Ad Hoc Wireless Networking
X. Cheng, X. Huang and D.-Z. Du (Eds.) pp. – – –
©2003 Kluwer Academic Publishers

Scalability of Routing in Ad Hoc Networks: Principles and Practice

César A. Santiváñez
Internetwork Research Department
BBN Technologies (A Verizon Company), Cambridge, MA 02138
E-mail: csantiva@bbn.com

Ram Ramanathan
Internetwork Research Department
BBN Technologies (A Verizon Company), Cambridge, MA 02138
E-mail: ramanath@bbn.com

Contents

1 Introduction

Recent years have witnessed a surge in the interest in ad hoc networks. Spurred by ever-decreasing form-factors and cost of wireless transceivers and processors, a multitude of new applications are emerging. These include short-range ad hoc wireless networks for ubiquitous computing, larger range indoor wireless LANs that operate in ad hoc mode, metropolitan area networks and sensor networks. Standards such as Bluetooth, HomeRF, and IEEE 802.11 are giving impetus to the growth in the number of ad hoc communication enabled devices. Propelled by these trends, ad hoc networks with a large number (e.g. 1000s) of nodes are moving rapidly from the realm of imagination to that of reality.

Deploying such large-scale networks requires *scalable* ad hoc routing protocols. There are a number of questions that arise when considering routing scalability, including: *what does scalability really mean*, *what factors does scalability depend on*, *how scalable are current-day ad hoc networks routing protocols*, etc. Answering these questions is of paramount importance in understanding and evaluating the set of mechanisms available for improving ad hoc routing scalability, including, for example: hierarchical routing, efficient flooding, limited search/local repair (for reactive protocols), limited dissemination, etc.

This chapter addresses the scalability of routing in ad hoc networks from a fundamental viewpoint, and covers the state of the art in the metrics, methodology, and techniques for designing highly scalable protocols. Both the *principles* underlying routing scalability, and the protocols that put them into *practice* are presented in a balanced manner. The intention is to provide the reader with an insight into the scalability of and tradeoffs inherent in a particular protocol, and introduce him or her to the design of highly scalable protocols.

Ad hoc routing protocols can be broadly classified into proactive (or table-driven) and reactive (or on-demand). While our treatment is mostly in terms of proactive protocols, we emphasize that the basic principles are quite generic and may well be applied to reactive protocols. In other words, we present scalability principles in a given context but the principles in general are not tied to that context.

This chapter is organized as follows. Section 2 presents a brief overview of current ad hoc routing protocols and provides a more detailed description of a selected set of representative protocols that will be used for comparison purposes. Section 3 presents a theoretical foundation for the study of scalability, defining a routing scalability metric and a framework that has proven to be useful in developing tractable models and obtaining closed form expressions for ad hoc routing protocols. Section 4 presents the results obtained when applying the methodologies described in Section 3 to the set of representative protocols described in Section 2.

These results shed a new light into our understanding of scalability. They provide the reader with a better understanding of the interactions and combined effect of increasing the network size, mobility, and/or traffic load. An important result is that limited-dissemination flat routing techniques, as well as hierarchical routing techniques, present the best scalability properties regarding network size. Thus, an important question arises: *Which approach is preferable : hierarchical or flat routing?* And, *Under what circumstances is each of these approaches better?* Section 5 answers these questions. Section 5 starts by discussing the design approaches for making ad hoc routing protocols highly scalable. These approaches can be broadly classified into "flat" and "hierarchical". Using an exemplary protocol in each class, we discuss the merits/demerits of each approach and then compare their theoretical and experimental performance. Finally, Section 6 presents some conclusions and future research directions.

2 Overview of Ad Hoc Routing Protocols

The surge of interest in ad hoc networks has given rise to a plethora of ad hoc routing protocols. To individually study each of them would constitute a gigantic task. Fortunately, from a scalability point of view, we can group protocols together into classes, and therefore focus on representative protocols in each class.

Routing protocols can typically be classified as *proactive* and *reactive*. Proactive protocols attempt to constantly build routes to destinations, so that they are readily available when needed. Standard Link State (SLS) [1], Distance Vector (DV) algorithms based on the Distributed Bellman-Ford (DBF) algorithm[2], Optimized Link State Routing Protocol (OLSR)[3], and Topology Broadcast Based on Reverse Path Forwarding (TBRPF)[4] are examples of proactive routing protocols. Reactive protocol build routes upon request (from the upper layers) so they do not waste bandwidth transmitting routing information when this information is not needed. Ad Hoc On-Demand Distance Vector (AODV)[5, 6], Dynamic Source Routing (DSR)[7], Associativity-Based Routing (ABR)[8], Temporally Ordered Routing Algorithms (TORA)[9], and Distance Routing Effect Algorithm for Mobility (DREAM)[10] are examples of reactive protocols.

Due to its simplicity, quick convergence, well understood dynamics, and good performance, SLS is a good representative of the class of *pure flat* proactive protocols. SLS will be used in the remainder of this chapter as a representative of the class of pure proactive protocols. However, it should be kept in mind that most of the conclusions hold for all the protocols in the same class. Similarly, DSR is chosen as the representative of pure reactive protocols, mainly due to its simplicity and the fact that this is the reactive protocol that has received most attention in the literature. Indeed, DSR is typically the benchmark most reactive routing protocol designers use to compare its designs against.

As size increases, flat routing techniques such as SLS and DSR are no longer efficient. A typical solution (for both proactive and reactive) protocols has been to build a hierarchical structure to limit the control overhead at the expense of route degradation. Hierarchical Link State (HierLS)[11], Hierarchical State Routing (HSR)[12], Link Cluster Architecture (LCA)[13], Clusterhead-Gateway Switch Routing (CGSR)[14], Multimedia support for Mobile Wireless Networks (MMWN)[15], and Adaptive Routing using Clusters (ARC)[16] are examples of hierarchical routing protocols.

The Hierarchical Link State algorithm presented in [11] is a good representative of hierarchical routing techniques. It captures the essence of the hierarchical approaches followed in [15, 12], still it is broad enough to allow analyzing a wide range of design choices for hierarchical routing, mainly regarding location management.

Besides the traditional definition of proactive versus reactive or flat versus hierarchical, there are hybrid protocols. Zone Routing Protocol (ZRP)[17, 18] is an example of such an hybrid approach. ZRP presents a proactive and a reactive component. It successfully adapts its components to different values of mobility over traffic activity radius[19], thus exhibiting a behavior that ranges from pure proactive to pure reactive. Furthermore, even though no hierarchy is being built or maintained, no node aggregation takes place and therefore all the nodes belong to the same level, ZRP behaves as a 2-level reactive hierarchical scheme. ZRP will be used as a representative of the class of hybrid approaches.

More recently, a new class of "limited information" protocols has been proposed. These protocols are flat in the sense that they do not aggregate routing information, and therefore they have the same memory requirements as SLS. But in the other hand, they limit the rate and scope of information dissemination so that the bandwidth consumed by routing updates propagation is reduced. Global State Routing (GSR)[20], and Source-Tree Adaptive Routing (STAR)[21] are examples of protocols in this class that limit the rate of information generation. Fisheye State Routing (FSR[12], and Hazy Sighted Link State (HSLS)[22] are examples of routing protocols that limit not only the generation rate but also the propagation scope of the routing information dissemination. HSLS presents the best performance among protocols in this class and therefore will be use as its representative.

Finally, from a conceptual point of view it is interesting to consider a mechanism that does not use a structured routing algorithm: Plain Flooding (PF). The consideration and comparison of plain flooding will help illustrate the point (network scenario) at which routing protocols break and, it becomes more efficient to just flood each data packet without attempting to locate/keep track of the destination (mobility).

In the remainder of this section a brief description of each of the routing protocols representative of its class is presented. The reader familiar with these protocols (PF, SLS, DSR, HierLS, ZRP, and HSLS) may skip this reading and proceed directly to Section 3.

2.1 Plain Flooding (PF)

In Plain Flooding, a source node S willing to send a packet to a destination D broadcasts this packet to all its neighbors, regardless of the destination identity. Node S's neighbors, in turn rebroadcast the packet once (and just once) to their neighbors, and so on. Every node in the network (unless the network is partitioned) will receive at least one copy of the packet (typically more) and will re-broadcast the packet only once. In particular, the destination node D will receive the packet and deliver it to it upper layers.

In PF, nodes then must keep track of the packets previously sent to avoid sending a packet more than once.

The flooding mechanism described above is typically used to propagate routing control messages inside a network. Flooding is not typically used for normal data packet delivery. However, if the network size is small, the traffic load is small and the mobility rate is very high, PF may be the best routing alternative. Thus, PF can be used as a benchmark against other routing protocols under extremely high mobility.

2.2 Standard Link State (SLS)

SLS and its variants are good representatives of proactive protocols. SLS was initially used in the ARPANET[1] as the replacement of the original Distance-Vector based routing protocol. Since then, several link state protocols have been developed and are being used over different networks, including OSPF, IS-IS, and NLSP, among others. SLS's success was a consequence of its being simple yet robust, and having predictable dynamics and quick convergence in the presence of topology changes.

In SLS, a node sends (floods) a Link State Update (LSU) containing a list of its current neighbors (and their associated link costs) to the entire network each time it detects a link status change. A node also sends periodic, soft-state LSUs every T_p seconds. Each node stores a copy of the latest LSU received from each node in the network in a local database referred to as *topology table*. The *topology table* provides each node with information about the entire network connectivity.

To find a route to a destination node D, the source node (as well as each intermediate node along the route) may apply Dijkstra's Shortest Path First (SPF) algorithm[23] over its local copy of the *topology table*.

2.3 Dynamic Source Routing without Route Cache (DSR-noRC)

DSR[7] is a good representative of reactive protocols. DSR has received considerable attention in the literature, especially due to the easy availability of source code for different platforms. Today, DSR is the typical benchmark used to compare against other ad hoc on-demand routing protocols.

In DSR, no proactive information is exchanged. A source node S builds a route to a destination node D by flooding the network with a route request (RREQ) message. When a RREQ message reaches node D (or a node with a cached route towards the destination, if the route-cache option is enabled) a route reply message is sent back to the source node S, including the newly found route. Node S attaches the new route to the header of all subsequent packets destined to node D, and any intermediate node along the route uses this attached information to determine the next hop in the route.

2.4 Hierarchical Link State (HierLS)

As the size of a network increases, maintaining full topology information may become prohibitively expensive. The bandwidth consumed in propagating up-to-date topology information to each node in the network may grow too large. Also, the memory required to store all this information may exceed the node capabilities. Finally, the processing power required to compute routes (for example executing the Dijkstra's algorithm in a link state protocol) in a timely fashion may also exceed the node's processing capabilities.

A hierarchical approach is a technique for aggregating information. Nodes are grouped in sets, several sets are grouped in supersets, and so on, forming a hierarchy. This reduces the size and frequency of information dissemination, and reduces table sizes and processing requirements. All of this is at the cost of reducing the quality (optimality) of the routes.

In this subsection we will discuss the implementation of this aggregation paradigm in a link state context. We will focus on a generic class of hierarchical algorithms named Hierarchical Link State (HierLS) routing[11]. However, the reader should keep in mind that the hierarchical paradigm may be applied to other routing techniques, not only proactive (e.g. Distance Vector-based) but also reactive. Also, different HierLS algorithms may represent (abstract) higher level elements in the virtual topology differently. This discussion focuses on the *virtual node* abstraction.

In the *m-level* HierLS routing[11, 15], 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 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), thus reducing the proactive overhead.

HierLS for mobile networks 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.[1] For example, consider a 4-level network as shown in Figure 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 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 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 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 a combination of both.

[1] Traditional, wireline-based hierarchical routing protocols do not need a location management service, since the address of the node is associated with its location. For example, in the IP protocol the first part of a node address contains the identity of the subnet the node belongs to.

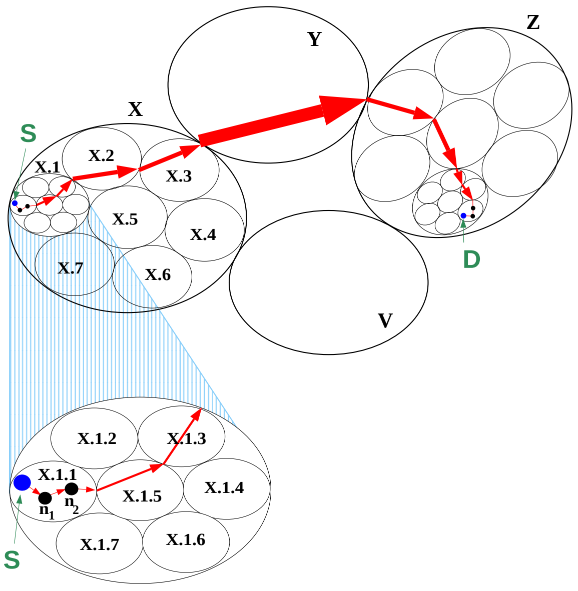

Figure 1: A Source (S) - Destination (D) path in HierLS.

2.5 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). The set of k-hop neighbors constitute the node's *zone*. Each node has full topology information of nodes inside its *zone* and may forward packets to any node within it. When a node needs to forward a packet outside its *zone*, 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 own *zones* (i.e. 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.

The maintenance of the zone structure allows for a reduction in the cost of the route discovery procedure, since instead of flooding the entire network with route request (as done in DSR) ZRP pokes a selected subset of ('border') nodes only. Also, the fact that routes are specified in terms of border nodes only allows 'border' nodes in a path to locally recover from individual link failures, reducing the overhead induced by the route maintenance procedure.

ZRP may dynamically adjust its zone size by increasing or decreasing the value of k to balance the proactive (i.e. propagation information inside the zone) and reactive (i.e. route discovery and maintenance) overheads. ZRP may morph from a fully proactive (k tends to infinity) protocol when a high traffic load is the main challenge to network survivability, all the way to fully reactive (k equal to 1) protocol if the network scenario changes to having node mobility as the main factor limiting network performance. For typical, non-degenerated values of k (i.e. $k > 1$, but smaller than the network diameter) ZRP will resemble a two-level hierarchical network.

2.6 Hazy Sighted Link State (HSLS)

HSLS 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 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 $2^{i-1}t_e$

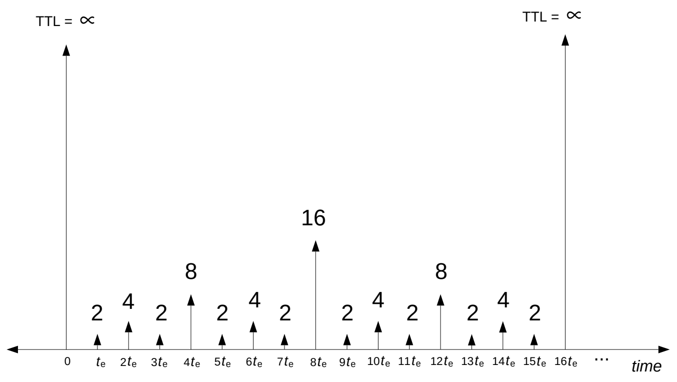

Figure 2: HSLS's LSU generation process (when mobility is high).

seconds. Figure 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 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 $16t_e$. The reader is referred to [22] and [24] for more details about HSLS.

HSLS is a flat routing protocol since each node is represented by an entry in the (distributed) topology table at each node and routes are built by applying Dijkstra's algorithm over the entire set of nodes in the network (high processing and memory requirements). However, HSLS differs from traditional flat protocols in that each node's vision of the topology table is different. Besides, the propagation of the topology information is not done by flooding the entire network each time but by sending information to smaller areas of the network more frequently and gradually increasing the areas' size while decreasing the frequency of propagation, resembling the information dissemination method employed in a multilevel hierarchical approach.

3 Routing Scalability : Theoretical Background

There is a wide consensus on the importance of understanding the scalability limits of both ad hoc networks and their related routing protocols. Surprisingly, however, there is not a consensus of a well defined routing scalability metric. Adding to the chaos, there is sometimes a confusion between the *routing protocol* scalability and the *network scalability* of the network the protocol is run over.

In this section, we review a promising metric, the *total overhead*, that captures the main impact of increasing network limiting parameters on performance of routing protocol running on bandwidth-limited networks. The *limiting parameters* of a network are those parameters – as for example mobility rate, traffic rate, and network size, etc. – whose increase causes the network (and oftentimes the routing protocol) performance to degrade. On the remainder of this chapter only limiting parameters will be considered, and therefore the terms 'parameter' and 'limiting parameter' will be used indistinctly.

The *Total Overhead* metric enables the derivation of tractable models and closed form expressions, providing us with the understanding of the scalability properties of different ad hoc routing protocols.

3.1 Scalability Aspects of Ad Hoc Routing Protocols

When a network limiting parameter such as the network size increases, it impacts the network at several, concurrent levels. For example, if the network is running the Standard Link State (SLS) protocol, where each node advertises its set of neighbors to all other nodes in the network, the increase in the network size will cause:

- An increase in the rate of the Link State Updates (LSU) sent through the network, and therefore an increase in the bandwidth consumed by the control messages of the routing protocol.
- An increase in the memory requirements of each node, since each node must store a local copy of the *Topology Table*, which contains an entry per node in the network.
- An increase in the processing requirements of each node, since the time complexity of the route computation algorithm (e.g. Dijkstra's) used for finding the best route to a given destination increases monotonically with the number of entries in its *topology table* (i.e. the number of nodes in the network). Thus, more operations will need to be done per unit of time, which may require the use of faster processors.
- Since more processing is being done, and more packets are being transmitted, the power consumption also increases. Thus, the battery re-

quirements of a mobile node also increases.

- Since the network diameter (i.e. maximum distance - in hops - between two nodes in the network) also increases, the average delay for delivering a packet to its destination – assuming uniform traffic distribution, i.e. non-local traffic profile – also increases. [2] Also, since the control packets (LSUs) and the data packets share the same transmission medium, the aforementioned increase on the number of LSUs will cause the data packets contending for access to the same channel experiencing a longer delay before succeeding. As size increases the delay may grow so large that real time applications like voice and video can no longer be supported on the network.

Thus, an increase in network size has an impact on several aspects of scalability. Which one of these aspects (bandwidth, memory, processing power, energy consumption, delay, etc.) is the most important depends on the characteristics of the network under study. In particular it depends on which resource is the most scarce, or depleted first. Ad hoc networks tend to be bandwidth-limited, i.e. bandwidth is the most scarce resource, but emerging applications such as sensor networks may well shift the relative importance of the scalability aspects to the point that processing power or battery life become the more important ones.

It is extremely hard to define a metric that simultaneously encompasses the effect of an increase in network parameters on all of the aforementioned scalability aspects. Therefore, in order to build tractable models we must be content with metrics that address each of these scalability aspects independently. Thus, when referring to the bandwidth aspect we can talk about the communication overhead or communication complexity of a protocol. Similarly, when referring to the processing or memory requirements aspect we can talk about the time or memory complexity of the protocol, respectively, and so forth.

As mentioned before, a large class of ad hoc networks are bandwidth-limited, that is, bandwidth is the most scarce resource. In other words, as network parameters increase, it is the lack of additional bandwidth which causes the network to collapse. While a bandwidth-related metric may not fully characterize all the performance aspects relevant to specific scenario

[2]Although in this case (using SLS) the increase in the delay is independent of the routing protocol being used, it is not rare to observe routing protocols (e.g. on-demand ones) where the packet delay increase is caused by latency in the route discovery procedure. This latency is dependent on the network size.

(for example it may fail to capture variation on packet delays) it does capture the main performance degradation due to a network parameter increase. Moreover, a bandwidth related metric is proportional to energy and processing requirements and it has been shown that even delay constraints can be expressed in terms of equivalent bandwidth[25]. In the remainder of this chapter we will present the state-of-the-art in the study of the bandwidth aspect of scalability. Theoretical results and insight into the scalability limits of ad hoc routing were enabled by the introduction of the *Total Overhead* metric in [22, 11].

The reader interested in other aspects of scalability, as for example memory requirements may want to take a look at [26].

3.2 Communication Overhead: Conventional Notions

Traditionally, the term *(communication) 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 such as Standard Link State (SLS) and Distance Vector (DV) the communication 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 such as Dynamic Source Routing (DSR) and Ad hoc On Demand distance Vector (AODV), the communication overhead has been described in terms of the bandwidth consumed by the route request/reply messages (global or local). A primary goal of ad hoc routing protocol research has been to design protocols that keep this control 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 [15, 12]) and/or limiting the time between successive topology information updates dissemination so that topology updates are no longer instantaneously event-driven (e.g GSR [20]).

This leads to the question : how can we define *overhead* so that it includes the effect of suboptimal routes in capacity limited systems? We need to do this since *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 control overhead, while in practice the resulting performance is seriously degraded as the extra bandwidth overhead induced by suboptimal routes increases with the network size. In the next subsections we discuss a more comprehensive definition of overhead that is more useful in the comparative analysis of protocols.

3.3 Emerging concept: Total (Communication) 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 is defined 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. This might be possible in fixed networks when 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 not 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. In an actual mobile ad hoc network, the bandwidth usage would be greater than the *minimum traffic load* value. This motivated the following definition of the *total overhead* of a routing protocol.

Definition 3.2 *The* **total overhead** *induced by a routing protocol X is the difference between the total amount of bandwidth actually consumed by the network running X minus the* **minimum traffic load***.*

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. The different sources of overhead that contribute to the *total overhead* may be classified into *reactive*, *proactive*, and *suboptimal routing overhead.*

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 the traffic *and* the rate 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 computation.

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 the limiting factor in terms of scalability.

3.4 Overhead: Achievable Regions and Operating Points

Having defined a fair metric for overhead, we now ask : is a pure proactive or a pure reactive protocol the best approach for routing scalability? What is the desirable relation/balance between the different classes of overhead in a scalable routing protocol?

We begin by noting that 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 the Zone Routing Protocol (ZRP) [17, 18] will reduce ZRP's proactive overhead, but will increase the overhead induced 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 a given scenario (traffic, mobility, etc.). Figure 3 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 [3], lower-bounded by the curve of overhead points achieved by the 'efficient' (i.e. minimizing some source of overhead given a constraint being imposed on the others) protocols. For example, point P is obtained by the best pure proactive approach given that optimal routes are required – that is, given the constraints that the suboptimal and reactive overheads must be equal to zero. 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 line $x + y = K$, where K is a numerical constant).

Different scenarios result in different slopes of the boundary of the achievable region and consequently different positions for Opt. For example, if the traffic increases (more sessions) or diversifies, R moves upward (pure reactive protocol induces more overhead) and, if mobility is low P moves to the

[3] 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) period of time and as protocol P_2 the remainder of the time.

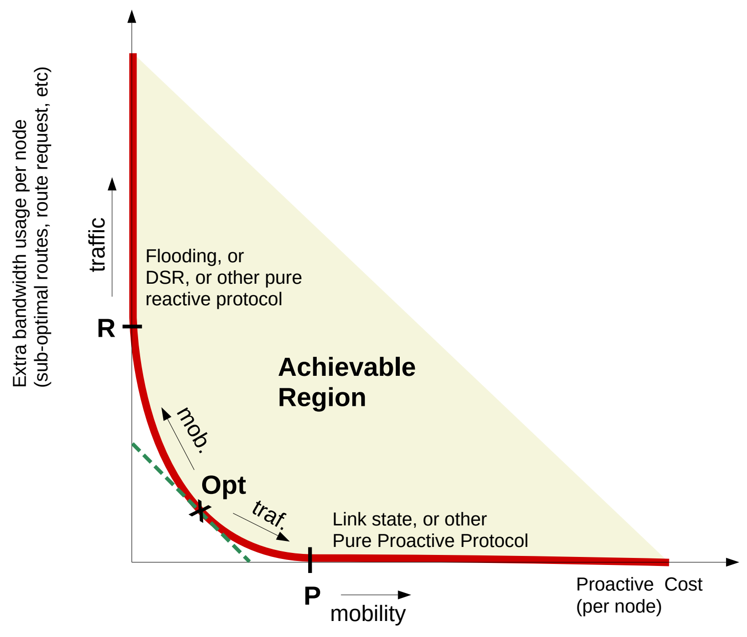

Figure 3: Overhead's achievable region.

left (pure proactive protocol induces less overhead) 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 4 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 along the X and Y axes as will be discussed next.

Pure proactive protocols, such as 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, such as 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

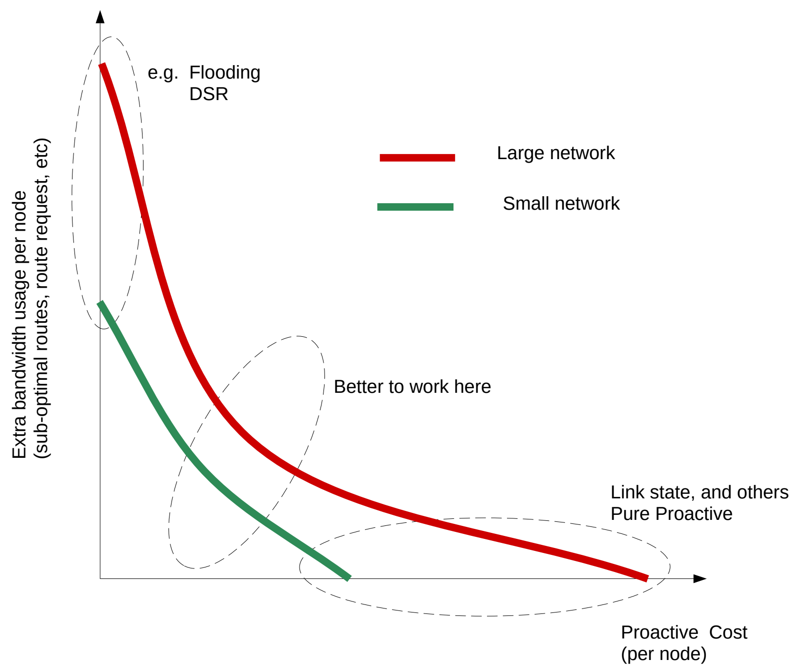

Figure 4: Change in achievable region due to size.

the network. Since both the rate of generation of RREQ and the number of retransmissions required by each RREQ message increases linearly with N, the total overhead of a purely reactive protocol (the point R) increases as rapidly as N^2.

On the other hand, the overhead of protocols at 'intermediate points', such as HierLS and ZRP, may increase more slowly with respect to N. In [11] it is shown that under a reasonable set of assumptions HierLS's and ZRP's overhead grows with respect to N is 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 $\Theta(N^{1.5})$. [4] Referring again to Figure 4, 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 in the "middle" region where the proactive, reactive, and suboptimal routing overheads are balanced.* One might

[4]Standard asymptotic notation is employed. A function $f(n) = \Omega(g(n))$ [similarly, $f(n) = O(g(n))$] if there exists constants c_1 and n_1 [similarly, c_2 and n_2] such that $c_1 g(n) \leq f(n)$ [similarly $f(n) \leq c_2 g(n)$] for all $n \geq n_1$ [similarly, $n \geq n_2$]. Also, $f(n) = \Theta(g(n))$ if and only if $f(n) = \Omega(g(n))$, and $f(n) = O(g(n))$.

reasonable argue that in order to achieve high scalability, one should operate in the intermediate region where suboptimal routes are present. In other words, suboptimal routes are a fact of life for ultimate scalability.

3.5 A Formal Definition of Scalability

As mentioned before, there is no established scalability metric for ad hoc networks. In this chapter we will follow the definitions and framework presented in [11] since they provide us with tractable models that capture the bandwidth aspects of routing protocol's scalability and properly distinguish a routing protocol scalability properties and limits from the scalability properties and limits inherent to the network (independent of the routing algorithm – if any[5] – run over it).

The key idea here is to separate out the concepts of *network* scalability and *protocol* scalability. Some networks are inherently unscalable and some are inherently scalable. It is only for the latter class that demanding protocol scalability – as traditionally understood the term – makes sense. We elaborate on this below.

Let's start by the intuitive definition of scalability:

Definition 3.3 Scalability *is the ability of a network to support the increase of its limiting parameters.*

Thus, scalability is a property. In order to quantify this property, the concept of *minimum traffic load* presented in Subsection 3.3, definition 3.1, is used to define the *network scalability factor*:

Definition 3.4 *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 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 :*

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

The *network scalability factor* is a number that asymptotically relates the increase in network load to the different network parameters. For example,

[5]Since the concepts are generic, they cover even the case of static networks which may use static, pre-defined routes.

let's consider the most efficient wireless ad hoc networks : the class of topology controlled ad hoc networks. For this class of networks the *minimum traffic load* $Tr(\lambda_{lc}, \lambda_t, N)$ as a function of the per node rate of link change λ_{lc}, per node traffic λ_t, and network size N is $\Theta(\lambda_t N^{1.5})$, [6] 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 3.5 *The* **network rate** R^{net} *of a network is the maximum number of bits that can be simultaneously transmitted in a unit of time. For the* **network rate** *(R^{net}) 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 3.6 *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^{net}) can support. That is, if and only if:*

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

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 [27, 28]). The class of ad hoc networks considered before (i.e. resulting from applying power control techniques) are precisely the class of networks that achieves that maximum *network rate*. Thus, in order for this class of ad hoc network to be regarded as scalable with respect to network size, we would need that $\Psi_N \leq 1$. Unfortunately, this is not the case (recalling that for these networks $\Psi_N = 1.5$) and as a consequence this (wide) class of ad hoc networks are not *scalable* with respect to network

[6] 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, L, the average path length, is $\Theta(\sqrt{N})$, the above expression was obtained in [11].

size. [7] Fully connected wireline networks, on the other hand, exhibit a *network scalability factor* $\Psi_N = 1$ while its network rate R^{net} increases as fast as $\Theta(N^2)$, and therefore they are scalable with respect to network size (in the bandwidth sense). Note, however, that this scalability requires the nodes' degree to grow with the network size which may become 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. Again, considering topology-controlled ad hoc networks we notice that they are *scalable* w.r.t. mobility ($\Psi_{\lambda_{lc}} = 0$), but are not *scalable* w.r.t. traffic ($\Psi_{\lambda_t} = 1$).

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 further considered in this chapter. For example, as transmission range increases (and assuming an 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. 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 : what does it mean for a routing protocol to be *scalable*?. The remaining of this subsection will clarify this meaning.

Definition 3.7 Routing protocol's scalability *is the ability of a routing protocol to support the continuous increase of the network parameters without degrading network performance.*

In other words, the scalability of a routing protocol is dependent on the scalability properties of the *network* the protocol is running over. That is,

[7] It has been shown in [28] 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.

the network's scalability properties provide the reference level as to what to expect of a routing protocol. If the overhead induced by a routing protocol grows faster than the *network rate* (eventually depleting the available bandwidth) but slower than the *minimum traffic load*, the routing protocol is not degrading network performance, which is being determined by the *minimum traffic load*. Roughly speaking, if a type of network can handle thousands of nodes, then an *scalable* routing protocol for this type of networks should be able to run over the thousand-node network without collapsing. But, if the network can only handle hundreds of nodes, the fact that a routing protocol collapses when run over thousand of nodes does not mean that the routing protocol is not scalable for this type of network. The routing protocol is not degrading network performance at the 100-node level. Performance is being dominated by the network limitations (*minimum traffic load* versus *network rate*). There is no point in requiring the routing protocol to operate at a point where the network collapses on its own!.

From the above discussion it is clear that a routing protocol may be deem to be scalable or not only in the context of the underlying network the protocol is running over. This chapter covers the scalability of routing protocol running over wireless ad hoc networks; specifically the (wide) class of networks defined by assumptions a.1-a.8.

To quantify a *routing protocol scalability*, the respective scalability factor is defined, based on the *total overhead* concept presented in Subsection 3.3, definition 3.2, as follows:

Definition 3.8 *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:

Definition 3.9 *A routing protocol X is said to be* **scalable** *with respect to the parameter λ_i if and only if, as 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:*

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

Thus, for the class of topology-controlled ad hoc networks, a routing protocol X is *scalable* with respect to network size if and only if $\rho^X_N \leq 1.5$; it is *scalable* w.r.t. mobility rate if and only if $\rho^X_{\lambda_{lc}} \leq 0$; and it is *scalable* w.r.t. traffic if and only if $\rho^X_{\lambda_t} \leq 1$.

Using the above definitions, we are now ready to assess the scalability of protocols described in Section 2.

4 Results on Scalability of Ad Hoc Routing Protocols

In this section, we present an overview of the main results for bandwidth-related scalability for the representative set of routing protocols described in Section 2. The interested reader is referred to [22, 11] for more information about the derivations.

4.1 Scalability Dimensions

Scalability is often interpreted as the ability to handle increasing *size.* While the size of an ad hoc network is a key parameter affecting the scalability, it is by no means the only one. Other scalability dimensions include mobility (for *mobile* ad hoc networks), network density, network diameter, traffic diversity, energy etc. These parameters may influence the design of the network control mechanisms at various layers. For instance, an increase in the diameter of a network implies a higher latency for control information propagation, leading to a greater risk of inconsistent routes and instability. Similarly, an increase in density results in decreased spatial reuse of the spectrum and consequent reduction in capacity.

Figure 5 shows some key scalability dimensions and their effect on the lower four layers of the ad hoc network stack. Different protocols may exhibit different levels of scalability with respect to each of these dimensions, and an understanding of this is essential to an informed choice of a protocol for a given application.

Out of the different parameters shown, we observe that size, density, diameter, and transmission range (not shown) are related. For a given net-

dimension layer	Size	Mobility	Density	Diameter
Transport		X		X
Network	Ⓧ	Ⓧ	X	X
Link/MAC		X	X	
Physical		X		

Figure 5: Scalability dimensions and the layers. An ‘X’ indicates that the scalability problem involving the dimension representing the column manifests itself at the layer representing the row. A circle around an ‘X’ indicates combinations that we address in this chapter.

work size and density, different transmission power levels will result in different combinations of node degree and network diameter (longer transmission range will result in higher node degree and smaller network diameters). The state of the art in the area of topology control for ad hoc networks provides effective algorithms which adjust the transmission power in order to obtain more advantageous topologies. It is well understood that in order to increase the overall network performance, the average node degree must remain bounded except when required to improve connectivity (a reasonable goal is to have a biconnected network). Thus, the density dimension can be addressed by means of effective topology control algorithms (see for example [29]). For this reason, in the remainder of this chapter we will consider topology-controlled networks where the density is not a limiting factor and where the network diameter and size are mutually dependent. Thus, we will only consider the network size and mobility dimensions in our discussion of scalability. Of course no treatment of scalability would be complete without addressing the third scalability dimension (not shown in the figure for being self-evident at every layer) : traffic load.

4.2 Network Model

In order to obtain concrete, closed-form expressions for total overhead induced by the representative set of protocols described in Section 2 it is necessary to refine the class of network under study. This refinement was done favoring the most common, but challenging types of networks. Alternatively expressions can be derived for other classes of network if they are of interest.

However, in order to maintain focus and obtain the desired insight, in the remainder of this chapter we will follow the work in [11] and will restrict our attention to the (broad) class of networks defined by the assumptions presented below.

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 and the availability of desirable topology control techniques, define the kind of scenarios under consideration:

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, 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 $g_{t/t_1}(x, y, x_{t_1}, y_{t_1}) = \frac{1}{(t_1-t)^2} g_{0/1}(\frac{x-x_{t_1}}{t_1-t}, \frac{y-y_{t_1}}{t_1-t})$.

For a discussion on the rationale behind these assumptions (besides the existence of an underlying topology control mechanism) the reader is referred to [22, 11].

4.3 Asymptotic Behavior of Ad Hoc Routing Protocols

Table 1 shows asymptotic expressions for the proactive, reactive and sub-optimal routing overhead (in bps) for the protocols described in Subsection 2 when run over the (wide) class of networks determined by assumptions a.1 through a.8.

PF induces no proactive or reactive overhead. But each packet generated (there are $\lambda_t N$ such packets per second) is flooded to the entire network (retransmitted N times), and therefore its sub-optimal routing overhead is linearly dependent on the traffic rate and on the square of the network size.

SLS builds optimal routes proactively, so there is no reactive or suboptimal routing overhead associated with it. Each time there is a link change ($\lambda_{lc} N$ times per second) an LSU is flooded throughout the entire network (N retransmissions) resulting in a proactive overhead that increases linearly with the rate of per node link changes and the square of the network size.

DSR-noRC has no proactive component. Its reactive overhead is lower bounded by the overhead induced by the route discovery procedures in response to new sessions ($\lambda_s N$ new sessions per unit of time). DSR-noRC

Protocol	Proactive Overhead	Reactive Overhead	Suboptimal Routing Overhead
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 1: Asymptotic results for several routing protocol for mobile ad hoc networks.

reactive overhead is upper bounded by assuming that each link change will trigger a new route maintenance procedure and that each route maintenance procedure will cause a global flooding (i.e. local repair did not succeed). The combined effect of these assumptions is that each link change event has the same effect as a new session event and therefore the combined rate of events – $(\lambda_s + \lambda_{lc})N$ new sessions plus link changes per second – results in the upper bound shown in table 1. Finally, DSR-noRC sub-optimal routing overhead's lower bound shown in Table 1 was derived by considering the extra bits $\Theta(\sqrt{N} log_2 N)$ required to add the source route to each packet. Recall that $log_2 N$ bits are required to specify a node address, and that the average route length is $L = \Theta(\sqrt{N})$.

Regarding HierLS, depending on the location management approach being used, HierLS may or may not induce reactive overhead. For example, if the location management approach requires that a node pages one or more location servers in order to find the current location of a destination and be able to build routes towards him, then the paging packet(s) will contribute to the reactive overhead. Table 1 shows HierLS's overhead results when a pure proactive location management is employed and therefore there is no reactive overhead associated with HierLS. In a pure proactive location management scheme (referred to as LM1 elsewhere in this chapter) each node S has a local copy of a location table where there is a map between every node in the network and the highest level cluster that contains the node but does not contain node S. The advantage of such a scheme is that there is no location server that may constitute single-point-of-failure for the entire network.

For HierLS-LM1, table 1 shows that there is no reactive overhead and that the proactive overhead is dominated by the location update cost $\Theta(sN^{1.5})$ – where s is the average node speed – which is far greater than the LSU propagation cost $\Theta(\lambda_{lc}N)$. This shows that in HierLS, for higher levels in the hierarchy, it is more likely to have cluster membership changes (due to node movements being unrelated to the cluster selection) than it is to have virtual link changes (since individual link changes get buffered out by the large number of links forming a virtual link). The sub-optimal routing overhead is determined by observing that for a fixed number of hierarchical levels, the average path length is a percentage above the optimal path length, and therefore the sub-optimal routing overhead was proportional to the actual rate of traffic transmission (hop-by-hop, not source-destination)i of $\Theta(\lambda_t N^{1.5})$. Further observing that the percentage of sub-optimality of the routes increased with the number of hierarchical levels, which in turn increased with the network size, determined the inclusion of the value δ (a small constant value dependent on the number of nodes in a cluster) in the sub-optimal routing expression.

For ZRP, the proactive overhead is dependent of the size of a node's *zone* (n_k). The bigger the zone the larger number of proactive control message that will need to be retransmitted. The reactive overhead, on the contrary, will decrease with the zone size. However, the reactive overhead does not vary inversely proportional to the *zone* size n_k but it depends on the square root of it. To understand this, consider that increasing the zone size will increase the area that each border node 'covers'. The larger the zone size the smaller the number of border nodes required to cover the entire network will be. Even though the border node zones are overlapping, still it is true that the number of border nodes required will be $\Theta(N/\sqrt{n_k})$. Thus, as the zone size increases a source node S in ZRP will need to poke a smaller number of 'border' nodes. However, the distance between border nodes also increases (in average $\Theta(\sqrt{n_k})$) resulting in the expression shown in Table 1. For ZRP's sub-optimal routing overhead an upper bound is provided that shows that ZRP's sub-optimal overhead does not affect ZRP's total overhead expression since it (sub-optimal routing) is (asymptotically) dominated by the reactive overhead. The sub-optimal routing upper bound is derived by considering the maximum route length after subsequent local repair procedures for long lived sessions. Since two border nodes, non-adjacents, in a

ZRP's source route may not belong to each other zone (otherwise the route may be shortcutted) then the number of border nodes in a path is at most N/n_k. Since border nodes are $\Theta(\sqrt{n_k})$ hops away from each other, then the maximum length of a packet roue to its destination is $\Theta(N/\sqrt{n_k})$, resulting in the expression shown in Table 1.

For HSLS, it is obvious that the proactive overhead will be inversively proportional to the LSU generation period $1/t_e$. However, the dependency on network size is not so obvious. t can be understood if we consider that in HSLS, the proactive overhead is dominated by the global LSUs, that is, LSUs that traverse the entire network. The next type of LSU in order of importance for the proactive overhead computation is the LSUs with the next higher time-to-live field. This is due to the fact that decreasing the time-to-live field by a factor of 2 reduces the number of LSU retransmissions by a factor of 4 while only increasing the frequency of transmissions by a factor of 2, providing a combined effect of reducing the proactive overhead induced by these LSUs by a factor of 2. When considering the global LSUs, we may notice that their generation rate according to HSLS rules is inversely proportional to the network diameter, which is $\Theta(\sqrt{N})$, resulting in the expression on Table 1

HSLS has no reactive overhead. Determination of HSLS sub-optimal routing overhead is not trivial, but the result can be understood by considering that for a given mobility rate and generation period, the probability of making an erroneous next-hop decision is bounded independently of the distance to the destination. This independence of the distance to the destination is a consequence of the LSU generation/propagation mechanism used in HSLS, which imposes a quasi-linear relationship between distance and routing information latency. Thus, the ratio of information latency (related to position uncertainty) over distance is bounded and so is the uncertainty about the destination angular position, which is the only critical information required to make a best next hop decision. Finally, having a bounded probability of a bad next hop decision, regardless of distance, ensures that the paths built by HSLS are just a fraction from optimal. This fraction depends on λ_{lc} and t_e as shown in Table 1.

As we may see from Table 1 and the above discussion, ZRP and HSLS behavior depends on configurable parameters. Table 2 shows the *total overhead* obtained when the routing protocol parameters are chosen to optimize performance. Values on Table 2 represents the best each protocol can do.

These asymptotic expressions provide valuable insight about the behavior of several representative routing protocols. They help network designers

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

to better identify the class of protocols to engage depending on their operating scenario. For example, if the designer's main concern is network size, it can be noted that HierLS and HSLS scale better than the others. Moreover, by observing the asymptotic expressions we may notice that when information dissemination (either link state, route request, or data itself) is flood to the entire network, the routing protocol scalability factor with respect to network size is equal to 2. Splitting the information dissemination at two different levels, like in 2-level hierarchical routing, NSLS, ZRP, and DREAM, can achieve a reduction in the routing protocol scalability factor down to 1.66. Allowing the number of levels of information dissemination grow as required when the network size increases, as done explicitly by m-level HierLS and implicitly by HSLS, can further achieve a reduction of the scalability factor down to 1.5, which seems to be the limit on performance for routing protocols for ad hoc networks defined by a.1 through a.8.

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. ZRP scales well with respect to traffic load since it can adapt its zone size, increasing it to the point that ZRP's behaves a a pure proactive algorithm (e.g. as SLS). HSLS scales better than PF, DSR, and HierLS since as traffic load increases, HSLS increases the value of its LSU generation rate ($1/t_e$), which causes more LSUs to be injected into the network, reducing routing information latency and improving the quality of the routes. This points out two observations: (1) as traffic load increases, the quality of the routes becomes more and more important; (2) as traffic increases, more bandwidth should be allocated for dissemination of routing information, so that the quality of the routes are improved. **The second observation contradicts the widely held belief that as traffic load is increased, less bandwidth should be allocated to control traffic and let more bandwidth available for user data.**

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 overheads* 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. However, 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 according to the scenario the designer is interested in.

Comparing HSLS and HierLS results, it is counter-intuitive to observe that HSLS –a flat, relatively easy to implement protocol – has better asymptotic properties than HierLS with respect to network size. This means that as size increases HSLS eventually outperforms HierLS. **This contradicts the widely held belief that as size increases the only routing solution is to shift from a flat to a hierarchical paradigm**. However, this section discussion suggests that building/maintaining/managing a complex routing hierarchy (a potential implementation nightmare) may not be necessary. The next section goes deeper into this issue, comparing flat and hierarchical routing techniques for ad hoc networks, trying to answer the question *which is better: flat or hierarchical?*.

5 Flat vs. Hierarchical Routing

Perhaps the most significant result from previous sections is this: a protocol that restricts the scope of control messages and takes the penalty of sub-optimal routes is more scalable than one that insists on “full information”. For instance, as the network size grows while the available bandwidth remains fixed, traditional routing protocols such as SLS and DV quickly collapse, since they waste all the available bandwidth in disseminating routing protocol control messages. On the other hand, all of the protocols with sub-exponent-2 asymptotic scalability (ZRP, HierLS, FSLS) are “limited information” protocols.

Within this class of limited information protocols however, it appears that one can achieve the scalability goal using either a “flat” routing approach or a hierarchical approach. The question then is: *Is the hierarchical approach better or flat?* Or, more specifically: *Under which circumstances does a flat approach outperform a hierarchical one?*

In this section, we compare and contrast the two approaches, bringing out the advantages and disadvantages of each. We take a representative protocol from each class – namely HSLS for flat, and HierLS for hierarchical – for a direct comparison using simulation. In a sense then, this section is the “finals” of a tournament, where all but the best have been eliminated and the interest zooms in on the two best.

This section is organized as follows. In the first subsection we present a taxonomy of the hierarchical approaches, and comment on the different subclasses from a bandwidth-scalability and implementation-complexity perspective. In the second subsection, we describe the main techniques for scaling flat routing protocols. Emphasis is on the most promising class of techniques, the FSLS family of algorithms, particularly the optimal algorithm in this class, namely the HSLS algorithm.

We then proceed, in the third subsection, to compare hierarchical and flat routing. In order to obtain concrete results, a representative protocol for each routing approach is chosen. HSLS, the best algorithm in the FSLS family, is picked as the representative of flat routing techniques. A highly efficient m-level hierarchical routing protocol based on MMWN is chosen as the representative of hierarchical routing. This protocol, while using the *virtual gateway* abstraction, sets the cost of each virtual link at a given hierarchical level to the same value, behaving as if the *virtual node* abstraction was used. Thus, this protocol behaves as belonging to the class of HierLS routing algorithm. This protocol was chosen since it presents good scalability properties without demanding an unreasonably high implementation cost.

We present a simulation study under moderate stress conditions in order to capture behavior that could be overlooked by the theoretical analysis. By virtue of the simulations, practical issues affecting hierarchical and flat routing protocol performance differently are identified. The level of incidence of these issues in a given network may shift the relative performance of the hierarchical and flat routing techniques. Finally, the last subsection discusses our conclusions from the comparison.

5.1 Hierarchical Routing Techniques

The core of a hierarchical algorithm consists in aggregating nodes into (level-1) clusters, clusters into superclusters (level-2 clusters) and so on. This grouping of nodes allows for an abstraction of the routing information. For example consider the HierLS algorithm presented in subsection 2.4. In HierLS a node may consider all clusters (level-1 and up) as virtual nodes in a virtual topology. In such a topology the set of links between two such clusters conform a virtual link. In such case, individual link variations will have a small impact in the virtual (aggregated) link state. If the routing algorithm restricts the generation of updates such that only changes above a predetermined threshold trigger updates, then the rate of updates sent as a response to virtual links' changes is significantly reduced.

It should be noted that the virtual topology may be built in a different manner. For example, MMWN [15] chooses the set of links between two clusters (namely virtual gateways) to become the virtual nodes. MMWN then chooses the set of nodes inside a cluster, needed to traverse from a virtual gateway to another in the same cluster, as the virtual link.

Whatever the definition of the virtual topology is, the main characteristics of a hierarchical approach is:

- Nodes are grouped in clusters. Clusters in Superclusters, and so on. Each cluster defines a cluster leader for coordinating functions.

- Information about nodes far away is aggregated, resulting in smaller memory requirements for storing topology information, lower processing requirements to build routes, and lower bandwidth requirements for propagating traditional (i.e. excluding location management) routing information updates.

- Due to mobility, a location management scheme is required. Note that this is a main difference between hierarchical schemes for fixed (e.g. IP networks) and mobile (e.g. ad hoc) networks.

Hierarchical approaches may then be classified by its cluster and cluster leader selection algorithm; by the abstraction used to map virtual nodes and virtual links to the actual elements of the physical world; and by the location management technique being used. In the next subsections, we present a quick overview of the current techniques used for hierarchical routing. Readers interested in a more extensive treatment of the subject will find reference [30] to be an excellent starting point.

5.1.1 Cluster and Cluster Leader Selection Methods

Clustering techniques may be classified by the radius of the cluster formed; by the affiliation method used; by the objective (gain) function used in the affiliation method; and by the cluster leader selection method used.

Cluster Radius
There is a class of clustering techniques that impose the restriction that the cluster radius (distance, in hops, from the cluster leader or center to any other node in the cluster) be at most 1 (e.g. LCA[13], CGSR[14], and ARC[16]). Thus, two cluster leaders belonging to neighboring clusters will be at most 2 hops away. The intermediate node in the 2-hop path is called a gateway node. The advantage of this kind of clustering techniques, especially if only two levels are being formed, is its simplicity. There are efficient algorithms that only require local (i.e. one-hop) information in order to make clustering decisions. These kind of techniques, however, will result in a large number of level-1 clusters. And, if higher level clusters are to be constructed by the same procedure, one finds that most of the simplicity advantage is lost. Thus this method is not particularly scalable. This technique is also used to build clusters for purposes other than routing: control of access to the shared medium (i.e. scheduling transmissions), efficient flooding of information (including routing related information as for example LSUs), etc.

The other class of clustering techniques does not require the cluster leader to be in direct communication (e.g. one hop away) from the cluster leader (e.g. HierLS[11], HSR[12], and MMWN[15]). Still, usually they impose a maximum limit on the distance between a cluster leader and nodes in the cluster boundaries, mainly for performance reasons. The HierLS algorithm presented in Subsection 2.4 belongs to this class. An advantage of this class of clustering techniques is that the clustering size (as well as other parameters) can be adjusted to optimize performance. For example, if a 2-level network must be formed out of 10000 nodes, a good clustering technique will result in roughly 100 clusters of 100 nodes each. Of course, for this cluster size, it will be even better to increase the number of levels in the network (although it should be kept in mind that building and maintaining a 3-level hierarchy is as complex as a m-level hierarchy, which is much more complex than a 2-level one).

Cluster Affiliation Method

Cluster affiliation refers to the way nodes are assigned to clusters. In some techniques, this decision is left to the node itself. In others, it is the cluster leader which assigns the nodes to its cluster.

The main advantage of leaving the 'joining' decision to the nodes, is that it allows for distributed algorithm implementation. On the other hand, the lack of a centralized control, and the latency in propagating control information may result in unpredictable dynamics causing, for example, cluster size to increase to unacceptable levels. This in turn may induce the splitting of a cluster, which may result (again due to information propagation latency) in a cluster that is too small and nodes rejoining the cluster, etc. In general, clustering affiliations where each node makes its own decisions are more susceptible to instabilities.

The other approach is to let the cluster leader to make the clustering assignments (it may 'grab' a set of nodes or may assign nodes previously in its cluster to another clusters). This approach may require the cluster leader to collect information about nodes more than one hop away, in order to decide which nodes to 'grab'. Also, the leader may decide to sequentially 'grab' nodes in the boundary of the cluster (resembling the dynamics of the previous technique when nodes 'join' the cluster) or it may grab a large set of nodes at once. The latter will speed up convergence time but require the leader to have up-to-date information about nodes outside of its cluster.

Performance objective

Clustering techniques may also be classified by the performance objective they target. Although one may expect that throughput or a routing performance metric be the goal of every clustering techniques, in reality the difficulty of mapping clustering parameters into actual routing performance metrics results in different hierarchical schemes targeting intermediate goals that are *suspected* to have a positive impact on performance.

Some protocols target clusters with balanced size. For example, MMWN defines a minimum and a maximum size for a cluster, and engages 'join' or 'split' procedures if these boundaries are crossed.

Other protocols target a desired level of connectivity inside the cluster (i.e. that the nodes inside the cluster form a k-connected) set. Similarly, the objective may be maximize connectivity of the nodes forming a virtual gateway (see MMWN [15]). The idea in targeting k-connectivity is to avoid the cluster to become partitioned in the near future. K-connectivity provide alternative paths in case of link failures. The idea of maximizing path availability inside a cluster is further explored in [31], where the authors propose a cluster formation technique (the (α, t) clustering) that targets the formation of clusters such that the probability that there will always be a path between two nodes inside the cluster for the next t seconds is at least α. The (α, t) clustering technique is mobility adaptive. Since a path availability is the product of the availabilities of the links forming the path; then the longer the path the lower the availability. Thus, higher speeds (and consequently smaller link availability) will result in smaller cluster diameter (and size). Lower speeds (and therefore lower link volatility) will allow the cluster diameter (and size) to grow.

Other performance objectives include a minimum level of 'affiliation' between a candidate node and the cluster. The 'affiliation' may be defined as the composite bandwidth between the node and all other nodes in the cluster; the distance to the cluster leader; a measure of similarity between the candidate node and the node inside the cluster (based on pre-assigned, task dependent role); or a linear combination of all the above.

Cluster leader selection

Clustering techniques for homogeneous networks usually do not distinguish between individual nodes, and therefore the identity of the cluster leader is not relevant. These techniques, however, require the leader selection to be unique, and therefore they need a common criteria for determining the cluster leader or a mechanism to solve conflicts if they occur.

A usual common criteria for cluster leader selection consists of picking the node with the lowest id among its unclustered neighbors. Since the id of the nodes do not follow any rational order, this amounts to having a random leader selection technique. Actually, there are clustering techniques where the selection of the leader is explicitly made at random (e.g. NTDR [32]).

At the other hand there are clustering techniques where the leader selection is preassigned, based on additional knowledge about the scenario. For example, LANMAR [33] preassigns the cluster leaders based on knowledge of the mobility patterns of the nodes. In LANMAR the nodes are assumed to exhibit group mobility, and the group leaders are selected as cluster leaders. Similarly, extra knowledge about a node capabilities: battery power, extra bandwidth, low mobility/high stability, extra processing power, susceptibility to destruction, mission role, etc. may be used in pre-determining the identity of the cluster leader.

Another cluster leader selection technique is based on picking as leader the node that maximizes a gain function among all the other nodes in its cluster (which initially may just be its 1-hop neighbors). A good gain function to maximize is the node degree, since a cluster leader with a higher degree assures that the cluster leader will likely remain connected to the cluster nodes over time. Besides, high degrees are usually associated with advantaged nodes (e.g. higher power, higher elevation, etc.). Even if there are no advantaged nodes, picking nodes with higher degree will result is cluster with smaller diameter, which improves performance. An obvious extension of this criteria is to define the gain function to be equal to the number of k-neighbors of a node, where k is the expected radius of the cluster. These gain functions, however, consider the network topology as something static, and therefore may choose the cluster leader (and the cluster around him) that is appropriate for a short period of time. Thus, a better gain function should take into account (as much as possible) node mobility patterns, and based on this knowledge pick up as cluster leaders those nodes that will maximize the expected number of k-neighbors over time. SOAP [34, 24] is an example of an algorithm implementing such a gain function. It should be noted that cluster (leader) selection techniques that take into account the mobility patterns as in LANMAR and SOAP has the potential to reduce or even eliminate the location management cost associated with hierarchical routing if the nodes mobility presents strong patterns, such as group mobility. As we saw in the previous section the location management cost dominates the link state information dissemination cost for HierLS approaches, so reducing this former cost will greatly improve performance and may even enable us to improve HierLS asymptotic performance, by trading off an increase in the (now small) proactive overhead for a reduction in sub-optimal routing overhead, resulting in a smaller combined total overhead.

Finally, the gain function may also be a weighted combination of the aforementioned quantities, plus additional quantities such as available power, processing speed, memory available, role, vulnerability, etc. that need to be pre-configured in each node.

Thus, there is a plethora of criteria for cluster formation. This diversity is symptomatic of our lack of understanding of the dynamics involved in clustering formation and maintenance and its impact in the generation of link state and location management information, the generation of clustering management messages, and the transient latencies incurred due to the handling of exception situations (e.g. a cluster leader is destroyed or is temporary partitioned from the cluster). The obscure nature of the impact of clustering in network performance has been the main obstacle to the design of highly efficient hierarchical algorithms.

5.1.2 Topology Abstraction Methods

Once the network nodes are organized in the clustering hierarchy, this structure is used to reduce the topology information that needs to be propagated inside the network. However, different techniques may be employed.

The HierLS algorithm presented in Subsection 2.4 is an example of a hierarchical system using the *virtual node* abstraction. In the virtual node abstraction, level-m clusters are considered level-m nodes. Real nodes are considered level-0 nodes. The set of links connecting real nodes in neighboring level-m cluster forms a level-m virtual link. A node keeps track of all the level-m virtual node and virtual links inside its level-$m+1$ cluster. Thus, routing information is reduced since a node does not need information about level-m virtual nodes outside its level-$(m+1)$ cluster. Subsection 2.4 presents a more detailed explanation of routing using the virtual node abstraction.

MMWN[15] is a protocol that uses the *virtual gateway* abstraction instead of the virtual node abstraction. To illustrate the way the virtual gateway abstraction works, consider a network formed by four clusters A, B, C, and D aligned horizontally as follows: $A - B - C - D$. A $-$ between B and C, for example, represents that there are some (physical) links connecting (physical) nodes in cluster B with nodes in cluster C. In the virtual gateway abstraction, each set of links connecting different clusters is called a virtual gateway and constitute level-1 nodes. Thus, at the level-1 the aforementioned network has three nodes: $A.B$, $B.C$, and $C.D$. Now, the level-1 link joining, for example, nodes $A.B$ and $B.C$ is formed by an aggregation of all the paths from nodes in $A.B$ to nodes in $B.C$. For example, if the link metric of interest is available bandwidth (for QoS-based routing), then this level-1 link metric is not associated with the number of nodes and links inside node B, but with the maximum flow from $A.B$ to $B.C$. Similar aggregation may be achieved if the link metric of interest is delay, etc.. Similarly, virtual gateways among level-2 cluster constitute level-2 nodes and aggregation of paths between these virtual gateways constitute level-2 links, and so for. Route computation is performed almost as in HierLS, with the difference being that the objective is to find a virtual gateway neighboring the destination cluster, as opposed to looking for the destination cluster itself. For example, if a node inside cluster A in the above example is looking for a node inside cluster D, its Dijkstra's computation will stop when the virtual gateway $C.D$ is found. The route obtained will be $source -$ level-0 nodes $- A.B - B.C - C.D$, instead of $source -$ level-0 nodes $- B - C - D$ which would be the case if the virtual node abstraction were used. Intermediate nodes in the path will expand the route as necessary, similar to the virtual node abstraction case (e.g. HierLS).

The virtual node abstraction is more intuitive and therefore easier to analyze, implement, and debug. However, if QoS constraints are to be satisfied (as for example a minimum required bandwidth) the virtual gateway abstraction provides better link information aggregation. In the virtual node abstraction, clusters won't be able to properly estimate the virtual link cost because: (1) virtual links include links in two different clusters, and a cluster only has information about link inside itself, thus it only has information about half the link. And (2) the cost of traversing a cluster is dependent on the entry and exit points. For example, in the case of the $A-B-C-D$ network discussed before, the cost of going from A to C depends on the cost of traversing cluster B having $A.B$ as an entry point and $B.C$ as an exit point. The virtual node abstraction will estimate this cost as the sum of $A-B$ and $B-C$, where the cost of $A-B$ is computed without knowledge of the next link in the path (i.e. $B-C$) resulting in a lower quality estimate. Roughly speaking, the virtual node abstraction's link cost estimates will – at best – be equivalent to assuming that all paths go through the cluster leaders, which is a bad estimate. Thus, in general, the virtual gateway abstraction will produce routes of better quality than the virtual node abstraction. Of course, the price we pay is the extra complexity in maintaining the virtual gateway structure in addition to the clusters: some node inside the virtual gateway must to be chosen as leader and should propagate link state updates with the latest (virtual) link cost.

Besides the virtual node and virtual gateway abstractions, other techniques to exploit the hierarchical structure formed include the *quasi-hierarchic algorithm*[35, 39] and *Landmark routing*[36]. Both techniques try to maintain optimal (or good) paths toward higher level clusters. Therefore, some link changes may result in network wide propagation of updates. Thus, if propagation is event driven, these updates result in higher control overhead consumption. On the other hand, if information propagation is done periodically, the effect of these long-impact changes is long latency in routing information propagation which results in poor response to network dynamics.

5.1.3 Location Management Methods

The core of hierarchical routing consists on aggregating information by efficiently using the clusters built. Therefore, a node no longer has complete information about how to reach a node outside its level-1 cluster. To determine how to route packets to nodes outside its (level-1) cluster, a node needs to know the identity of a cluster associated with the destination. The service that provides the nodes with this information is referred to as *Location Management (LM)*. The need of a *LM* service is a main difference between hierarchical approaches for static (wireline) and mobile networks. In static networks, a LM service was not needed since the address of a node was tied to its location in the hierarchy. Due to mobility, this is no longer the case.

The LM service can be implemented in different ways, whether proactive (location update messages), reactive (paging), or a combination of both. 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. As an example let's consider Figure 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 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.

For example, if node n_2 in Figure 1 moves inside cluster $X.1.5$, then the level-2 location server of cluster $X.1$ will be notified, who in turn will notify the location servers of clusters $X.1.1$ through $X.1.7$. Alternatively, if node n_2 had moved inside cluster $X.4$ instead, then the location server of cluster X would have been notified, and he in turn would have trigger notifications to all level-2 and level-1 location servers inside cluster X. And so on.

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.e. 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. For example, if node n_2 in Figure 1 moves inside cluster $X.1.5$, then the location server of that cluster will notify the level-2 location server of cluster $X.1$. Additionally, the location server of cluster $X.1.1$ will also notify the level-2 location server that node n_2 does not belong to that cluster anymore. No other location server will be notified. Alternatively, if node n_2 had moved inside cluster $X.4$ instead, then the location servers of clusters $X.1.1$, $X.1$, and X would had been updated. Location servers on clusters $X.1.1$ and $X.1$ would learn that node n_2 does not belong to their clusters anymore, and the location server of cluster X would know that node n_2 belonged to cluster $X.4$.

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.

The LM1 technique is the simplest of the three, but it may consume significant bandwidth for propagating location update messages. Technique LM2 reduces the bandwidth consumption for reasonable rates of new session (requiring a local page to the local location server) arrivals but at the expense of complexity (selection and maintenance of LM servers) and an increase in the latency for route establishment. However, the asymptotic characteristics of the hierarchical protocol do not change whether we use approach LM1 or approach LM2[11, 24].

Approach LM3 is the more complex to implement and analyze. It will induce a fair amount of reactive overhead (susceptible to traffic), but will significantly reduce the amount of overhead induced by mobility. However, it is expected that the bandwidth consumption of approach LM3 is the smallest of the three for typical operating conditions. The price we pay is increased latency when building new routes, a high paging cost under high traffic load and diversity, much higher implementation complexity, and network susceptibility to single points of failure.

To summarize, we observe that there are a large number of variants of hierarchical routing. Each variant represents a different trade off between complexity and performance. We will expect the more complex approaches to present better performance. However, due to the unpredictable nature of the hierarchical routing dynamics, we can not be sure of this until after analyzing the protocol through extensive simulations. Thus, it is not clear until after a protocol has been designed, debugged, and tested whether or not the extra complexity has a payoff. This points out the need of theoretical models of performance. For example, from the results shown in Table 2, we get the insight that jumping from a 2-level hierarchy to a m-level hierarchy (not a small jump in implementation complexity) will allow us to reduce the protocol scalability factor with respect to network size from 1.66 to 1.5. Whether this reduction justifies the extra complexity will be a decision that the designer will make based on his perception of how large a network the protocol is intended to support.

Finally, the experience of working with hierarchical routing approaches, especially its high degree of complexity, has motivated a renovated interest for alternative approaches. Thus, there has been a surge of research for efficient flat routing algorithms whose performance (with respect to increase to network size) may be competitive (under a cost-benefit analysis) with hierarchical approaches. The next subsection presents a survey of these techniques. Some of them, specifically HSLS, has been shown to have better asymptotic scalability properties than some hierarchical algorithms (see Table 2).

5.2 Flat Routing Techniques

The term "flat routing" is used to contrast basic routing techniques from hierarchical routing applying a topology abstraction. Unlike hierarchical routing, there are no "boundaries" imposed between groups of nodes, nor is there an addressing scheme based on hierarchy.

In flat routing, then, there are no abstractions and no virtual nodes or links. Each node and link in the topology table of a flat algorithm represents an actual (physical) node or link. Thus, the topology table may grow large as the network size increases. However, in a flat routing scheme we do not need *all* the nodes and links be present in the topology table. Specifically, some links may be hidden if they are not expected to affect a node's route computation. Similarly, nodes may not be included in the topology table if they have no consequence for reaching destinations. Notwithstanding all of the above, as the network size increases, flat routing usually requires much more memory and processing power than its hierarchical counterparts. More importantly, if not carefully designed, flat routing techniques may result in much more bandwidth consumption than hierarchical approaches.

As previously discussed, except for very specific applications, the state of the art on microelectronics allows inexpensive memory chips inside the network nodes. These chips provide sufficient memory space to handle even tens of thousands of nodes. Processing power is not so inexpensive, but efficient (incremental) algorithms still allow network with reasonable priced processors to handle the route computation algorithms when run over a large topology. Thus, the main challenge to network survivability as size increases is the excessive bandwidth consumption. So, it is not surprising that significant effort has been directed in reducing this bandwidth consumption.

The techniques for bandwidth consumption reduction for flat routing can be classified into: efficient flooding, limited generation, limited dissemination. These techniques can be used in isolation or in combination.

5.2.1 Efficient Flooding

Most proactive and reactive algorithms rely on flooding of control packets to a subset of nodes in the network. However, classical flooding is a very inefficient technique, resulting in each node receiving the same packet several times.

Efficient flooding techniques reduce the number of times a flooded message is retransmitted, and at a minimum, each intended recipient receives each flooded packet at least once. For example, a technique may consist of finding a tree in the topology such that the set of nodes in the tree covers (i.e. is neighbor of) all the nodes in the network. An effective flooding technique may then consist of propagating the message across all the nodes in the tree. Every node in the tree will have to transmit the message once.

Optimized Link State Routing (OLSR) [3], Topology Broadcast based on Reverse Path Flooding (TBRPF)[4], and Core Extraction Distributed Ad Hoc Routing (CEDAR)[37] are examples of protocols implementing efficient flooding algorithms.

Typically, the performance improvements obtained by using efficient flooding techniques increases with the average node degree of the network. Thus, these techniques are especially useful for networks with high density. However, as pointed out earlier in this chapter, high density scenarios are better handled by means of a topology (power) control algorithm which reduces the average node degree to an acceptable level. If topology control mechanisms are in place and the network is of the kind defined by assumptions a.1 through a.8, then the performance improvement obtained by efficient flooding will be a constant factor independent of the network size, and therefore this technique will not affect the asymptotic behavior of the protocol being run. Thus, for bounded node degree, effective flooding techniques – while helpful – do not solve the routing protocol scalability problem and can not be used in lieu of hierarchical routing.

5.2.2 Limited Generation

Limited generation techniques limit the amount of control information being generated.

For example, Global State Routing (GSR)[20], and Discretized Link State (DLS)[22] routing limit routing update generation to times which are multiples of a base period t_e. At such times, all the changes since the last update are collected and sent to all other nodes in the network. This technique is effective for high mobility.

Source-Tree Adaptive Routing (STAR)[21] limits the update generation by only triggering updates for link state changes that affect another node's best route selection. Most other limited generation techniques (e.g. the one used in OLSR[3]) reduce the amount of control information by operating on a network subgraph formed by all the nodes and a subset of the links in such a way that the resulting subgraph is connected. The level of performance improvement that can be obtained with these *partial-topology* techniques is not easy to analyze. However, it is expected to be above the one achieved by efficient flooding, but below the one obtained by limited dissemination techniques.

5.2.3 Limited Dissemination

In limited dissemination techniques, most routing information updates are not sent to the entire network but to a smaller subset. The subset may change over time.

For example, ZRP[17] and NSLS[22] protocols limit the event-driven link state update propagation to their k-neighbors only.

In Fisheye State Routing[12], a node divides the set of nodes into the in-scope and the out-of-scope subsets. A node then propagates information about nodes in its in-scope subset with a pre-configured frequency. Information about out-of-scope nodes is propagated with a smaller frequency. In other words, most of the messages propagating routing information have been stripped of information related to the out-of-scope nodes.

The family of Fuzzy Sighted Link State (FSLS) algorithms[22] limits the LSU generation to multiples of a base time t_e. When a LSU is sent it does not (in general) travel to the entire network. Instead, it traverses the number of hops specified in the LSU's packet Time To Live (TTL) field. The value of the TTL field will depend on the current time index. Given its potential for scalability, the family of FSLS algorithm will be described in detail in the next subsection.

Limited dissemination techniques, by reducing the depth of propagation of routing updates to a small fraction of the network, hold better promise for scalability improvement for networks with a large diameter, as is the case when the network size increases and the average node degree is kept bounded. The challenge here is to do so in a way that does not overly compromise route optimally.

One technique, namely HSLS[22], – a member of the FSLS family – produces a significant change in link state asymptotic properties, reducing its scalability factor w.r.t. network size from 2 to 1.5, rendering the algorithm indeed scalable w.r.t. network size. Thus, remarkably, HSLS presents *even better* scalability properties than hierarchical routing approaches.

5.2.4 The family of Fuzzy Sighted Link State (FSLS) algorithms

In the FSLS family of algorithms[22], 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. 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.

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 6 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 6). 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 6, a LSU with 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 7. 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. Of particular interest are Discretized Link State (DLS), Near Sighted Link State (NSLS), and Hazy Sighted Link State, discussed next.

DLS is obtained by setting $s_i = \infty$ for all i (see Figure 8 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

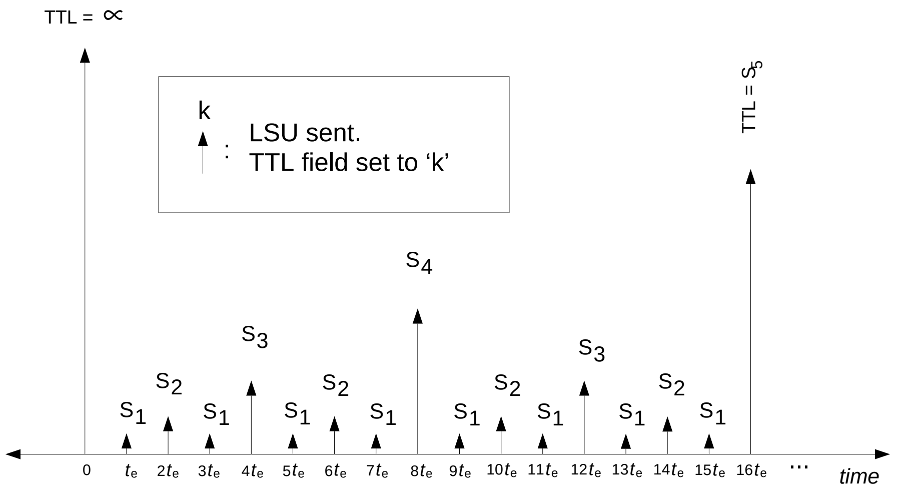
TTL = ∞
k
LSU sent.
TTL field set to 'k'
TTL = s_5
s_1 s_2 s_1 s_3 s_1 s_2 s_1 s_4 s_1 s_2 s_1 s_3 s_1 s_2 s_1
0 t_e $2t_e$ $3t_e$ $4t_e$ $5t_e$ $6t_e$ $7t_e$ $8t_e$ $9t_e$ $10t_e$ $11t_e$ $12t_e$ $13t_e$ $14t_e$ $15t_e$ $16t_e$ ...
time

Figure 6: Example of FSLS's LSU generation process

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.

NSLS is obtained by setting $s_i = k$ for $i < p$ and $s_p = \infty$ (for some p integer), as shown in Figure 8 (right). 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. NSLS has similarities with ZRP, DREAM, and FSR.

Suppose that initially, a node has knowledge 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

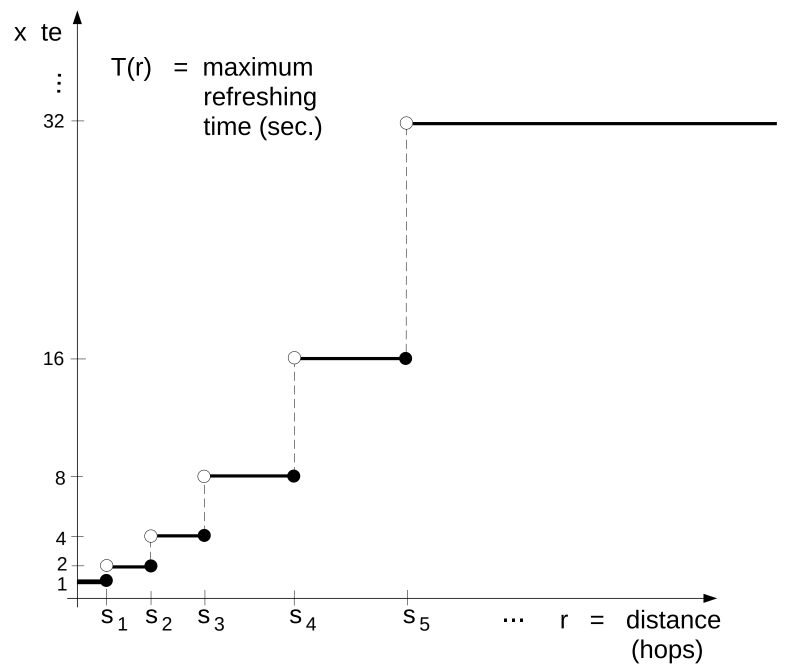

Figure 7: Maximum refresh time $T(r)$ as a function of distance from link event.

inaccurate, allows tracing the destination. NSLS has similarities with ZRP, DREAM, and FSR.

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 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. Based on the uniform traffic distribution (assumptions a.1 - a.8) among all the nodes in the network, the best values of $\{s_i\}$ were found (see [22]) to be equal to $\{s_i\} = \{2^i\}$. FSLS with $\{s_i = 2^i\}$ is called the Hazy Sighted Link State (HSLS) algorithm[22]. Figure 2 shows an example of HSLS's LSU generation process. HSLS induces an almost linear relationship between route information latency and the distance in hops. This in turn causes the uncertainty in the relative

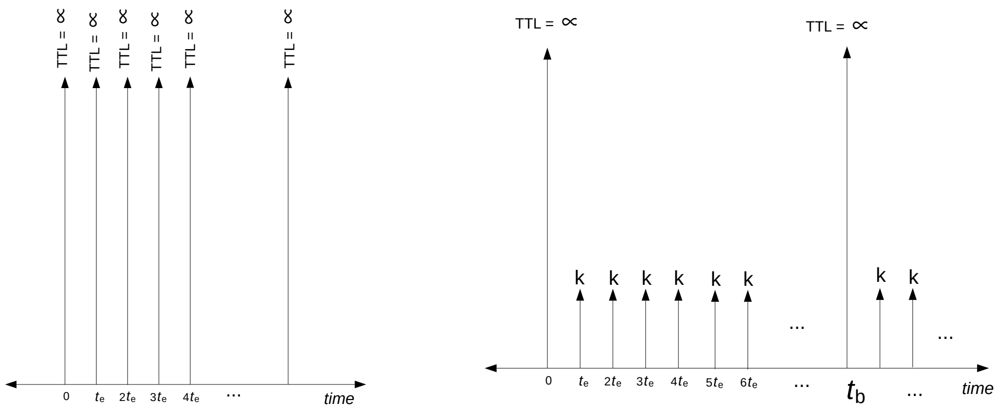

Figure 8: DLS's (left) and NSLS's (right) LSU generation process.

angular position of the distant node to be roughly constant independent of the distance. Since in hop-by-hop routing a node is only concerned with the next hop decision, and the probability of making a wrong decision depends mainly in the angular uncertainty, which was roughly constant independent of the distance, we end up with a probability of making a bad next hop decision to be also roughly constant independent of the distance. Out of all possible assignments of probability of error versus distance, it turns out that the best performance is obtained when all the values are balanced. That is, the probability of error is roughly constant independent of the distance. Thus, HSLS's dissemination results in a linear relationship between latency and distance represent the optimal balance between proactive and sub-optimal routing overhead. If the latency versus distance curve grows faster than linear, too many mistakes are made when forwarding packets to nodes far away. If the curve grows slower than linear, we make fewer mistakes when finding routes for nodes far away than when finding routes to nodes close by, but the proactive overhead increases a fair amount since global LSUs would be sent more frequently (to reduce the latency in routing information for nodes far away).

5.3 Comparing HierLS and HSLS

The theoretical results in Table 2 shows that both HierLS and HSLS present good scalability with respect to network size. This result may be explained by the fact that both protocols induce a multi-level information dissemination technique. HSLS outperforms HierLS since HSLS's routes' quality does not degrade with network size. HSLS's angular displacement uncertainty is mainly dependent on the nodes speed and the timer period t_e, which is optimally set based on the mobility and traffic rates (regardless of network size). HierLS's routes's quality suffer small degradation each time the number of hierarchical levels is increased. Moreover, HSLS is able to improve the quality of its routes as a response to an increase in traffic load. HierLS's route quality, on the other hand, is dependent on the number of hierarchical levels, which depend on the cluster size, a parameter that is independent of the traffic load, leaving HierLS powerless to react to an increase in traffic load. Thus, HSLS present better scalability properties than HierLS.

However, the constants involved in the asymptotic expression may be too large, preventing HSLS from outperforming HierLS under real life scenarios. Therefore, HierLS and HSLS were compared through simulation.

Table 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 chooses a random direction among 4 possible values, and moves in that direction at 28.8 mph. Upon reaching the area boundaries, a node bounces back. The radio link capacity was 1.676 Mbps. Simulations 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 the DAWN project [38] modification of the MMWN clustering protocol [15]. Following the taxonomy presented in this paper, this protocol can be classified as a m-level hierarchy[8] with a cluster radius greater than one. The node affiliation decisions were performed by the cluster leaders with the goal of balancing cluster sizes with a lower and upper bound on the cluster sizes of 9 and 35. The cluster leader selection criteria was to choose the node in the cluster with the largest number of (unassigned) k-hop neighbors. The virtual gateway method of topology abstraction was used.

The metric of interest is the throughput (i.e. fraction of packets successfully delivered). Table 3 shows the throughput obtained under two different MAC protocols: unreliable and reliable CSMA. For reliable CSMA, packets were retransmitted up to 10 times if a MAC-level ACK was not received in a reasonable time. We can see that in both cases HSLS outperforms

[8]Although m-level can be formed, since the network size was relatively small, only 2 levels were formed during the simulations.

Protocol	UNRELIABLE	RELIABLE
HSLS	0.2454	0.7991
HierLS-LM1	0.0668	0.3445

Table 3: Throughput of a 400-node network.

HierLS, although the relative difference is reduced under the reliable MAC case. This can be explained considering that the high rate of collisions experienced under unreliable CSMA favored shorter paths. For nodes close by, HSLS may provide almost optimal routes while HierLS routes may be far from optimal if the destination belongs to a neighboring cluster. Thus, we can see that an unreliable MAC biases performance towards HSLS. Another factor to take into account is the latency to detect link up/downs. Under HierLS this information is synchronized among all the nodes in the cluster and therefore some latency is enforced to avoid link flapping. In HSLS, on the other hand, each node may have its own view of the network, and as a consequence a node may be more aggressive in temporarily taking links down without informing other nodes. As a consequence, HSLS is more aggressive and reacts much faster to link degradation, using alternate paths if available.

The simulation results presented do not represent a comprehensive study of the relative performance of HierLS versus HSLS under all possible scenarios. They just present an example of a real-life situation to complement the theoretical analysis. The theoretical analysis focuses on asymptotically large networks, heavy traffic load, and saturation conditions where the remaining capacity determines the protocol performance. The simulation results, on the other hand, refer to medium size networks with moderate loads, where depending on the MAC employed, other factors such as the quality of the links that neighbor discovery declares up, the latency on detecting link failures, etc., may have more weight over the protocols' performance.

Thus, 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 requires a smaller topology table to be stored/processed. On the other hand, if implementation complexity is an issue, then HSLS should be preferred.

5.4 Discussion

Traditionally, as network size increases it was believed that the best alternative for routing scalability was the inclusion of hierarchical routing techniques. Several such techniques were designed, as for example the work done under DARPA's SURAN project (see [39, 40] for a survey).

Hierarchical routing solutions, however, quickly showed their drawbacks. For one, their implementations proved too complex, having to handle too many exception situations, especially in scenarios –as in the military – where nodes chosen for special functions (e.g. cluster leaders, location management servers, etc.) are susceptible to attack/destruction. In these scenarios, the routing protocol has to specify mechanisms for backup selection and activation. Another drawback is that the overhead induced for maintaining the hierarchy and for keeping up-to-date location management information reduces the bandwidth savings achieved due to reduction of link state information dissemination. These drawbacks have played a large part in the fact that in practice no multilevel hierarchical protocols has been implemented in real life networks. All current hierarchical routing implementations limit its number of hierarchical levels to 2, which in turn puts a limit to its scalability.

The difficulty in the implementation of hierarchical routing motivated the search of alternative, simpler techniques to improve routing protocol scalability with respect to network size, including but not limited to efficient flooding, limited generation, limited dissemination, and a combination thereof. This section presented a comparison of these new techniques against the classical hierarchical routing approach.

The theoretical analysis showed that there is no fundamental advantage provided by hierarchical routing over an efficient combination of these techniques, as for example, the HSLS algorithm. Indeed, HSLS scalability properties with respect to network size are not worse than that achieved by hierarchical routing. Furthermore, HSLS presented better scalability properties with respect to traffic rate.

The experimental study also pointed out that hierarchical routing implementations, while extremely more complex than HSLS's implementation, are not guaranteed to achieve better performance than HSLS. The relative performance of the protocols depends on other factors, such as the link layer latency on detecting link failures, or the MAC layer susceptibility to collisions between control and data packets.

Thus, we conclude that limited dissemination techniques are good candidates for achieving scalable routing protocols. Regarding which protocol should be preferred in a practical situation, we realize that this determination depends on several factors. We may say that when network size, mobility, and traffic increases; an efficient MAC is used; or implementation complexity is one of the main concerns; limited dissemination techniques as HSLS should tend to be preferred over hierarchical approaches.

But, in scenarios unfavorable to limited dissemination techniques such as HSLS, hierarchical approaches should tend to be preferred. Scenarios unfavorable to HSLS include scenarios where storage capacity at each node is limited, the topology is sparse, or there is a large amount of hostile misbehaving nodes. It was already noted that HSLS requires more storage space than hierarchical approaches. Sparse, tree-like topologies present a challenge to HSLS, since link status changes of links on shortest paths will have an effect on routing decisions taken by nodes several hopes away from the node experiencing the link status change. Roughly speaking, instead of 'locally repairing' the broken route, the network will have to back the old route up until reaching a node (maybe even the source) from where a new route segment to the destination node may be built. Since HSLS link state dissemination to nodes more than 2 hops away is not immediate but a latency is induced, this may result in temporary routing loops. The impact of these routing loops (other than rendering the destination unreachable) on the network performance depends on the loop detection/removal capabilities available on the network. Additionally, since in SLS a node receives 2 LSUs each time there is a link status change (one from each node at each extreme of the link) a node can validate routing information sent by misbehaving nodes. In HSLS – depending on the distance to the node experiencing a link status change – only one LSU may be received, making the problem of detecting misbehaving nodes more difficult.

Finally, the reader should keep in mind that hierarchical routing's relative performance (against limited dissemination techniques) may increase in scenarios different to the homogeneous network considered in this chapter (defined by assumptions a.1-a.8). For example if the network is formed by some low power terrestrial nodes and some high power/aerial nodes with much better coverage. Or if the network is formed by nodes whose movements are not uncorrelated but follow well defined group patterns. In these cases, a desirable property of the hierarchical routing technique would be to be able to extract the underlying network structure and mimic it in its cluster formation process. If successful, the clustering mechanism will significantly reduce the bandwidth consumed by the location management procedure, resulting in improved scalability with respect to the results shown in Table 2, where assumptions a.1 through a.8 were valid. These scenarios may provide hierarchical routing approaches an edge above limited dissemination flat techniques that do not try to exploit the underlying network structure.

6 Conclusions and Future Research Directions

This chapter addressed the issue of the scalability of routing protocol for bandwidth-constrained ad hoc networks from a fundamental viewpoint. It presented concepts, metrics, and methodologies for the study of routing protocols. Analytical results for the scalability of a representative set of routing protocols were discussed, providing a deeper understanding of the characteristics and tradeoffs associated with various classes of routing protocols for mobile networks. This treatment of the subject is not all-inclusive. Several (valid) assumptions about the networking scenario were adopted in order to achieve closed form expressions. We hope, however, to have succeeded in providing the reader with the necessary tools for performing his/her own analysis and performance assessment under the particular networking scenario he/she is interested in.

In particular, as a consequence of the fundamental analysis two common misconceptions were exposed:

- **Misconception 1:** As traffic load increases, the bandwidth allocated to routing information dissemination should decrease.
- **Misconception 2:** As network size increases the best option is to engage a hierarchical routing algorithm.

The analysis also pointed out the best performing approaches in the context of scalability with respect to network size: limited dissemination flat routing, and m-level hierarchical routing. Thus, a more in depth analysis of these 'winner' approaches were presented.

The treatment of hierarchical routing approaches showed that they are not only extremely complex to implement but they are also hard to analyze, to the point of not being clear if the performance improvement to be achieved with a particular hierarchical routing approach would justify the implementation headache. This disappointment with hierarchical routing complexity has motivated a surge of interest in the study of scalable non-hierarchical protocols.

We presented the main techniques to improve scalability for flat routing. We compare the (probably) best of this techniques against an average hierarchical routing technique and the result was that the flat routing scheme, while much easier to implement, outperforms the hierarchical approach under the high stress (asymptotic) regime and also under the moderate stress scenario.

In conclusion, it seems that imposing an arbitrary hierarchy in homogeneous ad hoc networks provides no scalability advantage (over flat-routing scalability-improving techniques). It seems that hierarchical routing would justify its high implementation complexity only if the hierarchy built was a response/reflection of an underlying hierarchy/structure in the network.

Future research should extend the results shown in Section 4 for scenarios different to the ones defined by assumptions a.1-a.8. Of particular interest are the group mobility scenarios, since it appears that they are likely to be present due to patterns on human motion following streets, highways, etc., and the task requirements of automated systems (robots, etc.). For these scenarios the theory can be easily extended, and should be used to help in the design of structure-learning gain functions for cluster formation, like the one developed in SOAP[24].

This chapter has addressed the scalability challenge from a bandwidth point of view. As ad hoc networks used become widespread, different applications will need to be supported. A particular challenge is posed by QoS demanding applications, where the question is not to get the best route to a destination but whether a particular QoS constraint can be satisfied (Call Admission Control) by the network and how. Call Admission Control (CAC) usually requires more information than say, minimum hop routing. Moreover, the impact of routing information latency or imprecision into system performance is not easy to evaluate. Defining a metric that captures the effect of routing protocols (control overhead, route information latency, etc.) in QoS related performance (as *Total Overhead* does for bandwidth related performance) is not an easy task. However, this task is paramount for the proper design of routing protocols enabling large ad hoc network running application with demanding QoS constraints such as voice and videoconferencing. Support of such applications may well be the rite of passage required for ad hoc networking technology in order to reach the mass market, and as such it may define the future of this technology.

References

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

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

[3] P. Jacquet, P. Muhlethaler, and A. Quayyum, “Optimized Link State Routing Protocol”, draft-ietf-manet-olsr-05.txt, Internet Draft, IETF MANET Working Group, Nov. 2000. Work in Progress.

[4] B. Bellur, R. Ogier, “A Reliable, Efficient Topology Broadcast Algorithm for Dynamic Networks,” *Proc. IEEE INFOCOM*, 1999.

[5] C. Perkins. “Ad-Hoc On-Demand Distance Vector Routing”. MILCOM’97 panel on Ad-Hoc Networks, Monterey, CA, November 3, 1997.

[6] C. Perkins and E. M. Roger, “Ad-Hoc On-Demand Distance Vector Routing ,” *Proceedings of IEEE WMCSA’99*, New Orleans, LA, Feb. 1999, pp. 90-100.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] C. Santivanez, A. B. McDonald, I. Stavrakakis, S. Ramanathan. “On the Scalability of Ad Hoc Routing Protocols,” in *Proceedings of IEEE Infocom’2002*, New York, USA, June 2002.

[12] 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.

[13] D. J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed ALgorithm," *IEEE Transactions on Communications*, 1981, 29(11): 1694-1701.

[14] C. -C. Chiang and M. Gerla, " Routing and Multicast in Multihop, Mobile Wireless Networks," *Proceeding of the IEEE UCUPC'97*, San Diego, CA, Oct. 1997.

[15] 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.

[16] E. M. Belding-Royer, "Hierarchical Routing in Ad Hoc Mobile Networks," *Wireless Communications & Mobile Computing*, No. 5, Vol. 2, August 2002.

[17] Z. Haas, " A New Routing Protocol for the Reconfigurable Wireless Networks," *Proceedings of ICUPC'97,* San Diego, CA, Oct. 12, 1997.

[18] Z. Haas and M. Pearlman, "The performance of query control schemes for the zone routing protocol," in *ACM SIGCOMM*, 1998.

[19] 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.

[20] T. Chen and M. Gerla, " Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks ," *Proceedings of IEEE ICC '98*, 1998.

[21] 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.

[22] C. Santivanez, S. Ramanathan, and I. Stavrakakis,"Making Link State Routing Scale for Ad Hoc Networks", In *Proceedings of MobiHOC'2001*, Long Beach, CA, Oct. 2001.

[23] E. W. Dijkstra, “A Note on Two Problems in Connection with Graphs,” *Numerische Math.*, 1:269-271, 1959.

[24] C. Santiváñez, “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,” *Doctoral thesis*, Electrical and Computing Engineering Department, Northeastern University, Boston, MA, November 2001.

[25] 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.

[26] X. Hong, K. Xu, and M. Gerla, “Scalable Routing Protocols for Mobile Ad Hoc Networks,” *IEEE Network Magazine*, Special Issue on Scalabili in Communication Networks, No. 4, Vol. 16, July/August 2002, pp. 11-21.

[27] P. Gupta and P.R. Kumar. “The Capacity of Wireless Networks”, *IEEE Transaction on Information Theory*, 46 (2):388-404, March 2000.

[28] M. Grossglauser and D. Tse. “Mobility Increases the Capacity of Adhoc Wireless Networks”, in *Proceedings of IEEE Infocom'2001*, Anchorage, Alaska, April 2001.

[29] S. Ramanathan and R. Hain, “Topology Control of Multihop Radio Networks using Transmit Power Adjustment,” in *Proceedings of IEEE Infocom'2000*, Tel Aviv, Israel, 2000

[30] M. Steenstrup, “Cluster-Based Networks,” In *Ad Hoc Networking*, C. E. Perkins, Ed., Chapter 4, Addison-Wesley, 2001, pp. 75-138.

[31] 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.

[32] J. Zavgren, “NTDR Mobility Management Protocols and Procedures,” In *Proceedings of IEEE MILCOM'97*, Nov. 1997.

[33] 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.

[34] 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.

[35] L. Kleinrock and F. Kamoun, “Hierarchical Routing for Large Networks,” *Computer Networks,* No. 1, January 1977, pp. 155-174.

[36] P. F. Tsuchiya, “ Landmark Routing : Architecture, Algorithms, and Issues,” *Technical Report MTR-87W00174*, Cambridge, MA: MITRE Corporation, September 1987.

[37] R. Sivakumar, P. Sinha, and V. Bharghavan, “ CEDAR: a Core-Extraction Distributed Ad hoc Routing algorithm,” *Proceedings of INFOCOM’99*, New York, 1999.

[38] http://www.ir.bbn.com/projects/dawn/dawn-index.html

[39] G. Lauer, “ Packet Radio Routing,” In *Routing in Communications networks*, edited by Martha E. Steenstrup, Chapter 11, pages 351-396. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[40] G. Lauer, “ Hierarchical Routing Design for SURAN,” *Proceedings of ICC’86*, 1986, pages 93-101.