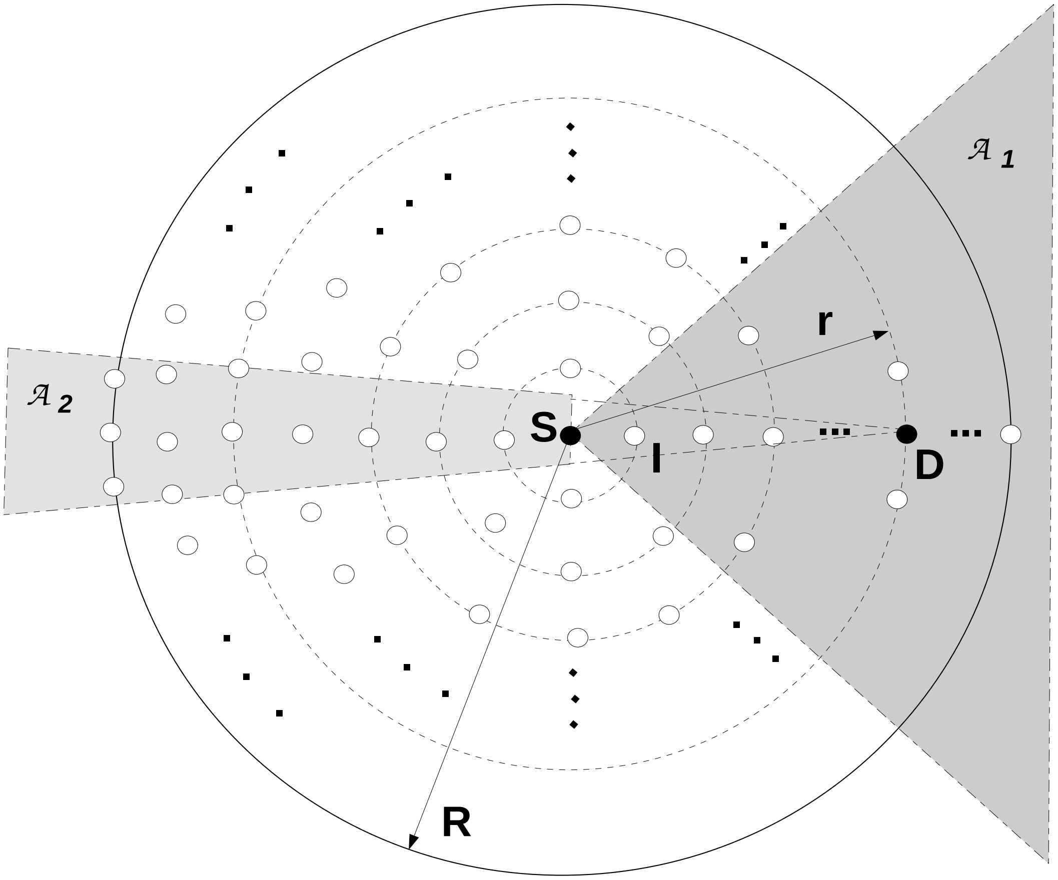


Figure A.1: A network snapshot, as seen by node S, running a FSLS algortihm.

is based on the same rare-event reasoning towards a first order approximation. Also, loop detection/removal mechanisms are relatively easy to implement. It should be noted, however, that loops are less rare when the packet is close to the destination, since in this region it is topology and not the underlying geography that dominates the routing decision. Thus, it is important that the selected FSLS strategy try to minimize these loops (this topic is further discussed in subsection 3.5.1).

As a consequence of the previous assumptions, a non-optimal next hop decision will induce exactly one extra hop transmission, and therefore the expected extra overhead induced by S when forwarding exactly one packet (from the $x \rightarrow y$ flow, at time t) is equal to $p(S,x,y,t)$. Thus, the expected suboptimal routing overhead (S_{sub}) induced by node S at time t is:

$$\begin{aligned} S_{sub}(t) &= \sum_{x \rightarrow y \in \mathcal{F}_t(S)} \lambda_f^{x \rightarrow y} p(S,x,y,t) \\ &\approx \frac{\lambda_t}{N} \sum_{x \rightarrow y \in \mathcal{F}_t(S)} p(S,x,y,t) \end{aligned}$$

Let $N_r(S,t)$ be the set of nodes that are exactly at a distance r in hops) from node S according to this node view of the network at time t, and let $\mathcal{F}_t(S,D)$ be the set of flows passing through S and ending at D (destination) at time t. Thus :

$$\begin{aligned} S_{sub}(t) &= \frac{\lambda_t}{N} \sum_{r=1}^{R} \sum_{D \in N_r(S,t)} \sum_{x \rightarrow D \in \mathcal{F}_t(S,D)} p(S,x,D,t) && \text{(A.1)} \\ &= \frac{\lambda_t}{N} \sum_{r=1}^{R} \sum_{D \in N_r(S,t)} \left|\mathcal{F}_t(S,D)\right| p(S,x,D,t) && \text{(A.2)} \end{aligned}$$

Where $|.|$ represents the cardinality (size) of a set, and the last equality comes from the observation that once a packet reaches node S to be forwarded to another node D, the packet's history is not relevant, and all such packets will be forwarded in the same manner regardless of the packet's origin (i.e., x).

To compute $p(S,x,D,t)$, consider Figure A.1 again. Node S will forward any data packet destined to node D toward node I, the next hop according to node S's view of the network. In Figure A.1 it can be seen that such a decision to forward the data packet through node I will be optimal if (and only if) node D is currently inside the area $\mathcal{A}_1$. However, since the figure represents node S's view, and this node (S) may not have received any update from D in the past $T(r)$ seconds (see section 3.3, on the maximum refreshing time), node D's current position may have changed.

Let t_1 be the most recent time at which node S received a LSU from node D. Let $p(S, x, D, t, t_1)$ be the probability of node S taking a non-optimal next hop decision when forwarding a packet to D through node I at time t, given that the last LSU update from D was received at time t_1. Since t_1 may be any time less than $T(r)$ seconds ago (in the interval $< t - T(r),\ t >$), [2] then:

$$p(S, x, D, t) \quad = \quad \frac{1}{T(r)} \int_{t-T(r)}^{t} p(S, x, D, t, t_1)\ dt_1 \tag{A.3}$$

$p(S, x, D, t, t_1)$ is equal to the probability of node D leaving the area $\mathcal{A}_1$ after time t_1 in Figure A.1 [3]. This probability depend on the node speed and the mobility model assumed. It is similar to the problem of residence time in cellular networks, where it is common practice to assume an exponential residence time. In our setting, $p(S, x, D, t, t_1)$ is the probability that the residence time inside the area $\mathcal{A}_1$ is less that $t - t_1$. Thus, assuming also an exponential model for the residence time [4], $p(S, x, D, t, t_1) \approx 1 - e^{-\frac{\mathcal{M}(t-t_1)}{r}}$. $\mathcal{M}$, in hops per second, is a constant that depends on the average relative node speed and on the geometry of the area $\mathcal{A}_1$ (which in turn depends on the degree of node S among other factors, but is independent of N or r);

[2] At time t_1, node D was r hops away from node S. If node D stays r or less hops away from S, it will 'refresh' node S at most after $T(r)$ seconds. Thus, in this case, if t were greater than $t_1 + T(r)$ node D would have 'refresh' node S about its new location, which would contradict the assumption that t_1 is the most recent time when node S received a LSU from node D. However, strictly speaking, node D may have moved more than r hops away from node S since time t_1. It will be assumed that node D is still *close* to r hops away, and that the above refreshing time is approximately correct. This assumption is further backed by the fact that function $T(r)$ remains constant under small variations of r (see Figure 3.4).

[3] Strictly speaking, area $\mathcal{A}_1$ is not necessarily a triangle. Figure A.1 just shows our expectation that on average that area will be a fraction, of roughly triangular shape, of the network. It is important to note that the area of $\mathcal{A}_1$ depends more heavily on the number of neighbors of node S than on the distance from D to S.

[4] Our results still hold as long as the cummulative density function (cdf) of the residence time ($F(t)$) is such that $\frac{\partial F(t)}{\partial t} \approx \frac{\mathcal{M}}{r}$, for small values of t, which is usually the case.

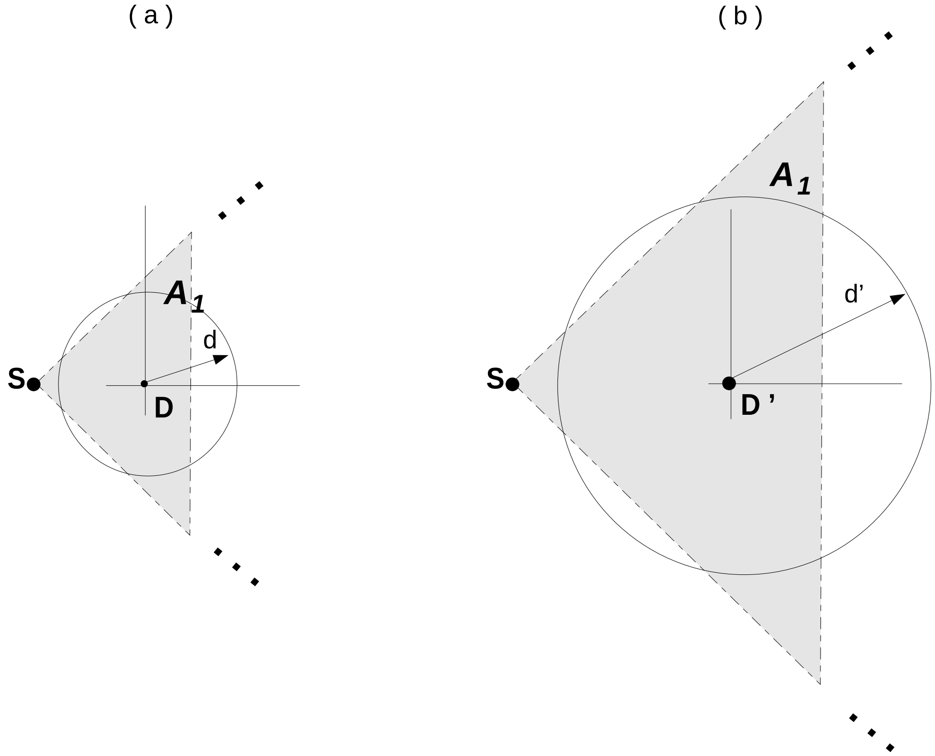


Figure A.2: Region $\mathcal{A}_1$ as seen by (a) node D, and (b) node D'.

$t - t_1$ is the elapsed time since the last LSU was received; r is the distance (at time t_1) in hops between S and D.

The dependence of $p(S, x, D, t, t_1)$ on the ratio $\frac{t-t_1}{r}$ follows from assumption a.8. Consider for example another node D' that is at a distance $2r$ hops from node S, in the same direction as D from S, so that node S also forwards a packet to D' through node I and an optimal next hop decision would be made as long as node D' remains inside the same area $\mathcal{A}_1$. Note however, that node D's view of the area $\mathcal{A}_1$ (regarding itself as the center) will differ from node D''s view. Basically, node D' will view area $\mathcal{A}_1$ (when it is at the center) as being a scaled version of node D's view, where the scale factor is equal to 2. Figure A.2 (a) and (b) show node D's and D''s view of the region $\mathcal{A}_1$ respectively.

If the elapsed time (since the last LSU was received from D') is also doubled, the maximum displacement ($maximum\_speed * elapsed\_time$) d' of node D' will also double with respect to node D's maximum displacement d, as also shown in Figure A.2. According to assumption a.8 about scaling with time, it would be expected that both probabilities of optimal and non-optimal next hop decisions (for nodes D and D') be the same. Thus, any simultaneous proportional increment on $t - t_1$ and r would cancel out. Strictly speaking, let p' and p be the probabilities of an optimal next hop decision for nodes D' and D respectively. Similarly, let $\triangle t'$ and $\triangle t$ be the elapsed time for nodes D' and D respectively. Thus:

$$p' = \int\int_{\mathcal{A}_1'} f_{t/t-\triangle t'}(x', y', 0, 0) dx'\, dy'$$

Where $f_{t/t-\triangle t'}$ is the probability distribution function of node D's position given that it was located at the origin at time $t - \triangle t'$. According to assumption a.8 about time scaling:

$$\begin{aligned} p' &= \int\int_{\mathcal{A}_1'} \frac{1}{(\triangle t')^2} f_{1/0}(\frac{x'}{\triangle t'}, \frac{y'}{\triangle t'}) dx'\, dy' \\ &= \int\int_{\mathcal{A}_1'} \frac{1}{4(\triangle t)^2} f_{1/0}(\frac{x'}{2\triangle t}, \frac{y'}{2\triangle t}) dx'\, dy' \end{aligned}$$

Where the last equality holds since $\triangle t' = 2\triangle t$. Then, applying the following variable transformation $x' = 2x$ and $y' = 2y$:

$$\begin{aligned} p' &= \int\int_{\mathcal{A}_1} \frac{1}{4(\triangle t)^2} f_{1/0}(\frac{2x}{2\triangle t}, \frac{2y}{2\triangle t})\, 4dx\, dy \\ &= \int\int_{\mathcal{A}_1} \frac{1}{(\triangle t)^2} f_{1/0}(\frac{x}{\triangle t}, \frac{y}{\triangle t})\, dx\, dy \\ &= p \end{aligned}$$

Which proves that $p' = p$ and therefore simultaneous increments on $t - t_1$ and r would cancel out.

Now, replacing the expression for $p(S, x, D, t, t_1)$ in equation A.3 the following is obtained:

$$p(S, x, D, t) = \frac{1}{T(r)} \int_{t-T(r)}^{t} (1 - e^{-\frac{\mathcal{M}(t-t_1)}{r}}) dt_1$$

$$
\begin{aligned}
&= 1 - \left[\frac{r\ e^{-\frac{\mathcal{M}(t-t_1)}{r}}}{\mathcal{M}T(r)} \right]_{t-T(r)}^{t} \\
&= 1 - \frac{r(1 - e^{-\frac{\mathcal{M}T(r)}{r}})}{\mathcal{M}T(r)} \\
&= \frac{1}{2}\frac{\mathcal{M}T(r)}{r} - \frac{1}{6}(\frac{\mathcal{M}T(r)}{r})^2 + \frac{1}{24}(\frac{\mathcal{M}T(r)}{r})^3 + \ldots
\end{aligned}
$$

where the last expression follows from the series expansion $e^{-x} = 1 - x + \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \ldots$. Thus, a first order approximation for $p(S,x,D,t)$ will be $p(S,x,D,t) \approx \frac{1}{2}\frac{\mathcal{M}T(r)}{r}$. Replacing this expression in equation A.2 the following is obtained:

$$
S_{sub} = \frac{\lambda_t}{N}\sum_{r=1}^{R}\sum_{D \in N_r(S,t)} |\mathcal{F}_t(S,D)| \frac{1}{2}\frac{\mathcal{M}T(r)}{r} \qquad \text{(A.4)}
$$

To compute $|\mathcal{F}_t(S,D)|$ consider that if all the nodes to the left of node S in Figure A.1 have the same information as node S, the set $\mathcal{F}_t(S,D)$ would be formed by the nodes inside the area $\mathcal{A}_2$. However, nodes to the left of node S may not have been updated at time t_1 (last time node S was updated), and therefore they may have a different view of the network. Thus, the actual set of nodes that forward packets to D through node S (and therefore I), may be different from the set of nodes inside area $\mathcal{A}_2$. Nevertheless, the *number* of such nodes must be roughly similar. The last statement follows from the observation that if the network in Figure A.1 is vertically split (passing through node S), all the nodes on the left half will have to go through some of the nodes on the boundary (node S or some other node along the vertical axis). Let $x_1, x_2, \ldots$ be the nodes on the vertical axis, and let $\mathcal{F}_t(x_1,D), \mathcal{F}_t(x_2,D), \ldots$ be the sets formed by the flows reaching node D through node $x_1, x_2, \ldots$ respectively. Thus, each node in the left half is the source to exactly one flow belonging to exactly one of the aforementioned sets. Thus, the problem of assigning a node (flow) to one set arises. If all the nodes in the left half of the network in Figure A.1 have the same view of the network as node S, the number of nodes assigned to $\mathcal{F}_t(S,D)$ is equal to $\sigma * area(\mathcal{A}_2)$. Since node D has been moving, nodes on the left half will have a different (older) view about node D, biasing the assignment. However, since node D's mobility is random, different possibilities cancel out the bias and at the end the proportionality in the assignments is maintained. Thus, the number of nodes assigned to $\mathcal{F}_t(S,D)$ will still be close to $\sigma * area(\mathcal{A}_2)$.

Approximating the area $\mathcal{A}_2$ by the trapezoid inscribed on the circle of radius R, and computing this area as the difference between the two similar triangles with a vertex at D (see Figure A.1) the following expression is derived: $area(\mathcal{A}_2) \approx \frac{\alpha}{2} * r * \ell * \left[(\frac{R+r}{r})^2 - 1\right]$, where α is the base of the smaller of the similar triangles, and is numerically equal to the distance between node S and one of its *close* neighbors in Figure A.1. Thus, α is more or less independent from r and R. ℓ is the average distance between a node and its neighbors in a min-hop path, so that nodes that are r hops away will be – on average – at a distance $r * \ell$. [5] Further simplifying the previous expression and inserting it in equation A.4, the following is obtained:

$$S_{sub} = \frac{\lambda_t}{N}\sum_{r=1}^{R}\sum_{D\in N_r(S,t)}\frac{1}{4}\alpha\sigma\ell\mathcal{M}R^2(1+\frac{2r}{R})\frac{T(r)}{r^2} \quad (A.5)$$

$$= \frac{\lambda_t}{N}\sum_{r=1}^{R}|N_r(S,t)|\frac{1}{4}\alpha\sigma\ell\mathcal{M}R^2(1+\frac{2r}{R})\frac{T(r)}{r^2} \quad (A.6)$$

where the last equality holds since the expression inside the inner summation depends only on r and not on the particular node D. Finally, from assumption a.3, the number of nodes seen by node S at exactly a distance of r hops will be $|N_r(S,t)| \approx \beta * r$, for some β real. Thus:

$$S_{sub} = \frac{\lambda_t}{N}\frac{\alpha\beta\sigma\ell}{4}\mathcal{M}R^2\sum_{r=1}^{R}(1+\frac{2r}{R})\frac{T(r)}{r} \quad (A.7)$$

Since the factor $(1+\frac{2r}{R})$ changes slowly and it is bounded between $1 < 1+\frac{2r}{R} \leq 3$, S_{sub} may be approximated by:

$$S_{sub} \approx \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma\ell}{4}\mathcal{M}R^2\sum_{r=1}^{R}\frac{T(r)}{r} \quad (A.8)$$

[5] It may appear that α and ℓ are the same quantity, but they differ in that α is the distance to the closest nodes and as such it is defined by the geographical node positioning. ℓ in the other hand, is the average distance between nodes in a min-hop path (longest links) and therefore it is close to the maximum transmission range, which depends on the nodes' transmission power. If the transmission power is large, ℓ may be significantly greater than α.

$$= \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma\ell}{4}\mathcal{M}R^2\sum_{i=1}^{n}\sum_{r=s_{i-1}+1}^{s_i}\frac{2^{i-1}}{r}t_e \quad \text{(A.9)}$$

$$= \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma\ell}{4}\mathcal{M}R^2 t_e\sum_{i=1}^{n}2^{i-1}f_i \quad \text{(A.10)}$$

where $1 < \gamma \leq 3$, and the last 2 equilities follow since $T(r) = 2^{i-1}t_e$ for $s_{i-1}+1 \leq r \leq s_i$ (see Figure 3.4) [6] and $f_i = \sum_{r=s_{i-1}+1}^{s_i}(\frac{1}{r})$. Finally, approximating f_i by using $f_i \approx \int_{s_{i-1}}^{s_i}\frac{dr}{r} = ln(\frac{s_i}{s_{i-1}})$ for $i \geq 2$, and $f_1 \approx ln(s_1)$, and recalling that $s_n = R$, the following expression is obtained [7]:

$$S_{sub} \approx \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma\ell}{4}\mathcal{M}R^2 t_e[ln(s_1) + \sum_{i=2}^{n}2^{i-1}ln(\frac{s_i}{s_{i-1}})] \quad \text{(A.11)}$$

$$= \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma\ell}{4}\mathcal{M}R^2 t_e[2^{n-1}ln(R) - \sum_{i=1}^{n-1}2^{i-1}ln(s_i)] \quad \text{(A.12)}$$

A.3 HSLS's Total Overhead

FSLS does not induce reactive overhead, thus taking into account the *proactive* and *suboptimal routing* overheads, FSLS total overhead (S_{total}) induced by a node S is found to be:

$$S_{total} = S_{pro} + S_{sub}$$

$$= \frac{c * size_{LSU}}{t_e}(\sum_{i=1}^{n-1}\frac{s_i^2}{2^i} + \frac{R^2}{2^{n-1}}) + \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma\ell}{4}\mathcal{M}R^2 t_e[2^{n-1}ln(R) - \sum_{i=1}^{n-1}2^{i-1}ln(s_i)]$$

[6] $s_0 = 0$ for consistency.

[7] The first term's (f_1's) approximation is a loose one, but this has little effect in the overall result since the weight of this term in the S_{sub} expression is relatively small (2^1). The accuracy of S_{sub} approximation mainly depends on the accuracy of the approximation of the highest weight terms (f_n and the like). It can be noted that for large values of i (and s_i), the approximation employed is tight.

Appendix B

Control Overhead Induced by a Node Running A-HSLS

In this appendix the control overhead induced by a node, say X, running the A-HSLS algorithm described in subsection 3.5.4 and Figures 3.11 and 3.12 is derived. This derivation is based on the assumption that the time between link failures experienced by a node is a exponentially distributed random variable with mean value $\frac{1}{\lambda_{lc}}$.

Let t_i^{glo} be the time at which node X sends the i-th global LSU. (t_1^{glo} is the time the algorithm was initialized). Let I_{i+} be the i-th interval between global LSUs including the time the (i+1)-th LSU was sent, i.e. $I_{i+} =< t_i^{glo}, t_{i+1}^{glo}]$ (it is important to note that I_{i+} includes the time point t_{i+1}^{glo}).

Let's define the interval type as either SLS or HSLS. The i-th interval is said to be of type SLS if the (i+1)-th LSU was sent by node X while in SLS mode. Similarly, the i-th interval is said to be of type SLS if the (i+1)-th LSU was sent by node X while in HSLS mode.

The number of LSU (re)transmissions induced by node X during the i-th interval, provided that its type is SLS, is N since at time t_{i+1}^{glo} each node will have to retransmit node X's LSU once. In the other hand, if the i-th interval was of type HSLS, node X would have sent several LSUs with TTL equal to 2, 4, 8, and so for up to $2R_x$. The (last) LSU with TTL equal to $2R_x$ will require N retransmissions (the entire network). The LSU with TTL equal to R_x will only require f_xN retransmissions since it will not reach the entire network. From assumption a.3, f_x should be between 0.25 (when R_x is one half of the network radius) and 1 (when R_x is almost equal to the network radius, as seen from node X). In practice, due to boundary conditions typical values of f_x will be in the interval $< 0.5, 1 >$. For LSUs with TTL equal to $R_x/2$ boundary conditions may be ignored and the number of LSU transmisions will be roughly $\frac{f_x}{4}N$. Recalling that there may be up to 2 LSUs with TTL equal to $R_x/2$, these LSUs contribute with with $\frac{f_x}{2}N$ LSUs transmissions to the control overhead. For smaller TTL values, it should be noted that reducing the TTL by a factor of 2 will reduce the number of (re)transmissions due to each LSU by a factor of 4, but at the same time, the number of such LSUs will double with the effect of reducing by one half their aggregated contribution to the control overhead. From the above, the number of LSU transmission indiced by node X during an interval of type HSLS will be $N + f_xN + \frac{f_x}{2}N + \frac{f_x}{4} + \cdots = N[1 + f_x(1 + 0.5 + 0.25 + \cdots)] \approx N(1 + 2f_x)$. The above expression correspond to the case where LSUs are sent at every time interval. In practice, LSUs will not necessarily be sent each t_e period. However, the LSUs with larger TTL value, i.e. the main contributors to the control overhead, will be sent by sure. Thus, using the above expression for all the HSLS intervals provides a good approximation, where the error in the results is on the conservative side. It should be noted that the above simplification is done for simplicity sake, because given the exponential distribution of the inter link change times it is perfectly feasible to compute a close expression for the expected number of LSU's transmission required during a HSLS interval. However, the solution so found will not differ much from the one obtained here, which is easier to understand. Finally, Let p_i be the probability that the i-th interval was a HSLS interval, then the expected number of LSU transmissions over the i-th interval is equal to $(1 - p_i)N + p_i(1 + 2f_x)N$.

Regarding the expected length of the i-th interval, let t_i^{elap} be the time elapsed since t_i^{glo} (the time the i-th global LSU was sent) and the time the next link change is detected. Given the memoryless property of the time between link change (exponential distribution), t_i^{elap} will also be exponentially distributed with mean $\frac{1}{\lambda_{lc}}$. If the i-th interval is of SLS type, the interval length will be equal to t_i^{elap}. If the i-th interval is of type HSLS, the interval duration will be equal to t_i^{elap} plus a fixed time equal to $(R_x - 1)t_e$, since $R_x - 1$ time intervals should elapse before *NumEventInt* reaches the value R_x and the algorithm is reinitiated. For simplicity, the above time is rounded and therefore the interval length is approximated by $t_i^{elap} + R_x t_e$. [1] Then, the expected lenght of the i-th interval is equal to $\frac{1}{\lambda_{lc}} + p_i R_x t_e$, that is, it is equal to the expected elapsed time t_i^{elap} plus the term $R_x t_e$ weighted by the probability that the interval was of type HSLS.

Combining the previous results, averaging the expected number of LSU transmissions per interval over the expected interval lenght, A-HSLS control overhead is found to be equal to :

$$\begin{aligned} X_{A-HSLS}^{control} &= \frac{E\{\text{No. of LSU transmission per interval}\}}{E\{\text{interval lenght}\}} \\ &= \frac{(1-p_i) + p_i(1+2f_x)}{p_i R_x t_e + \frac{1}{\lambda_{lc}}} N \\ &= \frac{1 + 2p_i f_x}{p_i R_x t_e + \frac{1}{\lambda_{lc}}} N \qquad \text{(B.1)} \end{aligned}$$

Finally, p_i (the probability that the i-th interval be of type HSLS) is equal to the probability that t_i^{elap} be smaller than $\frac{R_x t_e}{2}$. Since the time between link changes is exponentially distributed (memoryless), the elapsed time will also be exponentially distributed with the same mean value ($\frac{1}{\lambda_{lc}}$). Thus, t_i^{elap} *probability density function (pdf)* is $f(t) = \lambda_{lc} e^{-\lambda_{lc} t}$. Then, p_i is obtained as follows:

$$\begin{aligned} p_i &= P\{t_i^{elap} \leq \frac{R_x t_e}{2}\} \\ &= \int_0^{\frac{R_x t_e}{2}} \lambda_{lc}\, e^{-\lambda_{lc} t}\, dt \\ &= 1 - e^{-\frac{\lambda_{lc} R_x t_e}{2}} \qquad \text{(B.2)} \end{aligned}$$

Thus, by applying equation B.2 in equation B.1, the control overhead induced by a node X, running the A-HSLS algorithm ($X_{A-HSLS}^{control}$) is obtained.

[1] It may be noted that for the special case where t_i^{elap} is lower than t_e the correct expression for the interval duration is $R_x t_e$. However, since the percentual difference is small (less than $\frac{1}{R_x}$) the previous approximation is used for simplicity sake. Once again, a more detailed analysis is possible, but it will unnecessarily complicate the resulting expressions, hiding the insight we expect to gain.

Appendix C

Asymptotic Expression for ZRP's Total Overhead

The Zone Routing Protocol (ZRP) is a hybrid approach, combining a proactive and a reactive part. ZRP attempts to minimize the sum of the proactive and reactive overhead.

In ZRP, a node propagates event-driven (Link State) updates to its k-hop neighbors (nodes at a distance, in hops, of k or less). Thus, each node has full knowledge of its k-hop neighborhood and may forward packets to any node on it. When a node needs to forward a packet outside its k-hop neighborhood, its sends a route request message, but this packet must only be sent to a small subset subset of nodes, namely, 'border nodes'. The nodes in this subset have enough information about their k-hop neighborhoods as to decide whether to reply to the route request or to forward it to its own set of 'border' nodes. The route formed will be described in terms of the 'border' nodes only, thus allowing 'border' nodes to locally recover from individual link failures, reducing the route maintenance cost. The algorithm responsible for 'bordercasting', i.e. sending the route request messages (query message in ZRP's terminology) to the border nodes is not trivial. If effective and efficient algorithms as the ones presented in [112] are not in place, the process of disseminating teh query messages to the border nodes may even result in more control overhead than just flooding the network, eliminating the advantage of keeping proactive information.

ZRP's *total overhead* components will be analyzed in the next subsections, where lower bounds will be derived.

C.1 ZRP Proactive Overhead

After a node S detects a link status change, it generates and propagates an LSU with a Time To Live (TTL) field set to k. Thus, the LSU is retransmitted for all the nodes that are $k-1$ or less hops away from S. In average there are $\Theta((k-1)^2) = \Theta(k^2)$ such nodes (assumption a.3). Thus, in ZRP each LSU induces $\Theta(k^2)$ transmissions. Since the LSU generation rate (for the whole network) is $\lambda_{lc} * N$, then we conclude that ZRP *proactive* overhead per second is $\Theta(k^2 * \lambda_{lc} * N)$.

Similarly, if we let n_k represent the average number of nodes inside a node (say S) 'zone' (i.e. less that k hops away from S), and recalling (assumption a.3) that $n_k = \Theta(k^2)$, then we obtain that ZRP *proactive* overhead per second is $\Theta(n_k * \lambda_{lc} * N)$.

C.2 ZRP Reactive Overhead

For ZRP's *reactive* overhead, a lower bound will be provided. This lower bound will be obtained by considering only the bandwidth consumed by each new session's route request (RREQ). The bandwidth consumed for new RREQ required for repairing paths (which could be significant in a highly mobile environment) will be ignored.

To compute the bandwidth consumed by new sessions' route requests (RREQs), we first consider the bandwidth consumed by one (1) route request (RREQ) generated by a node (say S) due to the beginning of a new session.

Let $\{B_i^S\}$ be the set of border nodes that will need to be paged by node S. ZRP tries to efficiently minimize the number of nodes paged. Thus, considering that each border node 'covers' on average $n_k = \Theta(k^2)$ nodes (assumption a.3) and that each node is 'covered' by at least onea border node, it is concluded that adding up the number of nodes covered by any of the border nodes or the source will result in a number greater than N (i.e. the entire network is 'covered' when trying to find a destination). As a consequence $|\{B_i^S\}| + 1) * n_k \geq N$. The last inequality implies

$|\{B_i^S\}| \geq \frac{N}{n_k} - 1$. Thus, the number of border nodes is $\Omega(N/n_k)$. The more effective the implementation of ZRP, the tighter the bound.

Each border node has to receive at least once the RREQ message originated by node S. For example, assume that border node B_i^S received the RREQ for the first time from border node B_j^S. This RREQ packet has to travel k hops from B_j^S to B_i^S and therefore it consumes $size\_of\_RREQ * k$ bits. Adding up all such transmissions over all border nodes a bandwidth consumption of $size\_of\_RREQ * k * |\{B_i^S\}|$ bits is obtained. The last quantity is a lower bound on the bandwidth required for the propagation of a node's (S) RREQ message, since it does not consider duplicate transmissions and back-propagation of RREQ messages, that although minimized by ZRP efficient query control schemes [112] process, they can not be totally eliminated. In general, the more efficient the query control schemes the tighter the bound.

Since $\lambda_s * N$ new RREQ are generated each second, the ZRP *reactive* overhead per second is lower bounded by $\lambda_s * N * size\_of\_RREQ * k * |\{B_i^S\}| \geq \lambda_s * N * size\_of\_RREQ * k * (\frac{N}{n_k} - 1)$. Finally, the ZRP *reactive* overhead per second is $\Omega(\lambda_s N^2/k) = \Omega(\lambda_s N^2/\sqrt{n_k})$.

C.3 ZRP Suboptimal Routing Overhead

ZRP tries to maintain the same path (in terms of border nodes) to a destination as long as it is possible, regardless of route optimality. As a consequence, as time evolves, ZRP's paths will degrade. Thus, ZRP's *suboptimal routing* overhead increases with mobility and session duration, and decreases with the 'zone radius'. In one extreme case in which one node's zone is the entire network, then the induced suboptimal overhead is zero; in the other extreme case in which the zone radius is the smallest possible the DSR's performance is obtained.

An upper bound on ZRP's suboptimal routing overjead may be obtained by considering that the worst possible path 's length for a ZRP session grows as $\Theta(N/k)$. To visualize this, consider a sequence of border nodes from a source (S) to a destination (D) long after the initial path was constructed and after several repair procedures have taken place : $S - B_1^S - B_2^S - B_3^S - B_4^S - ...D$. One property of this sequence is that 2 non-adjacent members of the list can not belong to each other's zone (i.e. can not be less that k hops apart). [1] This property causes (as it will be explained below) the largest possible sequence to increase as $\Theta(N/k^2)$. Also the number of transmissions required to forward one packet from one node of the sequence to the next is on average $\Theta(k)$. [2] Then, the maximum bandwidth that may be required to forward one packet (a long time after the session was first created, and assuming high mobility) is $\Theta(\frac{N}{k^2}) * \Theta(k) = \Theta(\frac{N}{k})$ bits. Since $\lambda_t * N$ packets are generated each second, an upper bound for the suboptimal routing overhead per second for ZRP is given by the difference between this maximum bandwidth employed $(\Theta(\frac{N}{k} * \lambda_t * N))$ and the optimal value $(\Theta(\lambda_t * N * L) = \Theta(\lambda_t * N^{1.5}))$ that would have been obtained if full topology information were available. Thus, the ZRP suboptimal routing overhead per second is $O(\lambda_t * \frac{N^2}{k})$ (upper bound).

Note that the upper bound for suboptimal routing overhead represents the same asymptotic behavior as the lower bound for the ZRP *reactive* overhead (considering that λ_t and λ_s are directly proportional, where the proportionality constant is the average number of packets transmitted as part of a session). Thus, it can be stated without loss of generality that the asymptotic behavior of ZRP is captured by the reactive and proactive overhead alone, and that the analysis of *total overhead* asymptotic behavior can be characterized fully by these values. Thus, it is not necessary to further improve the loose upper bound on suboptimal routing overhead already derived.

Next, it needs to be demonstrated that since the sequence of intermediate nodes from source S to destination D has the property that non-adjacent nodes are more than k hops apart, then the maximum possible length of such a sequence increases as $\Theta(N/k^2)$. Indeed, consider only the sub-sequence formed by the odd-placed nodes in

[1] For example, if B_1^S and B_4^S are less than k hops apart, then B_1^S will shorten the sequence from S to D as follows : $S - B_1^S - B_4^S - ...D$. Thus, by repeating the above procedure we always get a sequence with the aforementioned property.

[2] We know that the maximum distance between consecutive nodes in the sequence is k (one is in the zone of the other) and the minimum distance between nodes two position apart (e.g. B_1^S and B_3^S) in the sequence is at least $k + 1$ (since they do not belong to each other zone because of the aforementioned property). Then, the average number of transmission required is between $\frac{k+1}{2}$ and k (i.e. $\Theta(k)$).

the original sequence (i.e. S, B_2^S, B_4^S, B_6^S ...). The presence of S in the subsequence inhibits (i.e. prevents from being in the above sequence of odd-placed nodes, as discussed before) any other node in S's zone. Similarly, the presence of B_2^S in the sub-sequence inhibits all other nodes in B_2^S's zone. Let r_j be the maximum number of times a node j can be inhibited (with amean value r over all nodes); it will be shown later that r increases as $\Theta(1)$ with respect to network size (N) and zone radius (k). Adding up the number of nodes inhibited by all the nodes in the subsequence, the resulting value can not be larger than $N * r$. Thus, the length of the subsequence is smaller than $N * r/n_k$, and the length of the original sequence is smaller than $2 * N * r/n_k$. Thus, the maximum sequence length is $\Theta(N/k^2)$ (provided r is $\Theta(1)$).

Finally, to visualize the behavior of r (bounded, independently from N and k), consider a node X and the set of nodes $\{Y_i\}$ that inhibits node X from belonging to the aforementioned sub-sequence. As mentioned earlier nodes $\{Y_i\}$ are at least k hops away from each other. They also must be k or less hops away from X (to be able to 'inhibit' it). Thus, the limit in the number of times node X can be inhibited is equal to the maximum number of inhibitors inside node X's zone. In other words, r is the maximum number of nodes that can be inside node X's zone (less than k hops away), given that they are all more than k hops away from each other. Before attempting to answer this question in a graph theoretic framework, it is possible to formulate a similar (in view of assumptions a.2, a.3, and a.4) geometric problem: what is the maximum number of points that can be placed inside a circle of radius R, such that the minimum distance between any of this points is greater than R?.

If the condition would have been "greater or equal" to R, it is easily verified that the solution would be the 7 points shown in Figure C.1: P_1, the center of the circle, and the 6 vertices ($P_2 \ldots P_7$) of a regular hexagon with its side lengths equal to R and its center coinciding with the circle's center (i.e. P_1). Thus the six vertices will be exactly on the border of the circle of radius R. Any repositioning of the points trying to give room for another one will result in the reduction of some of the distances to less than R. Thus, if 7 is the maximum that can be achieved provided that the distances be "greater or equal" to R, then when the distances are restricted to be only "greater", 7 is no longer achievable and the maximum is at most 6. Two important

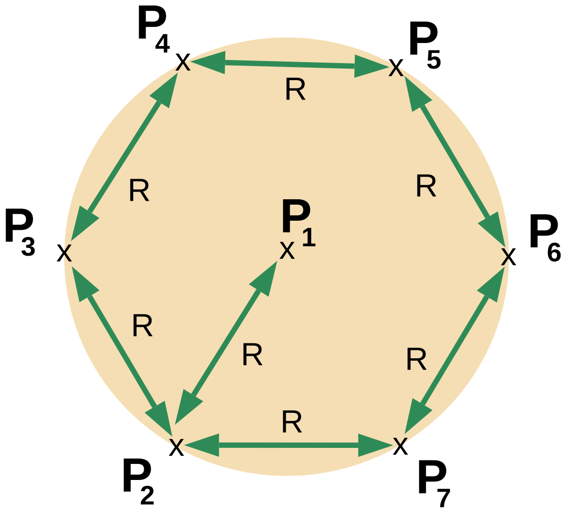


Figure C.1: Geometrical interpretation of the maximum number of 'inhibitors' that 'cover' a node

observations are that this value (6 or 7) does not depend on the circle radius and that the maximum number of points is achieved when most of the points try to be on the border of the circle (that is, trying to expand the distances as much as possible).

Next, we utilize the insight gained in the geometrical interpretation to justify that r is bounded independently of N and k. It is known that r is the (average) maximum number of nodes that can be inside node X's zone (less than k hops away) given that they are all more than k hops away from each other. From the geometric insight we know that on average the number of nodes may be maximized by putting the nodes on the boundaries of node X's zone. Assume that similarly to the geometric case, X is chosen as its own inhibitor (without loss of generality) and also the $r-1$ remaining nodes are chosen to belong to the border of node X's zone. Let $\mathcal{C}$ be the shortest loop (formed only by nodes inside X's zone) containing all these $r-1$ nodes. The length of this cycle will increase as $\Theta(\sqrt{n_k}) = \Theta(k)$, because assumption a.4 implies that the one-dimensional metrics (average path, maximum path, and consequently also the set 'perimeter') of a set of nodes are directly proportional to the square of the number of nodes in the set. Finally, since the loop length is $\Theta(k)$, and the distance (length) between two consecutive 'inhibitor' nodes in the loop is greater than or equal to $k+1$; then the maximum number of 'inhibitor' nodes is equal to the cycle length divided by $k+1$, which is $\Theta(k)/(k+1) = \Theta(1)$. Thus, r is bounded, and this bound is $\Theta(1)$, independent of k and N.[3]

C.4 ZRP Total Overhead

The previous subsection derived a lower bound for ZRP's reactive and proactive overheads. Also, an upper bound for ZRP's suboptimal routing overhead showed that inclusion of this term is not necessary to analyze the asymptotic properties on ZRP's total overhead. Thus, ZRP total overhead is $\Omega(n_k * \lambda_{lc} * N + \lambda_s N^2/\sqrt{n_k})$, which is obtained by adding the *reactive* and *proactive* overheads. Minimizing this lower bound by properly choosing the value n_k, shows that the best asymptotic behavior of the bound is obtained when (if possible) $n_k = \Theta((\frac{\lambda_s * N}{\lambda_{lc}})^{\frac{2}{3}})$, obtaining a *total overhead* that is $\Omega(\lambda_{lc}^{\frac{1}{3}} * \lambda_s^{\frac{2}{3}} * N^{\frac{5}{3}})$. [4]

When $\lambda_{lc} = \Theta(\lambda_s * N)$, n_k must be $\Theta(1)$ and therefore the *total overhead* induced by ZRP becomes $\Omega(\lambda_{lc} * N + \lambda_s N^2) = \Omega(N * (\lambda_{lc} + \lambda_s * N)) = \Omega(\lambda_s * N^2)$. If λ_{lc} grows faster that $\Theta(\lambda_s * N)$, values of n_k lower than 1 do not make sense. What happens is that ZRP behaves in a pure reactive mode (as DSR) and therefore the *total overhead* induced by ZRP in those cases is also $\Omega(\lambda_s * N^2)$.

On the other hand, if $\lambda_{lc} = \Theta(\lambda_s/\sqrt{N})$, the best achievable value of n_k is $\Theta(N)$. Thus, ZRP's *total overhead* becomes $\Omega(\lambda_{lc} * N^2 + \lambda_s N^{1.5}) = \Omega(N^2 * (\lambda_{lc} + \lambda_s/\sqrt{N})) = \Omega(\lambda_{lc} * N^2)$. If $\lambda_s/\sqrt{N}$ grows faster than λ_{lc}, then n_k can not grow more than N and therefore ZRP behaves in a pure proactive mode (as SLS) and induces a total overhead of $\Omega(\lambda_{lc} * N^2)$.

[3] This paragraph does not intend to be a mathematical proof, but only an intuitive explanation of the reasoning for considering r to be $\Theta(1)$.

[4] Note that in this case k, the zone radius, is $\Theta((\frac{\lambda_s * N}{\lambda_{lc}})^{\frac{1}{3}})$. Thus, k should increase with traffic and decrease with mobility as expected; but the dependency is not linear.

Finally, ZRP's total overhead is:

$$ZRP_{total\_overhead} = \begin{cases} \Omega(\lambda_{lc} * N^2) & \text{if } \lambda_{lc} = O(\lambda_s/\sqrt{N}) \\ \Omega(\lambda_{lc}^{\frac{1}{3}} * \lambda_s^{\frac{2}{3}} * N^{\frac{5}{3}}) & \text{if } \lambda_{lc} = \Omega(\lambda_s/\sqrt{N}) \text{ and } \lambda_{lc} = O(\lambda_s * N) \\ \Omega(\lambda_s * N^2) & \text{if } \lambda_{lc} = \Omega(\lambda_s * N) \end{cases}$$

Appendix D

Asymptotic Expression for HSLS's Suboptimal Routing Overhead

The key to the derivation of the HSLS *suboptimal routing* overhead is to understand that if the probability of making a non-optimal next hop decision at any time is independent of the network size N and traffic λ_t, then the *suboptimal routing* overhead increases with respect to traffic and size as $\Theta(\lambda_t * N^{1.5})$. [1]. To visualize this, consider a simple network model where each node's probability of making a non-optimal next hop decision, independently of the distance k to the destination, [2] is p. Moreover, consider the following scenario (shown in Figure D.1) where a node S must forward a packet towards a destination D (in general, S is not the source of the packet but it is relaying it).

Let k be the minimum distance (in hops) from S to D. There are several paths that achieve this distance, and the set of nodes forming part of these paths are enclosed in the small region $\mathcal{B}$ including I_1, I_2, and I_3. Thus an optimal next hop decision will be made if the packet is delivered to any of these three nodes (I_1, I_2, or I_3). Clearly, if the packet is delivered to any of these nodes, the distance from the new

[1] Note that the *suboptimal routing* overhead also depends on the mobility and/or rate of topological change so the above expression is not complete. Later, a more precise expression will be derived

[2] Later in this subsection we will show that for HSLS, the probability of a non-optimal next hop decision roughly remains constant (or at least lower- and upper- bounded) with respect to the distance to the destination.

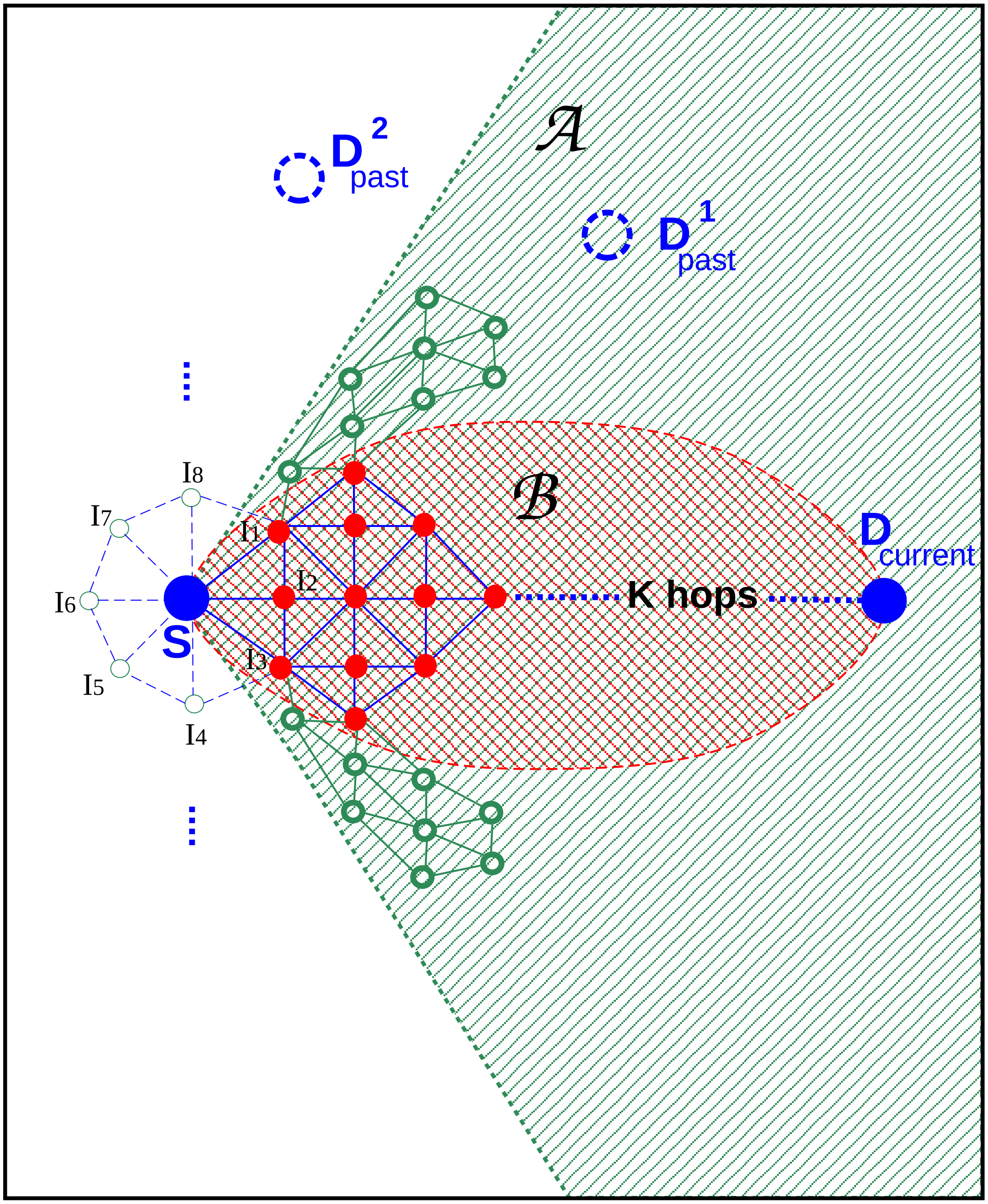


Figure D.1: Optimal and non-optimal next hop decision in the HSLS protocol

location of the packet (i.e. I_1, or I_2, or I_3) to D will decrease by one to $k-1$. If a non-optimal next hop decision is made (packet delivered to $I_4, \ldots, I_8$, the distance from the new location of the packet to D will be k (remain the same) or increase to $k+1$ (can not be more since there is at least one path of $k+1$ hops from any I_j to D, i.e. the path $I_j - S - minimum\_path_{S\_to\_D} - D$). It can be intuitively seen that the probability that a next hop error actually produces an increase in the distance to the destination is very low in a network, unless the density is very small (sparse network). We can then simplify our model and assume that a packet can be successfully delivered to a proper next hop node (thus reducing the distance to the destination by 1) with probability $1-p$, and can be delivered to a non-optimal next hop leaving the packet's distance to the destination unaltered with probability p. If we further assume that consecutive routing decisions regarding a packet are independent, then we can estimate the expected number of packets transmissions (trials) necessary to move a packet from a distance k to a distance $k-1$ as $\frac{1}{1-p}$. Since the optimal number of trials is 1, then the (average) wasted bandwidth in this one hop transmission is $size\_of\_data * (\frac{1}{1-p} - 1) = size\_of\_data * \frac{p}{1-p}$. Thus, the average number of transmissions wasted when forwarding a packet from a source L_i hops away from the destination is $size\_of\_data * \frac{p}{1-p} * L_i$. Finally, since $\lambda_t * N$ packets are generated each second, and the average (optimal) path length of a packet is L, the bandwidth wasted due to suboptimal routes is $\lambda_t * N * size\_of\_data * \frac{p}{1-p} * L = \Theta(\lambda_t * N^{1.5})$, where the last equality holds since $L = \Theta(\sqrt{N})$ (assumption a.4).

The above assumption that consecutive routing decision are independent is obviously not true since node routing decisions (and therefore mistakes) are highly correlated over time (until new link status information is available or some other mechanism – as for example loop detection – is provided to help not to repeat a wrong decision) and space. Thus, this model ignores the presence of loops and other phenomena. However, the model is good enough to make the point that if p does not depend on size or traffic, a protocol's suboptimal routing overhead increases as $\Theta(\lambda_t * N^{1.5})$ with respect to size and traffic.

The remaining of this subsection will focus on showing that the HSLS probability of a non-optimal next hop decision is independent of the network size and approximately constant for different distances to the destination. The independence of p from traffic is obvious since HSLS follows a proactive approach where routing information is propagated as a consequence of events (link status changes) that are independent of the traffic. [3]

To analyze the probability of a non-optimal next hop decision we need to go back to Figure D.1. There $D_{current}$ represents the actual (topological) position of node D. D^j_{past}, with $j = 1, 2$ represents two possibilities of node D's topological position as seen by node S, k hops away (who may not have up to date information). The small region $\mathcal{B}$ is the set of nodes that belong to any of the minimum distance paths from S to D. I_1, I_2, and I_3 belong to this set, and therefore if S chooses one of these nodes as the next hop, an optimal next hop decision will have been made. The larger region ($\mathcal{A}$) represents the set of nodes for whom the shortest path first algorithm run over S's (out-of-date) topology table gives as output I_1, I_2, or I_3. In our example D^1_{past} belongs to this set whereas D^2_{past} does not. At this point the reader may be confused since Figure D.1 seems to present areas while our discussion refers to topologies. In general the sets aforementioned do not have to cover compact areas and may have arbitrary shapes. Figure D.1 presents what is expected to be an average case (due to our geometrical analogies motivated by assumptions a.2, a.3, and a.4). Thus, it is expected that the set of nodes described above cover a more-or-less compact area and that the success of the next hop decision made by S is intimately related to the fact that at the time when the last LSU was received from D, the physical position (that induces the network topology) of node D was inside the green area (as for example is the case on D^1_{past}). This assumption is more realistic when dealing with large distances.

In the above setting, the probability of a non-optimal next hop decision p (at least for an asymptotically large network) will be the probability that at the time t_{past} when the last LSU was received (or assumed) the node's position was not inside the green

[3] In some protocols LSU generation and traffic could be correlated. For example, if eavesdropping of application level acknowledgments is used to estimate the status of a link. This case has not been considered in this work, although it can be intuitively understood that in such a case the protocol performance will only improve due to quick link failure detection.

area ($\mathcal{A}$) given that at the current time $t_{current}$ the node is in the position $D_{current}$. This probability clearly depends on the time elapsed ($t_{elapsed} = t_{current} - t_{past}$) since 'fresh' information was last received, and on the node mobility model. Assumption a.8 implies that p is a function of $\frac{t_{elapsed}}{k}$. The particular form of this function will depend on $g_{0/1}(x, y)$ (i.e. the traffic model) and the normalized area $\mathcal{A}'$ (result of compressing area $\mathcal{A}$ so that the distance from S to D is unity). Thus $p = h(\frac{t_{elapsed}}{k}, g_{0/1}(x, y), \mathcal{A}')$. $h(.,.,.)$'s form may be complex but it is clear that it will be nondecreasing with $\frac{t_{elapsed}}{k}$ and non-increasing with $\mathcal{A}'$.[4]

Thus, if we can lower bound $\mathcal{A}'$ and upper bound $\frac{t_{elapsed}}{k}$ when N increases to infinity we will have shown that p is bounded independently of N (we already assumed that a node mobility model – defined by its function $g_{0/1}(x, y)$ is independent of the number of nodes). To see that the average $\mathcal{A}'$ is lower bounded, consider that in the worst case the set of optimal next hop decisions is formed by just one node (out of the d neighbors – on average – of a node). Thus, if the network is balanced (in average), the number of nodes that will return the optimal next hop node as the output of their shortest path first algorithm should be roughly N/d. Therefore, the region $\mathcal{A}'$ will consist (on average) of a fraction of at least $1/d$ of the total area.

To see that $\frac{t_{elapsed}}{k}$ is upper bounded, we must consider that a node (say S) that is k hops away from another (say D), where $2^i < k \leq 2^{i+1}$, will receive updates about any link change detected by node D at most after $2^i * t_e$ seconds. Thus, if no LSU has been received in a long time, then at time t the node (S) knows that up until time $t - 2^i * t_e$ no link status change has been experienced by node D (which is equivalent to saying that the relative position of node D with respect to their neighbors has not changed much). [5] It still remains the possibility that D's neighbors move as a group but this will be detected by nodes closer to S and S will be alerted of this changes. Thus, it is safe to say that the time $t_{elapsed}$ since S heard about D's whereabouts for the last time is lower than $2^i * t_e$. Thus $\frac{t_{elapsed}}{k} < \frac{2^i * t_e}{k} = \frac{2^i}{k} * t_e < t_e$. And since t_e is independent from the network size, we conclude that p is bounded as N grows to

[4] i.e. if one region totally contains another, h evaluated in the former may not be greater than h evaluated in the latter.

[5] Recall that in this normalized model the distance between 2 neighbors is small compared with the area cover by a large number of nodes since density is not allowed to increase beyond a limit.

infinity. Thus, HSLS's suboptimal routing overhead is $\Theta(\lambda_t * N^{1.5})$.

It is interesting investigate the HSLS's suboptimal routing overhead dependency on t_e and if possible on speed (s). In order to gain insight with tractable solutions, some extra assumptions need to be made. For example, consider that $g_{0/1}$ is such that the functional form of p follows the probability distribution function typically used for analyzing residence time in cellular systems; we can consider mobility models that produce a value of $p = 1 - e^{\frac{-s\bar{t}_{elapsed}K_1}{k}}$, where K_1 is a constant that depends on the topology, average node degree, etc., and $\bar{t}_{elapsed}$ is the average time elapsed since correct link status information regarding the destination was available. Such a function is based on the underlying assumption that the expected node position after a given time varies linearly with the speed, thus the speed is directly proportional to the rate of topological change (i.e. doubling the speed will be equivalent to 'compressing' the time between events by a factor of 2). Then, we can focus on networks where the functional form of p is defined by $p = 1 - e^{\frac{-\lambda_{lc}\bar{t}_{elapsed}K_2}{k}}$. Note that this expression for p, depending on the rate of link changes (λ_{lc}), makes more sense when dealing with topologies and may even be true if we relax our mobility constraints (assumptions). In networks where the above assumptions are true, the HSLS suboptimal routing overhead is equal to $K_3\frac{p}{1-p}\lambda_t N^{1.5} = K_3 * (e^{\frac{\lambda_{lc}\bar{t}_{elapsed}K_2}{k}} - 1)\lambda_t N^{1.5}$, where K_2 and K_3 are constants. Considering the ratio $\frac{\bar{t}_{elapsed}}{k}$ it has already been shown that a node (S) that is k hops away of another (D) with $2^i < k \leq 2^{i+1}$ will experience a delay in the reception of new link state information about D that is bounded by $2^i * t_e$ seconds. It is not difficult to visualize that on average node S will experience a delay of $\frac{2^i * t_e}{2}$. Thus, the average ratio $\frac{\bar{t}_{elapsed}}{k} = \frac{2^i * t_e}{2 * k}$ will be bounded by $\frac{t_e}{2} > \frac{\bar{t}_{elapsed}}{k} \geq \frac{t_e}{4}$. Thus, the HSLS's suboptimal routing overhead is equal to $K_3 * (e^{\lambda_{lc}t_e K_4} - 1)\lambda_t N^{1.5}$, where K_3 and K_4 are constants.

Bibliography

[1] R. Gallager, D. Bertsekas, *Data Networks.* Prentice Hall, New Jersey, 1992.

[2] R. Perlman, *Interconnections: Bridges and Routers.* Addison-Wesley, Readings, Massachusetts, 1992.

[3] J. McQuillan, I. Richer, and E. Rosen, " The new routing algorithm for the ARPANET," *IEEE Transactions on Communications*, 28(5):711-719, May 1980.

[4] International Standards Organization, *Intermediate system to intermediate system intra-domain routing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473).* ISO DP 10589, February 1990.

[5] J. Moy, " OSPF version 2," *Internet Request For Comments RFC 1247*, July 1991.

[6] J. McQuillan and D. Walden, " The ARPA network design decisions," *Computer Networks*, 1(5):243-289, August 1977.

[7] C. Hedrick, " Routing Information Protocol ", *Internet Request For Comments RFC 1058*, June 1988.

[8] G. Sidhu, R. Andrewa, and A. Oppenheimer, *Inside Appletalk.* Addison-Wesley, Reading, Massachusetts, 1990.

[9] P. Turner. " NetWare communications processes," *NetWare Application Notes*, Novell Research, pages 25-81, September 1990.

[10] Xerox Corporation, *Internet transport protocols.* Xerox System Integration Standard 028112, December 1981.

[11] J. Jubin and J. Tornow, " The DARPA packet radio network protocols," *Proceedings of the IEEE*, 75(1):21-32, January 1987.

[12] P. Merlin and A. Segall, " A Failsafe Distributed Routing Protocol," *IEEE Transactions on Communications*, September 1979.

[13] J. Jaffe and M. Moss, " A Responsive Distributed Algorithm for Computer Networks," *IEEE Transactions on Communications* 30(7), July 1982, pages 1758-1762.

[14] A. Segall, " Distributed Network Protocols," *IEEE Transactions on Information Theory* 29(1), pages 23-35, January 1983.

[15] J. J. Garcia-Luna-Aceves, " A Distributed loop-free, shortest-path routing algorithm," *Proceedings of IEEE INFOCOM'88*, March 1988.

[16] J. J. Garcia-Luna-Aceves, " Loop-free Routing using Difussing Computations," *IEEE Transactions on Networking*, 1(1), February 1993.

[17] C. Perkins, and P. Bhagwat, " Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proceedings of ACM SIGCOMM'94*, vol.24, no.4, October 1994.

[18] P. Humblet, " Another Adaptive Shortest-Path Algorithm ", *IEEE Transaction on Communications*, June 1991.

[19] E. Gafni and D. Bertsekas, " Distributed Algorithms for Generating Loop-Free Routes in Networks with frequently changing topology,"*IEEE Transactions on Communications*, (January 1981).

[20] R.S. Kahn, J. Gronemeyer, J. Burchfield, and R. Kunzelman, " Advances in Packet Radio Technology," *Proceedings of the IEEE*, 66(11)(1978) 1468-1496.

[21] K. Klemba et al., *Packet Radio Executive Summary.* Technical Report prepared for Defense Advanced Research Projects Agency by SRI International, July 1983.

[22] G. Lauer, " Packet Radio Routing," In *Routing in Communications networks*, edited by Martha E. SteenStrup, Chapter 11, pages 55-76. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[23] G. Lauer, " Hierarchical Routing Design for SURAN," *Proceedings of ICC'86*, 1986, pages 93-101.

[24] N. Shacham and J. Westcott, " Future Directions in Packet Radio Architectures and Protocols," *Proceedings of the IEEE* 75(1)(1987) 83-99.

[25] J. Stevens, " Spatial Reuse Through Dynamic Power and Routing Control in Common-Channel Random-Access Packet Radio Networks," *SURAN Program Technical Note (SRNTN) 59*, Richarson, TX: Rockwell Inc., August 1988. Available from Defense Technical Information Center (DTIC).

[26] M. Pursley, " Routing in Frequency-Hop Packet Radio Networks with Partial-Band Jamming," *Proceedings of the Tactical Communications Conference*, Fort Wayne, April 1990, pages 117-126.

[27] R. Ogier, V. Rutenburg, and N. Shacham, " Distributed Algorithms for computing shortest Pairs of Disjoint Paths," *IEEE Transaction on Information Theory*, March 1993, pages 443-455.

[28] A. Segall, " Advances in Verifiable Fail-Safe Routing Procedures," *IEEE Transactions on Communications,* Vol 29, pages 491-497. 1981.

[29] J. Garcia-Luna-Aceves and N. Shacham, " Analysis of Routing Strategies for Packet Radio Networks," *Proceedings of IEEE INFOCOM'85*, Washington, DC, March 1985, pages 292-302.

[30] W. Zaumen and J. Garcia-Luna-Aceves, " Steady-State Response of Shortest-Path Routing Algorithm," *Proceedings of IPCCC'92*, April 1992, pages 323-332.

[31] R. Perlman, *Network Layer Protocols with Byzantine Robustness.* Ph.D. Thesis, Massachusetts Institute of Technology, August 1988.

[32] J. Zavgren and M. Leib, " High-Throughput, Survivable Protocols for CDMA Packet-Radio Networks," *SURAN Program Technical Note (SRNTN) 74*, Cambridge, MA : BBN Systems and Technologies Corporation, February 1990.

[33] P. Bausbacher, " Steady State Evaluation of transmission Parameter Selection Algorithms Using Markov Techniques," *SURAN Program Technical Note (SRNTN) 87*, Richarson, TX: Rockwell Inc., January 1990.

[34] J. Escobar, " Radio-Parameter Selection Algorithm for Receiver-Directed Packet-Radio Networks," *SURAN Program Technical Note (SRNTN) 73*, Cambridge, MA : BBN Systems and Technologies Corporation, 1989.

[35] B. Leiner, D. Nielson, and F. Tobagi, " Issues in Packet Radio Network Design," *Proceedings of the IEEE* 75(1)(1987)6-20.

[36] J. Stevens, " Spreading Connectivity Information out over Multiple PROP Periods, and Timeliness of Information," *SURAN Program Technical Note (SRNTN) 21*, Richarson, TX : Rockwell Inc., May 1985. Available form Defense Technical Information Center (DTIC).

[37] R. Callon and G. Lauer, " Hierarchical Routing for Packet Radio Networks," *SURAN Program Technical Note (SRNTN) 31*, Cambridge, MA : BBN Systems and Technologies Corporation, June 1985. Available form Defense Technical Information Center (DTIC).

[38] R. Callon and G. Lauer, " More Issues in Hierarchical Routing for SURAP2," *SURAN Program Technical Note (SRNTN) 34*, Cambridge, MA : BBN Systems and Technologies Corporation, July 1985. Available form Defense Technical Information Center (DTIC).

[39] K. Klemba and N. Shacham, " An Architecture for Large Packet Radio Network and Some Implementation Considerations," *SURAN Program Technical Note*

(SRNTN) 11, Menlo Park, CA : SRI International, October 1983. Available form Defense Technical Information Center (DTIC).

[40] N. Shacham, " Organization of Dynamic Radio Network by Overlapping Clusters : Architecture Considerations and Optimization," *Proceedings of the Tenth International Symposium on Computer Performance*, Paris, December 1984, pages 435-447.

[41] N. Shacham, " Hierarchical Routing in Large, Dynamic Ground Radio Networks," *Proceedings of the Eighteenth Hawaii International Conference on System Sciences*, 1985, pages 292-301.

[42] P. F. Tsuchiya, " Landmark Routing : Architecture, Algorithms, and Issues," *Technical Report MTR-87W00174*, Cambridge, MA: MITRE Corporation, September 1987.

[43] P. Karn and H. Price, " Packet radio in the amateur service," *IEEE Journal on Selected Areas in Communications*, 3(3):431-439, May 1985.

[44] D. Frank, " Transmission of IP datagrams over NET/ROM networks," *Proceedings of ARRL Amateur Radio 7th Computer Networking Conference*, pages 65-70, October 1988.

[45] B. Garbee, " Thoughts on the issues of address resolution and routing in amateur packet radio TCP/IP networks," *Proceedings of ARRL Amateur Radio 6th Computer Networking Conference,* pages 56-58, August 1987.

[46] J. Geier, M. DeSimio, and B. Welsh, " Network routing techniques and their relevance to packet radio networks," *Proceedings of ARRL/CRRL Amateur Radio 9th Computer Networking Conference,* pages 105-117, September 1990.

[47] Internet Engineering Task Force (IETF), Mobile Ad Hoc Networking (MANET) Working Group drafts in http://info.internet.isi.edu:80/R284713-291510-1m/in-drafts/id-abstracts.html

[48] J. Macker and S. Corson, “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations,” IETF MANET Working Group Internet-Draft, Work in Progress.

[49] C. Perkins, “ Mobile Ad Hoc Networking Terminology,” IETF MANET Working Group Internet-Draft, Work in Progress.

[50] S. Corson, J. Macker, and S. Batsell, “ Architectural Considerations for Mobile Mesh Networking, ” IETF MANET Working Group Internet-Draft, Work in Progress.

[51] S. Corson, “ MANET Routing Protocol Applicability Statement,” IETF MANET Working Group Internet-Draft, Work in Progress.

[52] S. Corson, V. Park, P. Papadopoulos, S. Papademetriou, and A. Qayyum, “ An Internet MANET Encapsulation Protocol (IMEP) Specification,” IETF MANET Working Group Internet-Draft, Work in Progress.

[53] D. Johnson, D. Maltz, and J.Broch, “ The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks,” IETF MANET Working Group Internet-Draft, Work in Progress.

[54] C Perkins, and E. Royer, “ Ad Hoc On Demand Distance Vector (AODV) Routing,” ETF MANET Working Group Internet-Draft, Work in Progress.

[55] S. Corson, V. Park, “ Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification,” IETF MANET Working Group Internet-Draft, Work in Progress.

[56] Z. Haas and M. Pearlman, “ The Zone Routing Protocol (ZRP) for Ad Hoc Networks,” IETF MANET Working Group Internet-Draft, Work in Progress.

[57] P. Jacquet, P. Muhlethaler, and A. Quayyum, “Optimized link state routing protocol”, IETF MANET Working Group Internet-Draft, Work in Progress

[58] R. Sivakumar, P.Sinha, and V. Bharghavan, " Core Extraction Distributed Ad hoc Routing (CEDAR) Specification," IETF MANET Working Group Internet-Draft, Work in Progress.

[59] M. Jiang, J. Li, and Y. C. Tay, " Cluster Based Routing Protocol(CBRP) Functional Specification," IETF MANET Working Group Internet-Draft, Work in Progress.

[60] R. Sivakumar, P. Sinha, and V. Bharghavan, " CEDAR: a Core-Extraction Distributed Ad hoc Routing algorithm," *Proceedings of INFOCOM'99*, New York, 1999.

[61] C. Perkins, " Ad-Hoc On-Demand Distance Vector Routing," *Proceedings of MILCOM'97 panel on Ad-Hoc Networks*, Monterey, CA, November 3, 1997.

[62] Z. Haas, " A New Routing Protocol for the Reconfigurable Wireless Networks," *Proceedings of ICUPC'97,* San Diego, CA, Oct. 12, 1997.

[63] V. D. Park, and S. Corson, " A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *Proceedings of IEEE Infocom '97,* Kobe, Japan(1997).

[64] D. B. Johnson and D. Maltz," Dynamic Source Routing in Ad Hoc Wireless Networks," In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Kluwer Academic Publishers, 1995.

[65] C. K. Toh, " A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing," *Proceedings of the International Phoenix Conference on Computers and Communications (IPCCC'96),* pages 480-486, March 1996.

[66] C-K Toh, " Associativity Based Routing For Ad Hoc Mobile Networks," *Wireless Personal Communications Journal*, Special Issue on Mobile Networking & Computing Systems, Vol. 4, No. 2, March 1997.

[67] A. B. McDonald and T.F. Znati. “A Mobility Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks”. *IEEE Journal of Selected Areas on Communications*, col. 17, no. 8, pp. 1466-1487, Aug. 1999.

[68] T. Chen and M. Gerla, “ Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks ,” *Proceedings of IEEE ICC ’98*, 1998.

[69] T. Chen, M. Gerla, and J. T. Tsai, “ QoS routing performance in a multi-hop, wireless networks,” *Proceedings of ICUPC ’97*, 1997.

[70] S. Corson, and A. Ephremides, “ A Distributed Routing Algorithm for Mobile Wireless Networks,” *ACM/Baltzer journal on Wireless Networks*, January 1995.

[71] S. Murthy, and J. Garcia-Luna-Aceves, “ A Routing Protocol for Packet Radio Networks,” *Proceedings of MOBICOM’95*, November 14-15, 1995.

[72] S. Murthy and J. J. Garcia-Luna-Aceves, “ An efficient Routing Protocol for Wireless Networks,” *ACM Mobile Networks and applications Journal*, Special issue on Routing in Mobile Communication Networks, 1996.

[73] B. Das, S. Raghupathy, and B. Vaduvur, “ Routing in Ad Hoc Networks Using a Spine,” *Proceedings of the 6th International Conference on Computer Communications and Networks,* Las Vegas, USA, September 1997.

[74] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradham, “ A Cluster-based Approach for Routing in Dynamic Networks,” *Proceedings of the Second Usenix Symposium on Mobile and Location-Independent Computing*, 1995.

[75] M. Gerla and T.C. Tsai, “ Multicluster, mobile, multimedia radio network,” *ACM/Balzer Journal of Wireless Networks*, 1995.

[76] C.C. Chiang, H. K. Wu, W. Liu, M. Gerla, “ Routing in Clustered Multi-hop, Mobile Wireless Networks with Fading Channel,” *Proceeding of the IEEE SICON,* 1997.

[77] C.R. Lin and M. Gerla, " Multimedia Transport in Multihop Dynamic Packet Radio Networks," *Proceedings of IEEE GLOBECOM'95,*, pages 209-216, 1995.

[78] V. D. Park, and S. Corson, " A Performance Comparison of the Temporally-Ordered Routing Algorithm and Ideal Link-State Routing," *Proceedings of IEEE Symposium on Computers and Communications ISCC'98*, Athens, Greece, June 1998.

[79] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, " A Performance Comparison of Multihop Wireless Ad Hoc Network Routing Protocols," *Proceedings of MOBICOM'98*, Dallas, TX., October 1998.

[80] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks", *IEEE Personal Communications Magazine*, Vol. 8, No. 1, Feb. 2001.

[81] P. Jacquet and L. Viennot, 'Overhead in Mobile Ad-hoc Network Protocols", *INRIA Research Report 3965, Institut National de Recherche en Informatique et en Automatique (INRIA)*, France, June 2000.

[82] R. Guerin, et. al., "Equivalent Capacity and Its Applications to Bandwidth Allocation in High Speed Networks," *IEEE Journal of Selected Areas on Communications*, vol. 9, no. 7, pp. 968-981, Sept. 1991.

[83] P. Gupta and P.R. Kumar. "The Capacity of Wireless Networks", *IEEE Transaction on Information Theory*, 46 (2):388-404, March 2000.

[84] M. Grossglauser and D. Tse. "Mobility Increases the Capacity of Ad-hoc Wireless Networks", in *Proceedings of IEEE Infocom'2001*, Anchorage, Alaska, April 2001.

[85] M. Sidi and A. Segall, " Busy-Tone-Multiple-Access-Type Scheme for Packet-Radio Networks," *Proceedings of the performance of Data Communications Systems and Their Applications Conference,* pages 1-10, Paris, September 1981.

[86] M. Pursley, "The role of Spread Spectrum in Packet Radio Networks," *Proceedings of the IEEE*, Vol. 75, No 1, pages 116-134, January 1987.

[87] A. Ephremides, and T. Truong, " Scheduling Broadcasts in Multihop Radio Networks," *IEEE Transactions on Communications,* Vol. 38, No. 4, April, 1990.

[88] P. Karn, " MACA – A New Channel Access Method for Packet Radio," *Proceedings of the ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134-140, September 1990.

[89] W. Diepstraten, G. Ennis, and P. Berlanger, " DFWMAC : Distributed Foundation Wireless Medium Access Control," *IEEE Document* P802.11-93/190, November 1993.

[90] F. Talucci and M. Gerla, " MACA-BI (Maca by invitation). A Wireless MAC Protocol for High Speed Ad Hoc Networking," *Proceedings of IEEE ICUPC'97*, 1997.

[91] J. Deng and Z. J. Haas, " Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks," *Proceedings of IEEE ICUPC'98*, Florence, Italy, October 5-9, 1998.

[92] G. Lauer, " Address Servers in Hierarchical Networks," *Proceedings of the ICC'88*, pages 443-451, 1988.

[93] T.C. Hou and V.O.K. Li, " Position Updates and Sensitivity Analysis for Routing Protocols in Mobile Packet Radio Networks," *Proceedings of IEEE GLOBECOM'85,* pages 243-249, 1985.

[94] T. Hou and V. Li, " Performance Analysis of Routing Strategies in Multihop Packet Radio Networks," *Proceedings of GLOBECOM'84*, pages 487-492, Atlanta, November 1984.

[95] H. Takagi and L. Kleinrock, " Optimal transmission Ranges for Randomly Distributed Packet Radio Terminal," *IEEE Transactions on Communications,* 32(3), pages 246-257, March 1984.

[96] L. Kleinrock and J. Silvester, " Optimum Transmission Radii for Packet Radio Networks, or Why Six is a Magic Number," *Proceedings of the National Telecommunications Conference*, pages 4.3.1-4.3.5, December 1978.

[97] J.R. Zavgren, " The Moment of Silence Channel-Access Algorithm," *Proceedings of MILCOM'89,* Cambridge, MA, October 1989.

[98] S. Ramanathan and M. Steenstrup, " A survey of routing techniques for mobile communications networks," *ACM/Baltzer Mobile Networks and Applications*, Vol. 1, No. 2, pp. 89-103.

[99] I. Stavrakakis, " Management of Communication Resources in a Hierarchical Mobile Mesh Network.," *Technical Report TR-CDSP-98-49*, Communications and Digital Signal Processing Center, ECE Dept., Northeastern University, Boston, MA. 1998.

[100] A.S. Tanenbaum, "Computer Networks", Prentice-Hall, 1996.

[101] C. Perkins, and P. Bhagwat. "Highly Dynamic Destination-Sequenced Distance -Vector Routing (DSDV) for Mobile Computers," *ACM SIGCOMM*, vol.24, no.4, October 1994.

[102] S. Murthy and J.J. Garcia-Luna Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM Mobile Networks and Applications* Special Issue on Routing in Mobile Networks, Oct. 1996, pp. 183-97.

[103] B. Bellur, R. Ogier, "A Reliable, Efficient Topology Broadcast Algorithm for Dynamic Networks," *Proc. IEEE INFOCOM*, 1999.

[104] J.J. Garcia-Luna-Aceves and M. Spohn, "Source-Tree Routing in Wireless Networks,", *Proc. IEEE ICNP 99: 7th International Conference on Network Protocols*, Toronto, Canada, October 31–November 3, 1999.

[105] S. Ramanathan, M. Steenstrup, "Hierarchically-organized, Multihop Mobile Networks for Multimedia Support", *ACM/Baltzer Mobile Networks and Applications*, Vol. 3, No. 1, pp 101-119.

[106] G. Lauer, “Packet Radio Routing”, in *Routing in Communication Networks*, ed. M. Steenstrup, Prentice-Hall, 1995.

[107] D. B. Johnson and D. Maltz,“*Dynamic Source Routing in Ad Hoc Wireless Networks.*”, In Mobile Computing, edited by Tomasz Imielinski and Hank Korth. Kluwer Academic Publishers, 1995.

[108] C. Perkins. “Ad-Hoc On-Demand Distance Vector Routing”. MILCOM’97 panel on Ad-Hoc Networks, Monterey, CA, November 3, 1997.

[109] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward, “A Distance Routing Effect Algorithm for Mobility (DREAM),” in *Proceedings of ACM/IEEE MobiCom’98*, Dallas, Tx, 1998.

[110] Y. B. Ko and N. H. Vaidya. “Location-Aided Routing (LAR) in Mobile Ad Hoc Networks”, in *Proceedings of ACM/IEEE MobiCom’98*, Dallas, TX, 1998.

[111] V.D. Park and M.S. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks,” in *IEEE INFOCOM ’97*, pp. 1405-1413, 1997.

[112] Z. Haas and M. Pearlman, “The performance of query control schemes for the zone routing protocol,” in *ACM SIGCOMM*, 1998.

[113] M. R. Pearlman and Z. J. Haas, “Determining the Optimal Configuration for the Zone Routing Protocol,” *IEEE Journal of Selected Areas on Communications*, vol. 17, no. 8, pp. 1395-1414, Aug. 1999.

[114] R. Ramanathan and R. Hain, “Topology Control of Multihop Radio Networks using Transmit Power Adjustment,” in *Proceedings of IEEE Infocom’2000*, Tel Aviv, Israel, 2000

[115] http://www.ir.bbn.com/projects/dawn/dawn-index.html

[116] B. A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, “Scalable Routing Strategies for Ad Hoc Wireless Networks”. *IEEE Journal of Selected Areas on Communications*, vol. 17, no. 8, pp. 1369-1379, Aug. 1999.

[117] C. Perkins, editor. *IP Mobility Support.* RFC 2002, October 1996.

[118] A. Myles, D. B. Johnson and C. Perkins, "*A Mobile Host Protocol Supporting Route Optimization and Authentication .*", IEEE Journal of Selected Areas in Communications, 13(5) pp 839-849, June 1995.

[119] D. B. Johnson and D. Maltz,"*Dynamic Source Routing in Ad Hoc Wireless Networks.*", In Mobile Computing, edited by Tomasz Imielinski and Hank Korth. Kluwer Academic Publishers, 1995.

[120] G. Pei, M. Gerla and X. Hong, "LANMAR: Landmark Routing for LArge Scale Wireless Ad Hoc Networks with Group Mobility", in *Proceedings of ACM Workshop on Mobile and Ad Hoc Networking and Computing MobiHOC'00*, Boston, MA, August 2000.

[121] C. Santiváñez and I. Stavrakakis, " A Framework for a Multi-mode Routing Protocol for (MANET) Networks", in *Proceedings of IEEE WCNC'99*, New Orleans, LO, September 1999.

[122] C. Santiváñez and I. Stavrakakis, " SOAP : a Self-Organizing, Adaptive Protocol for routing in large, highly mobile ad-hoc networks", *Technical Report TR-CDSP-99-50, CDSP center*, Ece Dept., Northeastern University, Boston, MA, 1999.