Making Link-State Routing Scale for Ad Hoc Networks

César A. Santiváñez
BBN Technologies
10 Moulton St.
Cambridge, MA 02138, USA
csantiva@bbn.com

Ram Ramanathan
BBN Technologies
10 Moulton St.
Cambridge, MA 02138, USA
ramanath@bbn.com

Ioannis Stavrakakis
Department of Informatics
University of Athens
Athens, Greece
istavrak@di.uoa.gr

ABSTRACT

In this paper, we introduce a class of approaches that attempt to scale link-state routing by limiting the scope of link state update dissemination in space and over time. We present the first fundamental analysis of this generic class, which we call "Fuzzy Sighted Link State routing". Using a novel perspective on the "overhead" of a protocol that includes not only the overhead due to control messages but also due to route sub-optimality, we formulate an analytical model whose solution automatically leads to the best algorithm in this class. This algorithm is shown to have nearly the best possible asymptotic overhead for any routing algorithm – proactive or reactive. Simulation results are presented that compare the performance of several algorithms in this class.

1. INTRODUCTION

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

Given its advantages, a sufficiently scalable version of link-state routing would be invaluable for ad hoc networks. Not surprisingly therefore, there are a number of approaches in the literature with this goal. These approaches may be classified into *efficient dissemination* approaches and *limited dissemination* approaches. Both attempt to reduce the routing update overhead, but do so in different ways. In efficient dissemination, updates are sent throughout the network, but more efficiently compared to traditional flooding. Examples include TBRPF[2], OLSR [3], STAR [4], etc. In contrast, limited dissemination consists of restricting the scope of routing updates in space and time. Examples include hiearchical link state [5], FSR and GSR (see [6]), etc.

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

Limited dissemination techniques incur a cost in terms of sub-optimal routing that needs to be considered in formulating our problem and conducting the analysis. Indeed, this is a case with many other routing protocols as well, including DSR[8], AODV [9], etc. Traditionally, the cost of sub-optimal routing has been ignored, and only the cost of control message overhead been considered. We propose a new definition of "overhead" that includes not only the control message overhead but also the cost of sub-optimal routing. Such a definition facilitates fair comparison of protocols not only within the fuzzy-sighted family, but also amongst previously published protocols.

Our contributions include the following. We open a new design space for link state routing protocol by presenting a family of (potentially scalable) algorithms that are neither global nor local, but where each node may have a different view of the network. We introduce a new definition of overhead that allows for comparison among different protocols. We present an analytical model that facilitates the study of a large class of routing protocols.

In particular, a unique feature of our work is that the resulting algorithm is *synthesized* automatically from the analysis, rather than being followed by the analysis, which is normally the case. Morever, it is performance-driven, focusing on average system performance instead of focusing on handling exceptional (rare) cases [1], or achieving qualitative characteristics (loop freedom, database consistency, etc.) whose impact on the overall system performance is not clear.

The remainder of this chapter is organized as follows: Section 2 presents some related work. Section 3 presents a discussion on scalability that leads to the definition of the *total overhead* and to focus on limited dissemination link state approaches. Section 4 introduces the family of Fuzzy Sighted Link State (FSLS) algorithms, that are intended to reduce (limit) the routing information overhead at the expense of

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

some route sub-optimality. Section 5 presents an analytical model that determines the best algorithm in the family of FSLS algorithms, namely the Hazy Sighted Link State (HSLS) algorithm. Section 6 complements the analysis with simulation results. Finally, section 7 presents some conclusions.

2. RELATED WORK

There has been a vast amount of research on routing algorithms for ad hoc networks. Most routing algorithms can be classified as being proactive or reactive.

Proactive protocols attempt to continously maintain up-to-date routing information for each node in the network. Standard Link State (SLS) and Standard Distance Vector (SDV) (see [1]), TBRPF [2], OLSR [3], and STAR [4] are examples of proactive approaches.

One way of scaling proactive approaches is using hierarchical techniques. Hierarchical routing algorithms based on link-state have been developed and implemented as part of the DARPA Survivable Adaptive Networks (SURAN) program [7], and more recently as part of DARPA Global Mobile Information Systems (GloMo) program (see for example [5]). Hierarchical techniques, however, may be too costly or complicated to maintain, especially under high mobility.

Reactive protocols build the routing information "on demand", that is, only when there is a packet that needs to be routed. DSR[8], AODV [9], and DREAM [10] are examples of reactive protocols. Most of these protocols have been studied through simulations on relatively small (less than 100 nodes) networks. It is not clear that they will scale to larger sizes.

There are also some hybrid protocols that attempt to combine reactive and proactive features, as for example the Zone Routing Protocol (ZRP) [11]. ZRP attempts to balance the proactive and reactive overheads induced on the network by adaptively changing the size of a node 'zone'.

Scalability and other performance aspects of ad hoc routing have been studied predominantly via simulations. The lack of much needed theoretical analysis in this area is due, we believe, in part to the lack of a common platform to base theoretical comparisons on, and in part due to the abstruse nature of the problem. Despite limited prior related theoretical work, there have been notable exceptions. In [12] analytical and simulation results are integrated in a study that provides valuable insight into comparative protocol performance. However, it fails to deliver a final analytical result, deferring instead to simulation.

Our work is unique in several ways. First, our analysis considers all the different sources of overhead in a unified framework. Second, we relax the usual requirement on proactive approaches that all the nodes must have a consistent view of the network. Third, our results are derived from a mobility-based probabilistic analytical model instead of being derived from simulations, and therefore they have a broader applicability. Finally, this paper and [16] are the only attempts (to the authors knowledge) to theoretically understand the limits on scalability for large mobile ad hoc networks.

3. NEW PERSPECTIVE ON SCALABILITY

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

While it is true that the control overhead significantly affect 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 sub-optimal 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 sub-optimal routes become a fact of life in any scalable routing protocol. Sub-optimal 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 sub-optimal routes by limiting the scope of topology information dissemination (e.g. hierarchical routing [5]) and/or limiting the time between successive topology information updates dissemination so that topology updates are no longer instantaneously event-driven (e.g GSR [6]).

Thus, it is necessary to revise the concept of *overhead* so that it includes the effect of sub-optimal routes in capacity limited systems, that is, *sub-optimal 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 comparable to the other types of overhead. Approaches that attempt to minimize only the control overhead may lead to the (potentially erroneous) conclusion that they are "scalable" by inducing a fixed amount of the aforementioned overhead, while in practice the resulting performance be seriously degraded as the extra bandwidth overhead induced by sub-optimal routes increases with the network size. Thus, a more effective definition of the overhead – which will be considered in the remainder of this work – is introduced in the next subsection.

3.1 Total Overhead

Definition : *The* total overhead *is defined as the total amount of bandwidth used in excess of the minimum amount of bandwidth required to forward packets over the shortest distance (in number of hops) by assuming that the nodes had instantaneous full-topology information.*

The different sources of overhead that contribute to the *total overhead* may be grouped and expressed in terms of *reactive*, *proactive*, and *sub-optimal routing overheads.*

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

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

The *sub-optimal 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 sub-optimal routing overhead.

The *total overhead* provides an unbiased metric for performance comparison that reflects bandwidth consumption. Despite increasing efficiency at the physical and MAC-layers, bandwidth is likely to remain a limiting factor in terms of scalability, which is a crucial element for successful implementation and deployment of ad hoc networks. The authors recognize that *total overhead* may not fully characterize all the performance aspects relevant to specific applications. However, it can be used without loss of generality as it is proportional to factors including energy consumption, memory and processing requirements, and, furthermore, delay constraints have been shown to be expressed in terms of an equivalent bandwidth [13].

3.2 Achievable regions and operating points

The three different overhead sources mentioned above are locked in a 3-way trade-off since, in an already efficient algorithm, the reduction of one of them will most likely cause the increase of one of the others. For example, reducing the 'zone' size on ZRP will reduce ZRP's proactive overhead, but will increase the overhead incurred when 'bordercasting' new route request, thus increasing ZRP's reactive overhead. The above observation leads as to the definition of the *achievable region* of overhead as the three dimensional region formed by all the values of proactive, reactive, and sub-optimal routing overheads that can be achieved (induced) by any protocol under the same scenario (traffic, mobility, etc.). Figure 1 shows a typical 2-dimensional transformation of this 'achievable region' where two sources of overhead (reactive and sub-optimal 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 sub-optimal routing overheads.

It can be seen that the achievable region is convex [2], lower-bounded by the curve of overhead points achieved by the 'efficient' (i.e. minimizing some source of overhead given a condition imposed on the others) protocols.

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

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

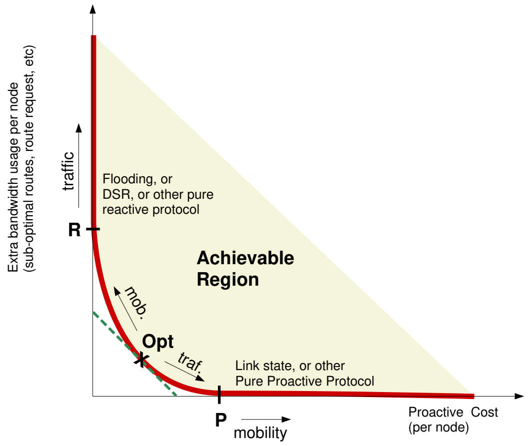

Figure 1: Overhead's achievable region.

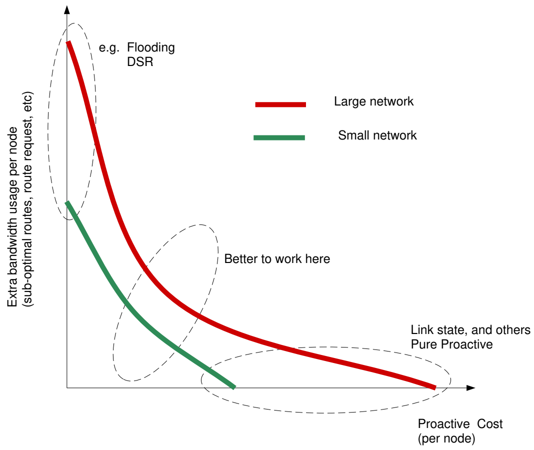

Figure 2: Change in achievable region due to size.

Different scenarios result in different slopes of the boundary of the achievable region and consequently different points Opt. For example, if the traffic increases or diversifies R moves upward and, if mobility is low P moves to the left and may cause Opt to coincide with the point P (pure proactive protocol with optimal routes). The reverse is also true as the mobility rate increases and the traffic diversity/intensity decreases. Figure 2 shows how the boundary of the achievable region is (re)shaped as the network size increases. The lower curve corresponds to the boundary region when the network size is small. The effect of increasing the network size is to 'pull' the boundary region up. However, the region displacement is not uniform as will be discussed next.

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

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

In the other hand, protocols inducing 'intermediate points', such as Hierchical link state (HierLS) and ZRP, may increase more slowly with respect to N. In [16] it is shown that under the same set of assumptions as this paper (Section 5.1) HierLS and ZRP growth with respect to N was roughly $N^{1.5}$ and $N^{1.66}$, respectively.

Summarizing, it can be seen that points P and R increase proportionally to $\Theta(N^2)$ whereas an 'intermediate' point as HierLS increases almost as $\Theta(N^{1.5})$. [3] Referring again to Figure 2, it is easy to see that the extreme points are stretched "faster" than the intermediate points. Thus, *as size increases, the best operating point is far from the extreme points P and R but in the region where the proactive, reactive, and sub-optimal routing overheads are balanced.*

Further research should be focused on protocols such as the Zone Routing Protocol (ZRP) [11] HierLS variants (e.g. [5] and [15]), and other protocols that operate in this (intermediate) region, where sub-optimal routes are present.

4. FUZZY SIGHTED LINK STATE (FSLS) ALGORITHMS

It was previously pointed out that a pure proactive protocol such as SLS may not scale well with size since the overhead it induces increases as rapidly as N^2). However, a reduction of the proactive overhead may be achieved both in space (by limiting which nodes the link state update is transmitted to) and time (by limiting the time between successive link status information dissemination). Such a reduction on proactive overhead will induce an increase in sub-optimal routing overhead, and therefore a careful balance is necessary. This observation has motivated the study of the family of Fuzzy Sighted Link State (FSLS) protocols introduced below, where the frequency of link state updates (LSUs) propagated to distant nodes is reduced based on the observation that in hop-by-hop routing, changes experienced by nodes far away tend to have little impact in a node '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 Status 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.[4]

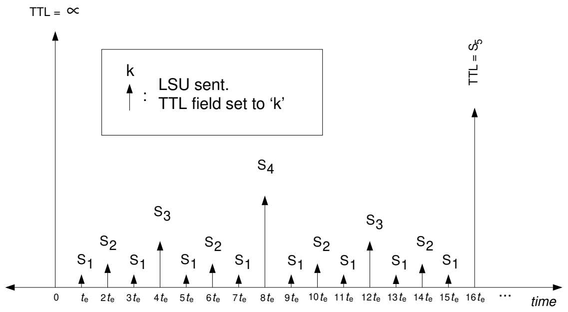

Figure 3: Example of FSLS's LSU generation process

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, broadcast LSUs are going to be transmitted with high probability.

Figure 3 shows an example of FSLS's LSU generation process when mobility is high and consequently LSUs are always generated every t_e seconds. Note that the sequence $s_1, s_2, \ldots$ is non-decreasing. For example consider what happens at time $4t_e$ (see figure 3). 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 (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 a odd number) the link state change activity during the past $4t_e$ seconds needs to be checked and if there is any a LSU with TTL set to s_3 will be sent. Thus, in the highly mobile scenario assumed on figure 3, 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 $T(r)$ versus distance (r) is as shown in Figure 4. The function $T(r)$ will determine the latency in the link state information, and therefore will determine the performance of the network under a FSLS algorithm.

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

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

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

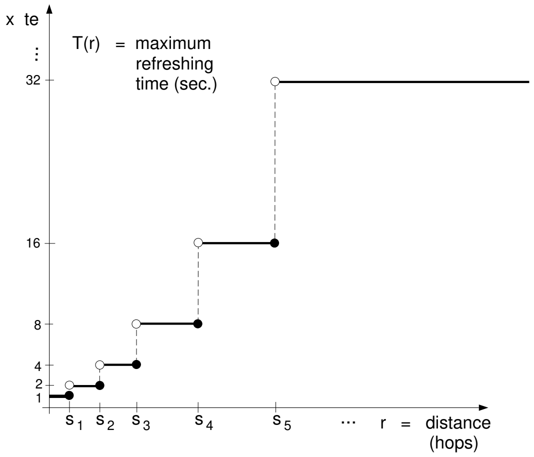

Figure 4: Maximum refresh time $T(r)$ as a function of distance from link event.

DLS is obtained by setting $s_i = \infty$ for all i. DLS is similar to the Standard Link State (SLS) algorithm and differs only in that under DLS a LSU is not sent immediately after a link status change is detected by only when the current t_e interval is completed, to collect several link status changes in one LSU. DLS is a modification of SLS that attempts to scale better with respect to mobility. Under high mobility, DLS presents some similarities with Global State Routing (GSR)[6], another protocol that attempts to scale with mobility. In GSR, a node exchanges its version of the network topology table with its one-hop neighbors each t_{flood} seconds. This way, GSR limits the frequency of link state updates to be no greater that $\frac{1}{t_{flood}}$.

In highly mobile scenarios (where LSUs are sent every t_e seconds) DLS induces the same proactive overhead (in bits) as Global State Routing (GSR) (setting $t_e = t_{flood}$), since they both require control packets transmission of the equivalent of N times the average topology table size (in bits) each t_e (t_{flood}) seconds (N is the network size). However, DLS latency on the transmission of LSUs to nodes far away is fixed, i.e. $T(r) = t_e$, while GSR's increases linearly with distance, i.e. $T(r) = t_e * r$ (since a link status update will have to wait at most t_e – and in average $\frac{t_e}{2}$ – seconds before it is propagated one more hop away from the node experiencing the link change). Thus, DLS is expected to outperform GSR, especially for large networks [5].

Another member of the FSLS family is NSLS, obtained by setting $s_i = k$ for $i \leq p$ and $s_p = \infty$ (for some p integer). [6] In NSLS, a node receives information about changes in link status from nodes that are less than 'k' hops away (i.e. inside its sigth area), but it is not refreshed with new link state updates from nodes out-of-sigth. Suppose that initially, a node has knowlegde 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 on 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 with the destination and going backwards until getting to the source; without removing old entries that although inaccurate, allows tracing the destination.

NSLS has similarities with the Zone Routing Protocol (ZRP) [11]. ZRP is a hybrid approach, combining a proactive and a reactive part. ZRP tries to minimize the sum of the proactive and reactive overhead. In ZRP, a node propagates event-driven (Link State) updates to its k-hops neighbors (nodes at a distance, in hops, of k or less). Thus, each node has full knowledge of its k-hop neighborhood and may forward packets to any node on it. When a node needs to forward a packet outside its k-hop neighborhood, it sends a route request message (reactive part) to a subset of nodes (namely, 'border nodes'). The 'border' nodes have enough information about their k-hops neighborhoods as to decide whether to reply to the route request or to forward it to its own set of 'border' nodes. NSLS is similar to the proactive part of ZRP [11] without the reactive route search.

Also, there are similarities between NSLS and the Distance Routing Effect Algorithm for Mobility(DREAM) [10], with the difference that NSLS limits the LSU propagation based on the number of hops traversed, meanwhile DREAM limits the position update message's propagation based on the geographical distance to the source.

There are aslo similarities between NSLS and Fisheye State Routing (FSR) [7] [6]. FSR uses the same topology dissemination mechanism as GSR, but it does not transmit the whole topology information each t_{flood} seconds. Instead, only a short version including only the closest ('in scope') nodes entries is transmitted. A second, larger timer (t_{large}) is used to exchange information about out-of-scope nodes (the rest of the network). Setting $t_e = t_{flood}$ and $t_b = t_{large}$, and k such that all the nodes in-scope are k or less hops away, NSLS induces the same control overhead as FSR; however, the latency in updating link state information – as reflected in the function $T(r)$ – is greater in FSR than in NSLS. In NSLS, $T(r) = t_e$ for $r \leq k$, and $T(r) = t_b$ for $r > k$. In the other hand, in FSR, a LSU have to wait at most t_e seconds (in average $\frac{t_e}{2}$) to be propagated one more hop away from the node experiencing the link event while it is in scope ($r \leq k$), and wait t_b seconds when it is 'out-of-scope' (i.e. $r > k$). Thus, for FSR $T(r) = t_e * r$ for $r \leq k$, and $T(r) = k * t_e + (r - k) * t_b$, which is significantly larger than the values for NSLS.

Finally, the family of Fuzzy Sighted Link State algorithms is based on the observation that nodes that are far away do

[5]GSR groups several LSUs in one packet. Thus, even if the same number of bits of overhead are transmitted, GSR transmits a smaller number of packets. In some scenarios, for example under a Request To Send (RTS)/Clear To Send (CTS)-based MAC with long channel acquisition and turn around times, the number of packets transmitted has a greater impact on the network capacity than the number of bits transmitted. In addition, GSR recovers faster than DLS from network partitions, especially under low mobility.

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

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

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. In the next sections, a uniform traffic distribution among all the nodes in the network is assumed and, as a consequence, the best values of $\{s_i\}$ were found to be equal to $\{s_i\} = \{2^i\}$, defining the Hazy Sighted Link State (HSLS) algorithm.

5. HSLS, THE OPTIMAL FSLS APPROACH

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

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

5.1 Network model

Let N be the number of nodes in the network, d be the average in-degree, L be the average path length over all source destination pairs, λ_{lc} be the expected number of link status changes that a node detects per second, λ_t be the average traffic rate that a node generates in a second (in bps). The following assumptions, motivated by geographical reasoning, define the kind of scenarios targetted on this work:

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

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

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

a.4 The maximum and average paths (in hops) among nodes in a connected subset of n nodes both increase as $\Theta(\sqrt{n})$. In particular, the maximum path across the whole network and the average path 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 the traffic from one node to a particular 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 $f_{t/t_0}(x, y, x_{t_0}, y_{t_0}) = \frac{1}{(t-t_0)^2} f_{1/0}(\frac{x-x_{t_0}}{t-t_0}, \frac{y-y_{t_0}}{t-t_0})$.

Assumption a.1 follows since imposing a fixed degree in a network is desirable and achievable. It is desirable, because allowing the density to increase without bound jeopardizes the achievable network throughput. It is achievable, because there are effective power control mechanisms available [14]. In general, a topology control algorithm should attempt to make the density as small as possible without compromising (bi)connectivity.

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

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

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

The traffic assumption is crucial to the analysis as it largely determines the effect of sub-optimal routing on performance. For example, if traffic is limited to the locality of the source then hierarchical routing [5] and ZRP [11] will benefit. On the other hand, having a small set of destinations will favor algorithms such as DSR [8]. Uniform traffic tends to favor proactive approaches as link state. In general, the effects of relatively equally distributed traffic tends to pose the most demanding requirements on a routing protocol. For this reason the analysis focuses on this case. Hence, assumption a.6 it is not considered an unfair bias towards link state approaches. A protocol that is scalable (with respect to traffic) under assumption a.6, will also be scalable under any other traffic pattern. On the other hand, a protocol that is scalable, under a localized traffic scenario, may fail when applied to a uniform traffic scenario.

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

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

5.2 Approximate expression for the total overhead

The following expression for the total overhead induced by a tagged node S runnning a generic FSLS algorithm under high mobility has been derived in [17]:[8]

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

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

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

5.3 Minimizing Total Overhead : The Hazy Sighted Link State (HSLS) algorithm.

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

5.3.1 A relaxed solution: lower bound

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

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

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

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

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

$$s_i = \max\{1,\ \min\{R,\ K*2^{i-1}\}\} \quad (3)$$

[8]The derivation has been removed out of this paper due to space constraints. The reader is referred to [17] for the details. [17] is available on-line and upon request.

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

$$E_1 = K^2 E_2 \quad (4)$$

$$E_1 = \sum_{i=1}^{n-1} \frac{s_i^2}{2^i} + \frac{R^2}{2^{n-1}} \quad (5)$$

$$E_2 = 2^{n-1} ln(R) - \sum_{i=1}^{n-1} 2^{i-1} ln(s_i) \quad (6)$$

Note that equations 3, 4, 5, and 6 define a system of equations that can be solved numerically as long as the values of n and R are known. Finally, by using the relationship between t_e and K (equation 2) in the optimal overhead expression the following is obtained:

$$\begin{aligned} S_{total} &= 2\, S_{proactive} \\ &= \sqrt{\frac{\lambda_t \alpha\beta\gamma\sigma L \mathcal{M} R^2 c\ size_{LSU}}{N}} \frac{E_1}{K} \quad (7) \end{aligned}$$

The above set of equations (from 3 to 6) is solved numerically for $R = 2, 3, \ldots, 500$ and for increasing values of n up to the point where incrementing n does not reduce the total overhead. [9] Thus, for each R, the best ratio $\frac{E_1}{K}$ obtained is recorded. This value will be all that is needed to compare the lower bound on total overhead derived here and the actual value achieved by the integer (feasible) solution presented in the next subsection (HSLS).

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

5.3.2 HSLS : An integer (feasible) solution

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

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

$$\begin{aligned} S'_{total} &= \min_{t_e} \{ \frac{c\ size_{LSU}}{t_e} E'_1 + \frac{\lambda_t}{N} \frac{\alpha\beta\gamma\sigma L}{4} \mathcal{M} R^2 E'_2 t_e \} \\ &= \sqrt{\frac{\lambda_t \alpha\beta\gamma\sigma L \mathcal{M} R^2 c\ size_{LSU}}{N}} \sqrt{E'_1 E'_2} \quad (8) \end{aligned}$$

where the prime suffix indicates a quantity associated with the integer (feasible) solution $s_i = 2^i$. E'_1, and E'_2 are computed according equations 5 and 6 respectively, but with the values of $s_i = 2^i$. Thus, these quantities become :

$$E'_1 = 2^n - 2 + \frac{R^2}{2^{n-1}} \quad (9)$$

$$E'_2 = (2^{n-1} - 1) ln(2) + 2^{n-1} ln(\frac{R}{2^{n-1}}) \quad (10)$$

and the value of t_e that achieves this minimum is :

$$t_e^{min} = \sqrt{\frac{4c\ size_{LSU} N}{\lambda_t \alpha\beta\gamma\sigma L \mathcal{M} R^2} \frac{E'_1}{E'_2}} \quad (11)$$

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

$$\delta = \frac{S_{total}^{integer} - S_{total}^{relaxed}}{S_{total}^{relaxed}} = \frac{\sqrt{E'_1 E'_2} - \sqrt{E_1 E_2}}{\sqrt{E_1 E_2}}$$

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

5.3.3 HSLS algorithm description and non-central nodes discussion

In the previous subsections, it has been determined that choosing $s_i = 2^i$ will probably minimize the total overhead induced by a node into the network. This assignment ($s_i = 2^i$) is referred to as the Hazy Sighted Link State (HSLS) algorithm. HSLS's generation process can be obtained by replacing $s_1, s_2, s_3, s_4, \ldots$ by $2, 4, 8, 16, \ldots$ respectively in Figure 3. HSLS's maximum 'refresh' time function is shown in Figure 5 It can be noted that there is an almost linear relationship between $T(r)$ and r. This linear relationship is responsible for HSLS's probable optimality for the central node studied in the previous subsection. This relationship reflects the fact that when forwarding packets to nodes far away, it is the angular displacement what really matters.

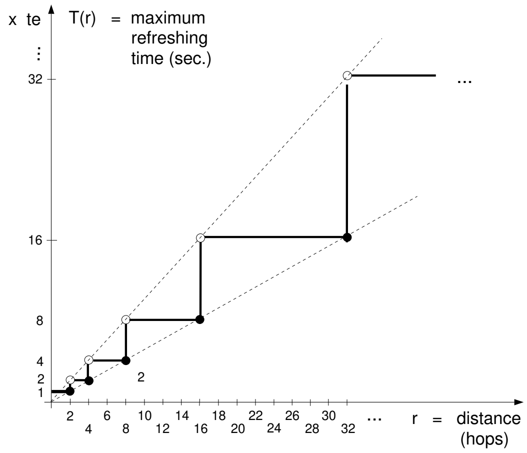

Figure 5: HSLS's maximum refresh time as a function of distance from link event.

Thus, HSLS successfully balances refresh periods and distances, so that the probability of making a suboptimal (bad) next hop decision is roughly the same for every destination

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

independently of the distance [10]. This balance is natural (avoiding 'hard' boundaries as in NSLS where a value has to be provided for k, the 'sight' area), and is typical when solving real life problems. It is the linear relationship between $T(r)$ and r what makes HSLS the winner algorithm regarding the centrally located node analyzed in the previous subsections. This property is kept when dealing with non-central nodes, so HSLS is expected to also be the winner FSLS algorithm when applied to a particular non-central node, and when considering the aggregation of all the nodes in the network. Then, the HSLS algorithm pseudo-code is provided in Figure 6. Note that the pseudo-code is slightly more complex than our discussion. It is because our discussion has focused on highly mobile scenarios. HSLS, however, adapts to slow varying scenarios, behaving like SLS when the rate of topological change is small (SLS mode in Figure 6). Also, the previous analysis – based on geographical reasoning – fails to capture the dynamics inside the 'topology region', that is, small scales. For practical implementations it was found through simulations that LSUs with small TTL do have a great impact in the algorithm performance. Level 1 LSUs do not induce much proactive overhead (just $\Theta(N)$) but they help to reduce loops and time to reaction to failures. So, every HSLS implementation should include them. [11] This does not contradicts the theoretical analysis, that did not care about them. The reader interested in a more detailed description of the HSLS protocol is referred to [17].

5.4 HSLS dependence on size, mobility and traffic

Equations 9 and 10 can be rewritten in function of a factor $f = \frac{R}{2^{n-1}} \in <1, 2]$ as:

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

And applying the above expressions on equation 8 (after simplification due to the fact that $cR^2 = N$ and $\sigma \approx \frac{1}{\alpha^2}$) the following expression is obtained:

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

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

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

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

The above expression shows that HSLS present excellent scalability properties, since it not only scales as well (or better) that HierLS with respect to the network size N, but also scales better than it with respect to mobility (HierLS total overhead is linear with mobility). It also shows good scalability with respect to traffic, since it is not linear (as DSR, flooding, and HierLS) but increases only as rapidly as $\sqrt{\lambda_t}$. A more detailed analysis may be found in [16].

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

Similarly, regarding the value of t_e that achieves the minimum overhead, it can be shown (from equation 11) that $t_e = \Theta(\sqrt{\frac{1}{\lambda_t\mathcal{M}}}\,(\frac{L}{\alpha})^{1.5}a\,)$. Thus, the optimal value of t_e is asymptotically independent of the network size depending only on the traffic, mobility, and transmission range. Thus, it is possible to set a value of t_e that works well independently of the network size.

6. SIMULATION RESULTS

The relative performance of the HSLS algorithm compared to SLS, DLS, and NSLS [12] on a integrated system (including radio, channel, and traffic models) has been evaluated from high fidelity simulations conducted using the CPT++ protocol toolkit and OPNET. The performance metric of interest is the *troughput*, which is the percentage of packets successfully received. The *throughput* results reflect the dynamic interaction of several factors, among them the network load : data and total overhead, the sub-optimality of routes (since packets traversing longer paths are more likely to experience a collision at some point along their route), link layer information latencies (e.g. having to wait t_e seconds to get information about a link gone down), routing inconsistencies due to different 'vision' of the network by different nodes, etc. Thus, it is of interest to assest the relative performance of HSLS and other algorithm under non-saturation scenarios. These results complements the previous theoretical analysis, where it was determined that HSLS induced a lower total overhead than other algorithm on the FSLS family and therefore will achieve a higher throughput (in number of bits) under saturation conditions.

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

Simulations were conducted for networks up to 800 nodes. In all of them, nodes were randomly located on a square area of varying size depending upon the density parameter. Each node choose a random direction among 4 possible values and move on that direction at maximum speed.

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

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

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

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

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

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

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

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

Figure 6: Pseudocode description of the Hazy Sighted Link State (HSLS) algorithm.

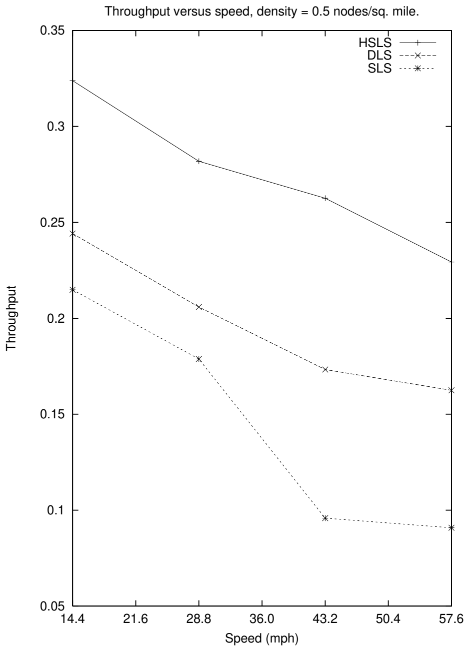

Figure 7: Throughput results for a 80-node network under different nodes' speed.

Algorithm	Throughput
NSLS	0.3516
HSLS	0.4465

Table 1: Throughput for a 800-node network, density = 4 nodes/sq. mile, velocity = 57.6 mph

Figure 7 shows simulation results obtained by CPT++ for a 80-node network with varying nodes' speed. The network density was set to 0.5 nodes per square mile. The radio link capacity was set to 300kbps, and there were 12 source-destination pairs chosen randomly. Each source generated 2048 bits packets with exponential interarrival time distributed around the mean of 1 packet per second (thus, there were 12 2Kbps streams). Figure 7 compares DLS and HSLS with SLS. At this size (and for the given radio link capacity) the performance degradation of SLS – due to its scalability problems – is already noticeable. Thus, SLS was no longer considered for larger size simulations.

Next, the network size was increased up to 400 nodes with 60 source-destination pairs (4 Kbps each). The radio link capacity was increased to 1.676 Mbps to match the Utilicom Longranger 2050 radio modem. [13] The density was increase to 4 nodes per square mile to get similar connectivity as before (transmission range decreases at higher frequencies). The OPNET results (see Figure 8) show that both NSLS and HSLS outperform DLS since they have better scalability properties. Also, at this network size and for this density there is not much difference between HSLS and NSLS and even there are cases where NSLS outperforms HSLS. This is not strange since at this network size, the network diameter is small and NSLS's and HSLS's LSU generation processes are almost the same (most nodes receive the LSUs with TTL equal to 2, as it were 'global'), so that their relative diference is subject to experimental error. Besides, our theoretical results hold for saturation condition (where remaining capacity is the more important factor) while the simulations are based on a lightly loaded scenario. However, as size increases, HSLS lead over NSLS increases and one will expect to see HSLS outperforming NSLS in the simulations.

Further increasing the network size up to of 800 nodes produced the results shown on Table 1. It can be noticed that NSLS's performance degrades significantly while the HSLS performance is still within acceptable levels.

These results not only indicate that HSLS is the best approach among the family of FSLS algorithm, but considering the demanding scenario (60 8Kbps streams under unreliable CSMA) they also show the feasibility of HSLS as an extremely easy-to-implement solution (see Figure 6) for scalability to networks of hundreds of nodes.

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

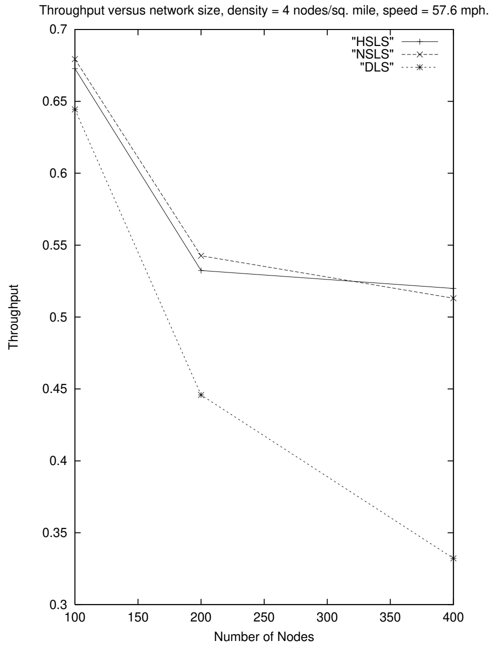

Figure 8: Throughput results for different network sizes.

7. CONCLUSIONS

We considered a class of approaches that attempt to scale link-state routing by limiting the scope of update dissemination in space and over time. This class opens a new design space, since it is not global nor local; representing a new way of thinking where each node may have a different view of the network. We presented the first fundamental analysis of this generic approach, which we called "Fuzzy sighted link-state routing".

Using a novel perspective on the "overhead" of a protocol that includes not only the overhead due to control messages but also due to route sub-optimality, we formulate an analytical model whose solution automatically leads to the best algorithm in this class, namely the HSLS algorithm. This algorithm, although extremely easy-to-implement, has nearly the best possible asymptotic overhead for any routing algorithm – proactive or reactive (see [16]).

Our framework also allows for analysis of different protocols on the literature. This task is undertaken on the sequel (the interested reader may review [16]).

Also, our work presents a new paradigma on the design of routing protocols for mobile ad hoc networks, where it is the overall system performance which take precedence over any other design criterias, and the theoretical analysis precedes the protocol design.

Finally, although our work has been focused on link state routing, it can be easily extended to geographical routing approaches. For example, it was stated that DREAM [10] has similarities with NSLS. Our analysis suggest that DREAM may be improved by employing the same information dissemination algorithm as HSLS instead (of NSLS's).

8. REFERENCES

[1] A.S. Tanenbaum, "Computer Networks", Prentice-Hall, 1996.

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

[3] P. Jacquet, P. Muhlethaler, and A. Quayyum, "Optimized link state routing protocol", IETF MANET Working Group Internet-Draft, Work in Progress

[4] 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.

[5] 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.

[6] 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, Aug. 1999.

[7] G. Lauer, "Packet Radio Routing", in *Routing in Communication Networks*, ed. M. Steenstrup, Prentice-Hall, 1995.

[8] 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.

[9] C. Perkins. "Ad-Hoc On-Demand Distance Vector Routing". MILCOM'97 panel on Ad-Hoc Networks, Monterey, CA, November 3, 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] Z. Haas and M. Pearlman, "The performance of query control schemes for the zone routing protocol," in *ACM SIGCOMM*, 1998.

[12] P. Jacquet and L. Viennot, 'Overhead in Mobile Ad-hoc Network Protocols", *INRIA Research Report 3965, Institut National de Recherche en Informatique et en Automatique (INRIA)*, France, June 2000.

[13] 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.

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

[15] 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*, vol. 17, no. 8, Aug. 1999.

[16] C. Santivanez, "Asymptotic Behavior of Mobile Ad Hoc Routing Protocols with respect to Traffic, Mobility, and Size," *Technical Report TR-CDSP-00-52*, CDSP Center, Northeastern University, Boston, MA, October 2000. Available at http://www.cdsp.neu.edu/info/students/cesar/analysis.ps.gz

[17] C. Santivanez, R. Ramanathan, "Hazy Sighted Link State (HSLS) Routing: A Scalable Link State Algorithm", *BBN technical memo BBN-TM-1301*, BBN technologies, Cambridge, MA, August 2001. Available at http://www.ir.bbn.com/documents/techmemos/index.html