Hazy Sighted Link State (HSLS) Routing: A Scalable Link State Algorithm

Article · January 2003

CITATIONS
64

READS
515

2 authors:

Cesar Santivanez
Pontifical Catholic University of Peru
31 PUBLICATIONS **1,148** CITATIONS
SEE PROFILE

Ram Ramanathan
Raytheon BBN Technologies
113 PUBLICATIONS **9,899** CITATIONS
SEE PROFILE

BBN Technical Memorandum No. 1301

Hazy Sighted Link State (HSLS) Routing : A Scalable Link State Algorithm

Job No. 86563005

August 31, 2001
Rev. March 2003

Prepared for:

Prepared by:

BBN Technologies
10 Moulton St.
Cambridge, MA 02138

Revision List

- Revision 1, March 2003:
  - Remove double definition of variable L in page 13. L is the average distance between a node and its neighbors, as defined in appendix $A.2$, page 39.
  - Typo in expression on last paragraph of section 5.3.3 (page 20) fixed.
  - A-HSLS pseudo-code (Figure 11, page 27) expanded, more detailed.
  - A-HSLS example (Figure 12, page 29) corrected.
  - Improper references to '*NumBlocks*' replaced by '*NumUndecidedBlocks*'.
  - Typo in series expansion of e^{-x} (page 38) fixed.

Hazy Sighted Link State (HSLS) Routing : A Scalable Link State Algorithm

Cesar A. Santiváñez Ram Ramanathan
Internetwork Research Department
BBN Technologies
10 Moulton St.
Cambridge, MA 02138

August 31, 2001

Abstract

In this work, 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 hierarchical link state [5], FSR and GSR (see [6]), etc.

In this report, limited dissemination techniques are considered from a fundamental viewpoint. This treatment is anchored around the following generalized link-state routing approach: send an update every t_i seconds with a network scope of r_i hops. This represents a family of techniques for each combination of instantiations of t_i and r_i. The family includes many intuitively feasible and useful techniques, including traditional link-state routing. In the context of this generalized approach, the problem of instantiating t_i and r_i so that the performance is optimized is considered. 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 the optimization 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. A new definition of "overhead" that includes not only the control message overhead but also the cost of sub-optimal routing is proposed. Such a definition facilitates fair comparison of protocols not only within the fuzzy-sighted family, but also amongst previously published protocols.

This work contributions include the following. A new design space for link state routing protocol is opened by presenting a family of (potentially scalable) algorithms that are neither global nor local, but where each node may have a different view of the network. A new definition of overhead that allows for comparison among different protocols is introduced. An analytical model that facilitates the study of a large class of routing protocols is presented.

In particular, a unique feature of the present 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 focus-

ing 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.

While the report addresses routing for ad hoc networks, the paper is also applicable to scalable QoS routing in wired networks. Although the links in wired networks are stable, QoS routing requires that the delay and residual link capacity be advertized throughout the network, resulting in overhead problems. The fuzzy sighted philosophy introduced in this report holds potential for reducing this overhead while maintaining adequate information for routing. In general, any network problem that presents a tradeoff between information dissemination accuracy and overhead is amenable to the techniques and analysis presented here, albeit with some modifications.

The remainder of this report 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 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 discusses some implementation issues and present the algorithmic description of the proposed protocol. Section 7 complements the analysis with simulation results. Finally, section 8 presents our 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.

[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.

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.

The present work is unique in several ways. First, the analysis considers all the different sources of overhead in a unified framework. Second, the usual requirement on proactive approaches that all the nodes must have a consistent view of the network is relaxed. Third, the 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 work is the only attempt (to the author's knowledge) to theoretically understand the limits on scalability for large mobile ad hoc networks.

3 A 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

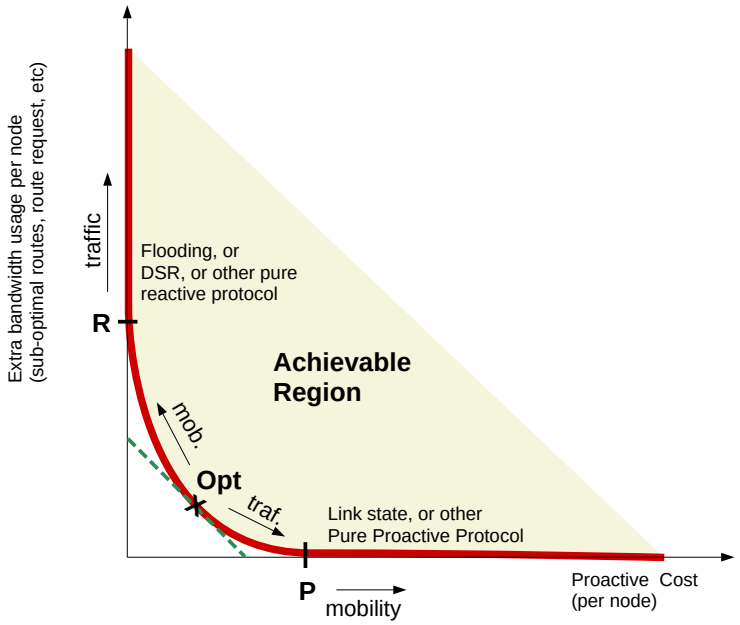

Figure 1: Overhead's achievable region.

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

[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.

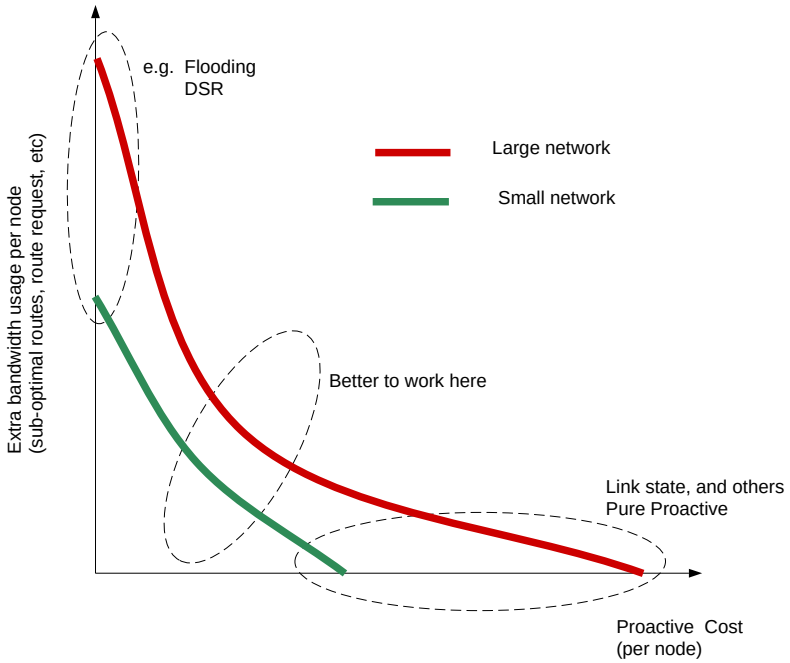

Figure 2: Change in achievable region due to size.

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 $x + y = constant$).

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 Hierarchical link state (HierLS) and ZRP, may increase more slowly with respect to N. In [16] it is shown that under assumptions a.1-a.8 (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 in 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

[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 $n \geq n_0$.

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]

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 (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 a odd number) the link state change activity during the past $4t_e$ seconds needs to be checked and, if there is any, then an LSU with TTL set to s_3 will be sent. Thus, in the highly mobile scenario assumed on figure 3, 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 reference 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) (see Figures 5 and 6) are discussed next.

DLS is obtained by setting $s_i = \infty$ for all i (see Figure 5 left). DLS is similar to the Standard Link State (SLS) algorithm and differs only in that under DLS a LSU is not sent immediately after a link status change is detected but only when the current t_e interval is completed. Thus, several

[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.

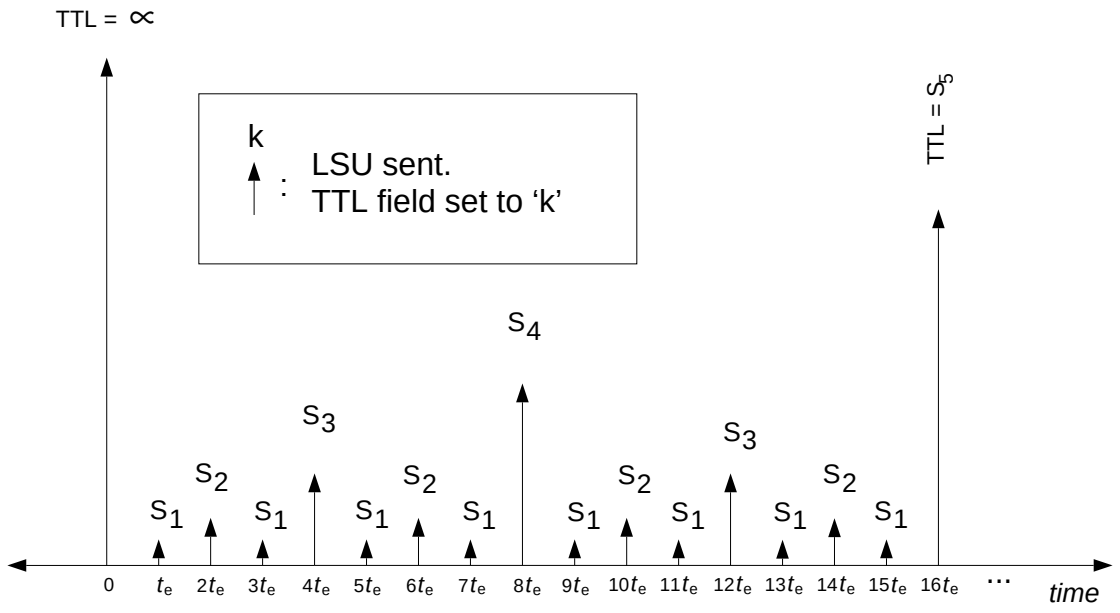

Figure 3: Example of FSLS's LSU generation process

link status changes may be collected in one LSU. DLS is a modification of SLS that attempts to scale better with respect to mobility. Under high mobility, DLS presents some similarities with Global State Routing (GSR)[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. $T(r) = t_e$ (see Figure 6 left), while GSR's increases linearly with distance (see Figure 7 left), 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 $i¡p$ and $s_p = \infty$ (for some p integer), as shown in Figure 5 right. [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 sight area), but it

[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.

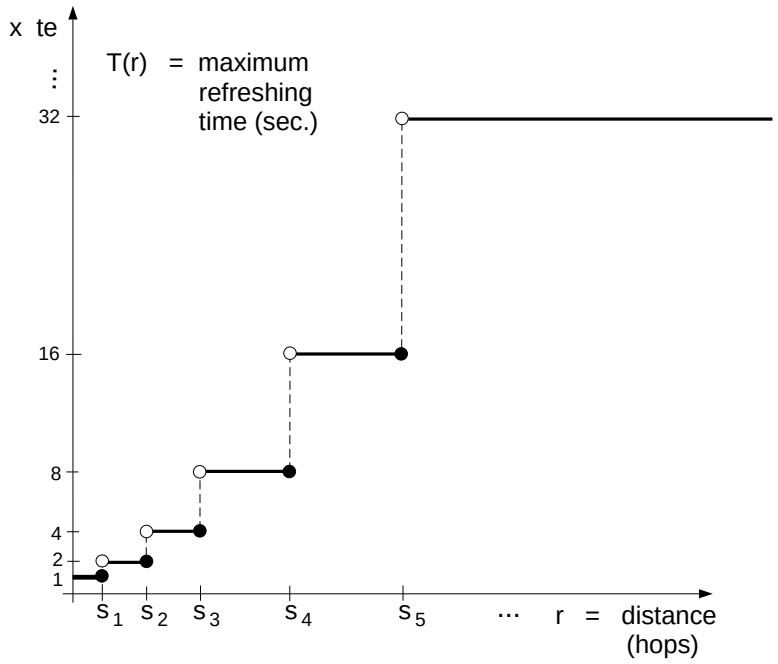

Figure 4: Maximum refresh time $T(r)$ as a function of distance from link event.

is not refreshed with new link state updates from nodes out-of-sight. 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 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-hop neighborhood to decide whether to reply to the route request or to forward it to its own set of

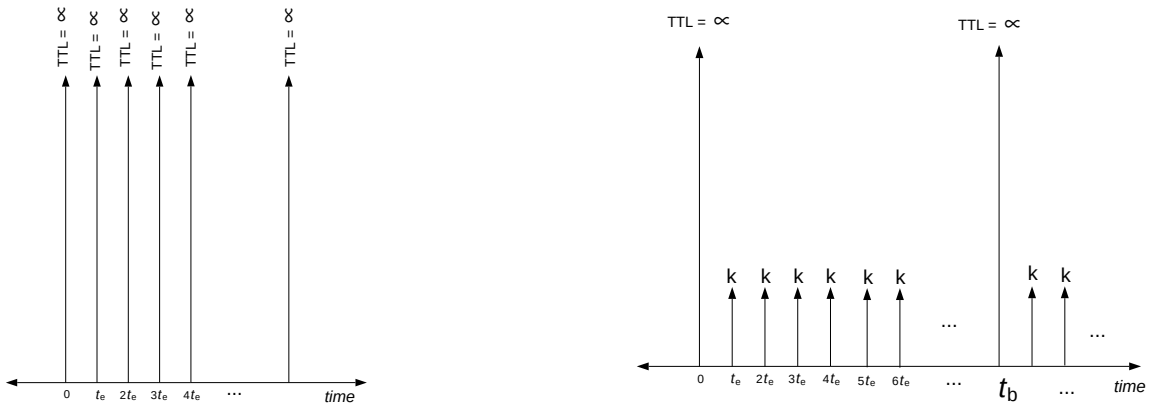

Figure 5: DLS's (left) and NSLS's (right) LSU generation process.

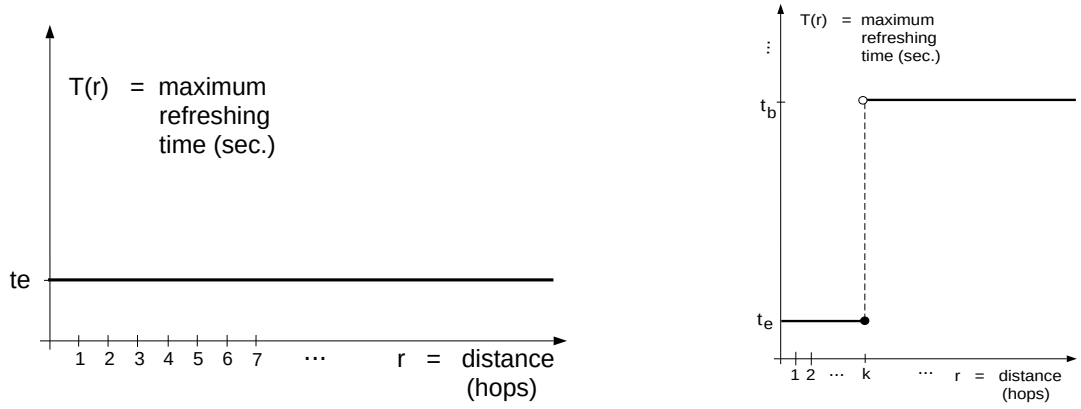

Figure 6: DLS's (left) and NSLS's (right) maximum refresh time $T(r)$ as a function of distance from link event.

'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

[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.

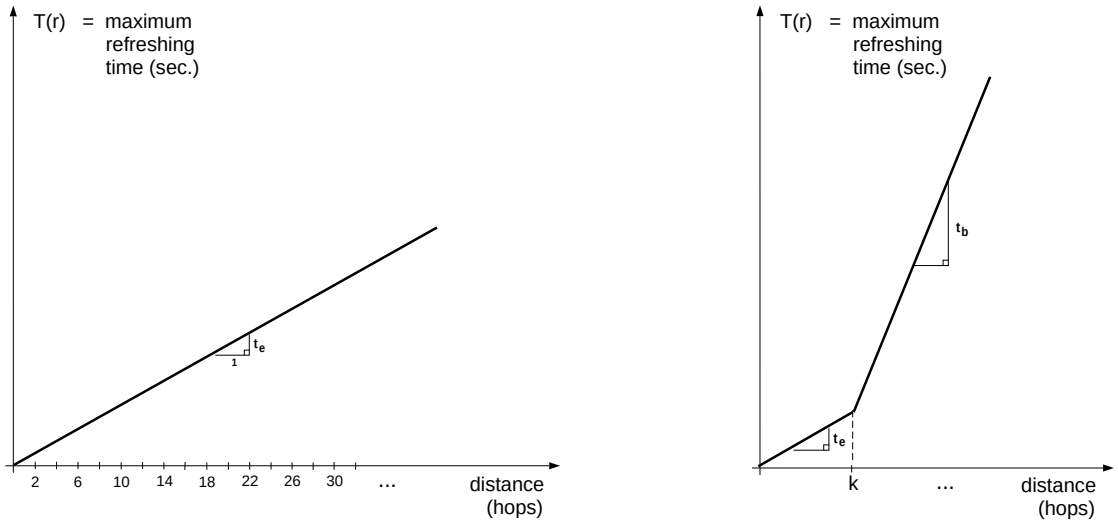

Figure 7: GSR's (left) and FSR's (right) maximum refresh time $T(r)$ as a function of distance from link event.

greater in FSR than in NSLS. In NSLS, $T(r) = t_e$ for $r \leq k$, and $T(r) = t_b$ for $r > k$, as shown in figure 6 (right). In the other hand, in FSR, a LSU have to wait at most t_e seconds (in average $\frac{t_e}{2}$) to be propagated one more hop away from the node experiencing the link event while it is in scope ($r \leq k$), and wait t_b seconds when it is 'out-of-scope' (i.e. $r > k$). Thus, for FSR $T(r) = t_e * r$ for $r \leq k$, and $T(r) = k * t_e + (r - k) * t_b$ (see Figure 7 right), which is significantly larger than the values for NSLS.

Finally, the family of Fuzzy Sighted Link State algorithms is based on the observation that nodes that are far away do not need to have complete topological information in order to make a good next hop decision, thus propagating every link 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 reference (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, λ_{lc} be the expected number of link status changes that a node detects per second, λ_t be the average traffic rate that a node generates in a second (in bps), and λ_s be the average number of new sessions generated by a node in a second.

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

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

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

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

a.4 The maximum and average path length (in hops) among nodes in a connected subset of n nodes both increase as $\Theta(\sqrt{n})$. In particular, the maximum path length across the whole network and the average path length across the network increase 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 the behavior of users, 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*. 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 reference node S runnning a generic FSLS algorithm under high mobility has been derived in Appendix A, and is reproduced here for clarity :

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

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 minimizing 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

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

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

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

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

$$E_1 = K^2 E_2 \tag{4}$$

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

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

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:

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

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. [8] 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 needed 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 we explore the performance degradation (compared to the relaxed case) experienced when K is fixed to 2.

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

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 :

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

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 :

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

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

and the value of t_e that achieves this minimum is :

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

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

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 8 It can be noted that there is an almost linear relationship between $T(r)$ and r. This linear relationship is responsible for HSLS's probable optimality for the central node studied in the previous subsection. This relationship reflects the fact that when forwarding packets to nodes far away, it is the angular displacement what really matters.

Thus, HSLS successfully balances refresh periods and distances, so that the probability of making a suboptimal (bad) next hop decision is roughly the same for every destination independently of the distance [9]. This balance is natural (avoiding 'hard' boundaries as in NSLS where a

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

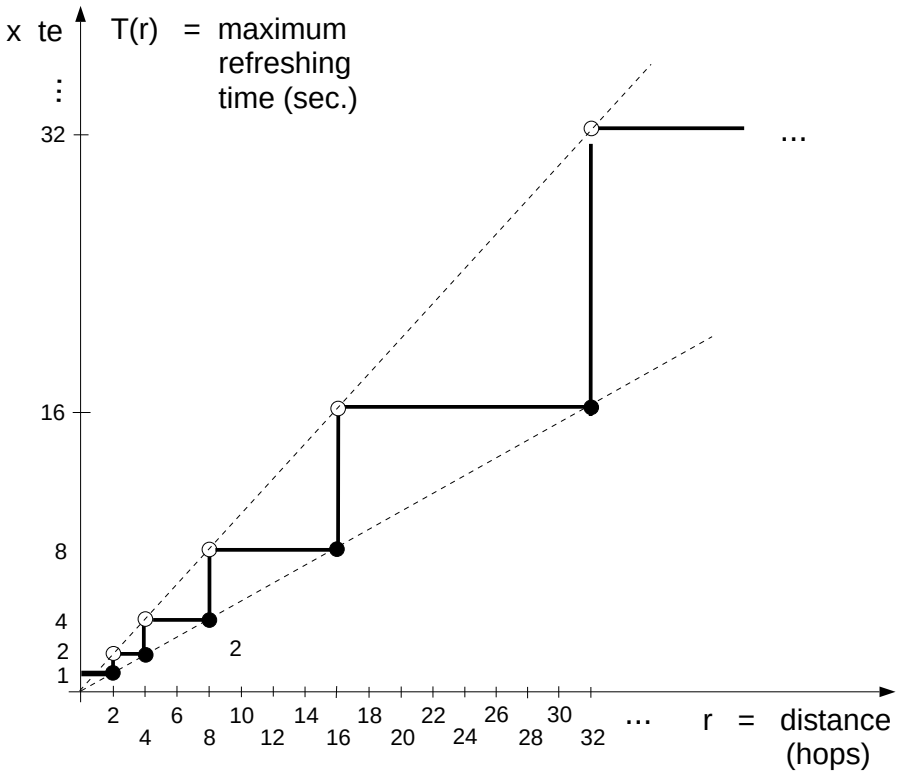

Figure 8: HSLS's maximum refresh time as a function of distance from link event.

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 11. 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 11). Also, the previous analysis – based on geographical reasoning – fails to capture the dynamics inside the 'topology region', that is, small scales. For practical implementations it was found through simulations that LSUs with small TTL do have a great impact 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. [10] This does not contradicts the theoretical analysis, that did not care about them. Further discussion about implementation issues is deferred to section 6.

[10] 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.

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 :

$$HSLS_{total} = \Theta((\frac{L}{\alpha})^{2.5}\lambda_t^{0.5}\mathcal{M}^{0.5}N^{1.5}) \quad (12)$$

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

6 HSLS's implementation issues

So far, several assumptions has been made during the analysis. In order for this assumptions to be valid, some additional mechanisms must be in place. These mechanisms are discussed next. Then,

this section finishes with an algorithmic description of HSLS.

6.1 Loops and level 1 LSUs

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

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

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

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

Thus, in HSLS level 1 LSUs are send frequently. In our particular implementation, however, there was no need for sending these LSUs explicitly, since the HELLO packets exchanged by the neighbor discovery module (link layer) provided these information in a timely manner. The HELLO

packets contained the id of the node sending the beacon, as well as a list of the neighbors that node could listen to. For each neighbor, and indicator whether the link was unidirectional (incoming) or bidirectional (incoming and outgoing) was included, along with the number of beacons received from that neighbor during the last time window. A node receiving the HELLO message will not only determine if it has an incoming link from the node sending the beacon, but it may look for its own id in the node list of neighbors, and if present, determine that the link to/from that neighbor is bidirectional. The indication of the number of beacons received, when eavesdropped by the routing module, allows to quickly detect degrading links and to use alternative paths when available. Thus, quick reaction to local changes was achieved.

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

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

6.2 Topology table maintenance

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

Thus, it is important that a node 'extracts' the most information from the LSU. For example, when the LSUs reports status of incoming links (i.e. when supporting unidirectional links), reception of a LSU from a node, say A declaring an incoming link from another node B may provide also information about incoming links to B. If node A's LSU declare the incoming link from node B as bidirectional (node A is aware of its one hop neighborhood topology), thus it must be inferred that node B also has a bidirectional link from node A. Alternatively, if the link from node A is declared to be unidirectonal (incoming only), thus it is inferred that node B does no longer have an incoming link from node A, and if such an entry exists in the topology table's entry for node B, it should be removed. In this case, we are relying on the information conveyed by node A since it has more up-to-date information than us (because it is closer – one hop away – from node B, and receiving level 1 LSUs or beacons).

Figure 9 presents the pseudocode employed in our implementations for updating the topology table upon reception of a LSU from node A. It is based on LSUs conveying incoming link information in order to support unidirectional links (however, unidirectional links were assigned a larger cost than bidirectional ones, and were used only if there were no bidirectional path available). Two

details should be observed. First, since the main principle is to relay on the information provided by the node with more up-to-date information, priority is given to the receiving node's own local neighborhood information. Thus, any conflict between the information provided by node A, and the receiving node local information (as provided by level 1 LSUs and beacons) is resolved in favor of the receiving node local information, which is more up-to-date since the beacons mentioned in the previous subsection are interexchanged frequently. Note, however, that there is no conflicting information, as for example a report from node A about an unidirectional incoming link from the node executing the algorithm. The executing node's local information, coming from the beacons it received, does not provide info about that node's unidirectional outgoing links. The second detail is more subtle, but not for that less important. Note that there are situations where we learn that a link to a third node, say C, is no longer there and should be removed. However, since we may not receive an update from node C for a while we may not be able to find a new, alternative route to it. However, nodes closer to node C will have fresher information and may be able to find routes to it. Thus, the solution is to send the packets destined to node C in the direction node C used to be, and hope that along the way a node with current information deliver the packet. The above is accomplished by not removing the link from node A to node C but elevating its cost to the maximum allowed value, so that that link will not be used if there is an alternative path. However, in the case where there are no alternative paths toward node C (the case we were woried about), the link from node A to node C will be used by route computation algorithm and determine that packets to node C be forwarded in the direction of nodes C and A, as desired. Thus, keeping those links (e.g. link A to C) allows to keep some memory and track a node shadow as it moves farther away from us.

6.3 Low mobility scenarios

So far, the discussion has been focused on highly mobile scenarios. It is beenm shown in equation 12 that HSLS total overhead increases as $\Theta(\sqrt{\mathcal{M}}) = \Theta(\sqrt{\lambda_{lc}})$, which is a significant improvement over SLS whose control (total) overhead increases linearly with λ_{lc}.

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

To fulfill those requirements, Adaptive HSLS (A-HSLS) was created. A-HSLS design was motivated by the observation that under extreme low mobile scenarios, when link changes were unfrequent, HSLS will require to send a series of LSUs with TTL equal to 2, 4, 8, and so for until a global LSU was sent, for each link change. In those cases it would have been more efficient to just send a global LSU to begin with, had we know that no other link change would follow.

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


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

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


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


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

Figure 9: Pseudocode for topology table update upon reception of a LSU from node A.

Unfortunately, there is no way to know for certain whether more link state changes will follow a particular one or not.

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

Under SLS, a node will generate LSUs at a rate of λ_{lc} LSUs per seconds, and each LSU will require N retransmissions. Thus, the node will induce a control overhead of $\lambda_{lc}N$ LSUs per second.

Under HSLS, the control (proactive) overhead induced by a single node, say X, is roughly $\frac{N}{R_x t_e}(1 + 2f_x)$ LSUS per second. Where R_x is the time index at which node X sends global LSUs, that is $R_x < MD_x \leq 2R_x$, where MD_x is the distance from node X to the node farthest apart according to node X's topology table. And f_x is a factor that varies between 0.25 and 1.0. The

above expression for HSLS control overhead is obtained by grouping the LSU transmission based on the original TTL field of the LSUs (i.e. the TTL value set at the LSU packet generation). Thus, each $R_x t_e$ seconds node X sends LSUs with TTL equal to $2R_x$ that reaches the entire network (N retransmissons) inducing a control overhead of $\frac{N}{R_x t_e}$ LSU retransmissions. At times $\frac{R_x}{2} t_e\, k$, with k is odd (i.e. also each $R_x t_e$ seconds), node X will send LSUs with TTL equal to R_x that will reach to a fraction f_x of the nodes in the network. [11] inducing a control overhead of $\frac{f_x N}{R_x t_e}$ LSU retransmissions. In general, at times $\frac{R_x}{4} t_e\, k$, $\frac{R_x}{8} t_e\, k$, etc. a node will send LSUs with TTL set to $\frac{R_x}{2}$, $\frac{R_x}{4}$, etc. respectively. Since for these LSUs boundary effects may be safely ignored, we can observe that as the original TTL field reduced by half, the composite effect is to reduce by one half the LSU retransmissions (control overhead) induced by the corresponding LSUs (i.e the generation rate doubles but the number of retransmissions is reduced by a factor of $2^2 = 4$. Then, HSLS's control overhead is equal to $\frac{N}{R_x t_e} + \frac{f_x N}{R_x t_e} + \frac{f_x N}{2R_x t_e} + \frac{f_x N}{4R_x t_e} + \ldots = \frac{N}{R_x t_e}[1 + f_x(1 + \frac{1}{2} + \frac{1}{4} + \ldots)] \approx \frac{N}{R_x t_e}[1 + 2f_x]$, as previously stated.

Now, equating SLS's and HSLS's proactive overheads, the link change rate (λ_{lc}) at which they become equivalent is obtained as follows:

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

Where the last equallity was obtained assuming a typical value of 0.5 for f_x. Thus, the link change rate at which HSLS and SLS induce the same proactive overhead is $\lambda_{lc} = \frac{2}{R_x t_e}$, and therefore the threshold used by A-HSLS is equal to $\frac{R_x t_e}{2}$.

Finally, in Appendix B it is shown that A-HSLS control overhead induced by a node is equal to equal to $\frac{1+2pf_x}{pR_x t_e + \frac{1}{\lambda_{lc}}} N$, where $p = 1 - \exp^{-\frac{\lambda_{lc} R_x t_e}{2}}$. Figure 10 compares this control overhead with SLS's and HSLS's for the case $f_x = 0.5$. It may be seen that SLS's and HSLS's control overhead coincide for $\lambda_{lc} = \frac{2}{R_x t_e}$. It can be noted that even for extremely low rate of link changes, A-HSLS induces the similar control overhead as SLS. For not-so-small rates of change, however, A-HSLS tends to converge to $\frac{2N}{r_x t_e}$, while SLS control overhead increases unbounded. In high mobility scenarios, A-HSLS proactive overhead will be equal to HSLS's.

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

[11] From assumption a.3, f_x should be around $(R_x/MD_x)^2$, i.e., $f_x \in [0.25, 1]$. In practical situations, due to boundary effects (i.e. the number of nodes at a maximum distance MD_x is small), we obtain that typically f_x is in the interval [0.5, 1].

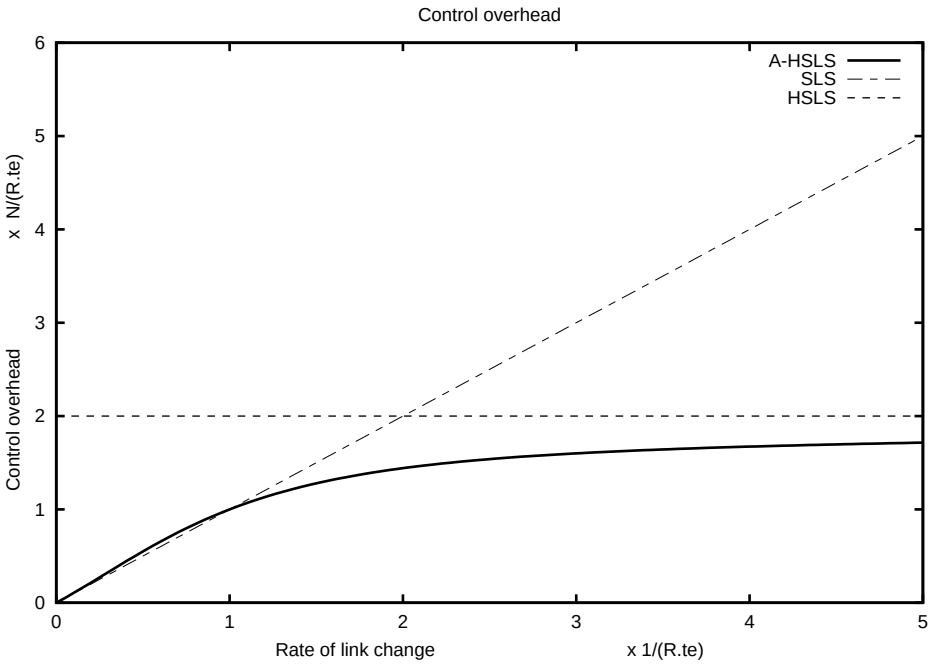

Figure 10: Control (proactive) overhead for A-HSLS, SLS, and HSLS

.

6.4 Protocol description

After the last subsections discussion, we are finally in position to discuss HSLS (more precisely A-HSLS) pseudo-code shown in Figure 11.

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

Topology table maintenance pseudo-code (Figure 9 has been discussed in subsection 6.2, thus, further discussion is not necessary.

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

Thus, the remainder of this subsection discusses the LSU generation algorithm presented in Figure 11.

Upon initialization (sending of the first global LSU and resetting of all counters and timers), a node running HSLS (A-HSLS) is in the UNDEC (Undecided) mode. After information about link change rates have been obatined the node will transit to either SLS (for Standard Link State -like) or HSLS mode. If the node is in SLS it means that it regards its local topology as slow changing

```
initialization:
    set timer t_p to small value & set mode = UNINIT


timer t_e expires:
    if (mode == SLS) then return
    reset timer t_e
    compare current LSU in TopoTable with LastLsuSent
    if (change) then 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: NumUndecidedBlocks++
            if (change) then
                        SendLSU(ttl = 2)
                        set mode = HSLS & NumEventInt= 1
                    else if (NumUndecidedBlocks >= R/2) then 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 ) then
                if ( 2^i < R) then SendLSU(ttl = 2^(i+1))
                              else SendLSU(ttl = ∞)


timer t_p expires:
    SendLSU(ttl = ∞)


Link state change occurs :
    update TopoTable
    if (mode == UNINIT) then return
    if (NumUndecidedBlocks == 0) then SendLSU(ttl = 1)
        else switch (mode)
            case(SLS)   : SendLSU(ttl = ∞)
            case (HSLS) : SendLSU(ttl = 1)
            case(UNDEC) : SendLSU(ttl = 2)
                          set mode = HSLS & set NumEventInt = 1
                          reset timer t_e


void SendLSU(int ttl)
    send LSU with TTL set to ttl
    if (ttl >= 2) then set LastLsuSent = current LSU
    if (ttl == ∞) then reset_everything()


void reset_everything()
    set mode = UNDEC and all counters to zero.
    (re) set timers t_e and t_p (to normal, large value)
```

Figure 11: Pseudocode description of the Adaptive Hazy Sighted Link State (A-HSLS) algorithm.

and the next time a link change is detected it will send a global LSU as in the SLS algorithm. If the node is in HSLS mode, it will send LSUs with values $2, 4$, and so on at time instants that are multiple of t_e. It will also send level 1 (TTL equal to 1) LSUs for changes that happens between those time instants.

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

In the other hand, if a link change is detected before Rt_e seconds have elapsed since the last global LSU, the node enters HSLS mode by sending a LSU with TTL equal to 2 and setting the *NumEventInt* counter to 1. This counter (*NumEventInt*) will be incremented by one each time the timer expires (each t_e seconds) while in HSLS mode. In HSLS mode, the node will propagate changes so that nodes that are less than 2 hops away are refreshed each t_e seconds, nodes that are less than 4 hops away are refreshed each $2t_e$ seconds, nodes that are less than 8 hops away are refreshed every $4t_e$ seconds, and so for. To achieve the behavior just described, a node computes the maximum power of 2 that is an exact divisor of the *NumEventInt*, let *window_len* be this value, thus for example if $NumEventInt = 20$, then $window\_len = 4$, since 4 divides 20 and 8 does not. Upon determination of *window_len*, the node determines that it should check for link changes in the past $window\_len\, t_e$ seconds, and if any, it should send a LSU with TTL equal to $2\, window\_len$. Also, if the TTL value ($2\, window\_len$) is greater or equal to $2R$ (i.e. reaching the entire network), the TTL value is overwritten with the infinity value, thus a global LSU is sent and the algorithm is reinitiated, i.e. switches back to UNDEC mode, and reset all timers and counters. Thus, in figure 11, the node checks whether the *TimeSinceLastChange* is greater than $window\_len\, t_e$ seconds, and if not, it sends a LSU as explained before.

Figure 12 shows an example of (A-)HSLS's LSU generation process. An 'x' marks the time when link status changes are detected. A vertical arrow signifies that a LSU with the specified TTL was sent. For clarity sake, level 1 LSUs (discussed) next) are not shown. The example xhown in this figure assumes that MD is somewhere between 9 and 16, so that R is equal to 8. We can see that the first link change after initialization (global LSU sent) happens after $R/2 = 4$ time intervals, i.e. when the node is in SLS mode. Thus, the node send a global LSU (TTL $= \infty$) and it is reset. The next link change happens before the node enters SLS mode. Since the time elapsed between the

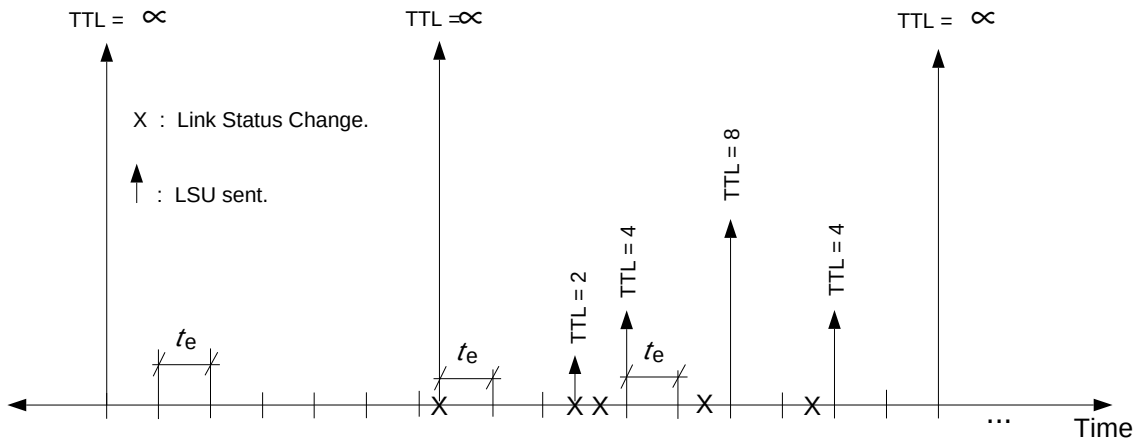

Figure 12: Example of A-HSLS's LSU generation process.

global LSU transmission and the link change detection is greater than t_e seconds, the node does not wait any time before sending a LSU with TTL equal to 2. The next time interval, th enode wakes up and update the value of *NumEventInt* to be equal to 2. Th enode then checks for link changes in the last $2t_e$ seconds, and since there have been link status changes, it sends a LSU with TTL equal to 4. The next time interval, the value of *NumEventInt* is equal to 3, so th enode check for link status changes in the past t_e seconds. Since there was no change, no LSU is transmitted. For the next time interval, *NumEventInt* is equal to 4 so the node check for changes in the past $4t_e$ seconds and sends a LSU with TTL equal to 8. For the next time interval, *NumEventInt* is equal to 5 so the node check for changes in the past t_e seconds, and since there was no change, no LSU is transmitted. Continuing for *NumEventInt* equal to 6, the node checks for changes in the past $2t_e$ seconds, and since there has been a link change, it sends a LSU. For *NumEventInt* equal to 7, there were no link change in the past t_e seconds so no LSU is sent. Finally, for *NumEventInt* equal to 8, the node checks for changes in the past $8t_e$ seconds, and since changes did happen it will send a LSU. However, since the corresponding TTL value 16 is greater than MD, the TTL field is overwritten with the value infinity and the algorithm is reinitiated in UNDEC mode.

Regarding the treatment of level 1 LSUs, while in HSLS mode each time the node detects a link change it will send a level 1 LSU. [12] The first link change detected (the node is still in the UNDEC mode) will receive special treatment. If the time elapsed between the global LSU transmission and the link change detection is greater than t_e, a LSU with TTL equal to 2 will be sent immediately and the node will switch to HSLS mode. However, if the elapsed time is lower than t_e (fast changing case), the node will wait until expiration of the t_e timer before sending the LSU with TTL equal to 2. A LSU with TTL equal to 1 will be send instead. This way, multiple

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

changes will be collected before sending a more costly LSU.

Finally, a global LSU are sent if the time elapsed since the last global LSU transmitted exceeds a predefined threshold (not shown in Figure 12 for clarity sake. This is equivalent to the periodic LSUs in SLS.

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

7 Simulation results

The relative performance of the HSLS algorithm compared to SLS, DLS, and NSLS [13] 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 *throughput*, which is the percentage of packets successfully received. The *throughput* results reflect the dynamic interaction of several factors, among them the network load : data and total overhead, the 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.

Figure 13 (left) shows simulation results obtained by CPT++ for a 80-node network with varying nodes' speed. The network density was set to 0.5 nodes per square mile. The radio link capacity was set to 300kbps, and there were 12 source-destination pairs chosen randomly. Each source generated 2048 bits packets with exponential interarrival time distributed around the mean of 1 packet per second (thus, there were 12 2Kbps streams). Figure 13 (left) compares DLS and

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

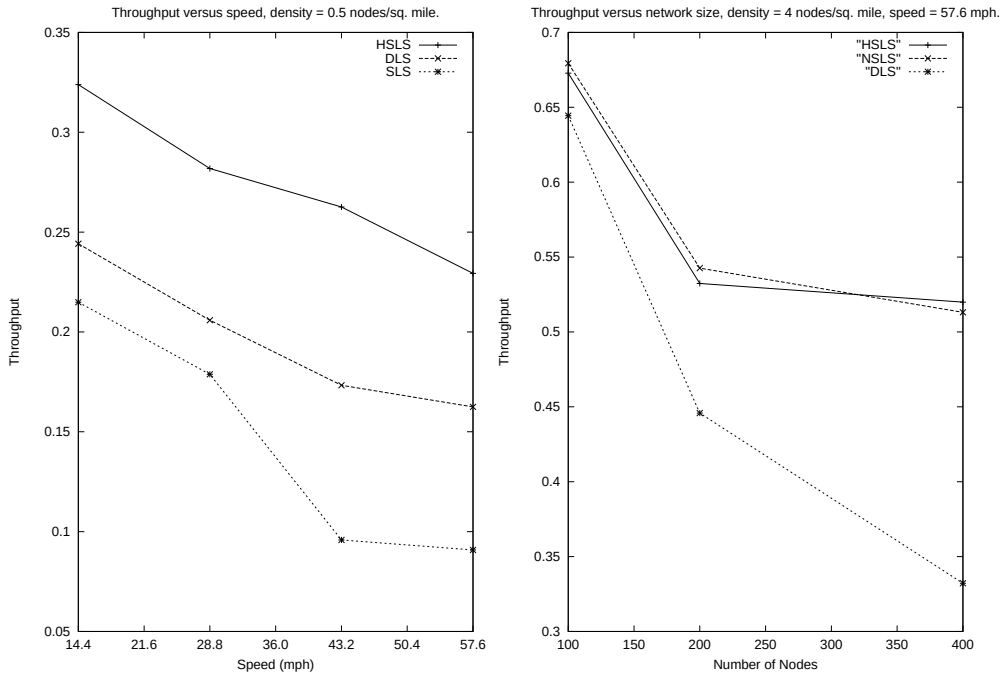

Figure 13: Throughput results for a 80-node network under different nodes' speed (left), and for different size networks (up to 400 nodes) (right).

HSLS with SLS. At this size (and for the given radio link capacity) the performance degradation of SLS – due to its scalability problems – is already noticeable. Thus, SLS was no longer considered for larger size simulations.

Next, the network size was increased up to 400 nodes in an OPNET simulation with 60 source-destination pairs (4 Kbps each). The radio link capacity was increased to 1.676 Mbps to match the Utilicom Longranger 2050 radio modem. [14] 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 show in Figure 13 (right) indicates 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.

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

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

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 4Kbps streams under unreliable CSMA) they also show the feasibility of HSLS as an extremely easy-to-implement solution (see Figure 11) for scalability to networks of hundreds of nodes.

8 Conclusions

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

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

Also, this work presents a new paradigm 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 criteria, and the theoretical analysis precedes the protocol design.

In addition, although this 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. This report's analysis suggest that DREAM may be improved by employing the same information dissemination algorithm as HSLS instead (of NSLS's).

A Approximate Expression for FSLS's Total Overhead

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

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

A.1 FSLS proactive overhead

Consider Figure 3 and a highly mobile environment so that and LSU is generated every time interval. Let $s_1 \leq s_2 \leq s_3 \leq \ldots \leq s_n = R$. Thus, every $2^{n-1} * t_e$ seconds the algorithm is reset.

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

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

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

Adding together the types 1 to n LSUs, letting $size_{LSU}$ be the average size of a LSU packet, and assuming $c_i \approx c$ the following is obtained:

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

A.2 FSLS sub-optimal routing overhead

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

Node S induces sub-optimal routing overhead each time it forwards a data packet (whether it is the originating source or not) to a node that is not along the shortest path to the data packet's destination. For example, let's assume that node S receives a data packet destined to node D, which is r hops away. If node S forwards the packet to a node $r-1$ hops away from D then no sub-optimal routing overhead will be induced. However, if node S forwards the packet to a node that is also r hops away from D, then a sub-optimal routing overhead equivalent to the size of the data packet will be induced. In general, if node S forwards the packet to a node that is $r+j$ hops away from D [15], then a sub-optimal routing overhead equivalent to $j+1$ times the size of the data packet will be induced.

To analyze the sub-optimal routing overhead induced by node S, consider Figure 14 where a snapshot of the network, as seen by node S, is presented. It is assumed that all the nodes are within R hops from S. The network shown in Figure 14 presents a great deal of uniformity but it is not our intention to limit our analysis to just such a regular network. The regularity observed in Figure 14 just represents the *average* or *expected* values based on assumptions a.1 through a.4 (subsection 5.1). Thus, the network deployed in Figure 14 reflects our expectation on average values for networks with the same density σ (i.e. $c_1 * r^2$ nodes at a distance of r hops or less from S, and $c_2 * r$ nodes exactly at a distance of r hops).

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

To compute the average sub-optimal routing overhead induced by S at time t, consider that every time node S receives a packet destined to a node, say D, that is r hops away, it may or may not make a proper next hop decision. Let $p(S, x, y, t)$ be the probability that node S's next hop decision for flow $x \rightarrow y$ at time t is non-optimal. It is further assumed that if a non-optimal next hop decision is made, the packet is forwarded to a node that is at the same distance from D as node S, and that the flows are loop free. The first assumption is based on the observation that sending a data packet to a node that is $r+1$ or more hops away from the destination will be equivalent to sending the packet on the direction totally opposite to the destination. While the above event is possible, it is highly unlikely during normal protocol operation (i.e. before the protocol breaks) for dense networks, specially when the destination is far away and geographical constraints dominate the routing decisions (i.e. the 'geographical' region discussed in subsection 5.1). Thus, ignoring these unlikely events will provide us with a first order approximation on the network performance. The second assumption is based on the same rare-event reasoning towards a first order approximation.

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

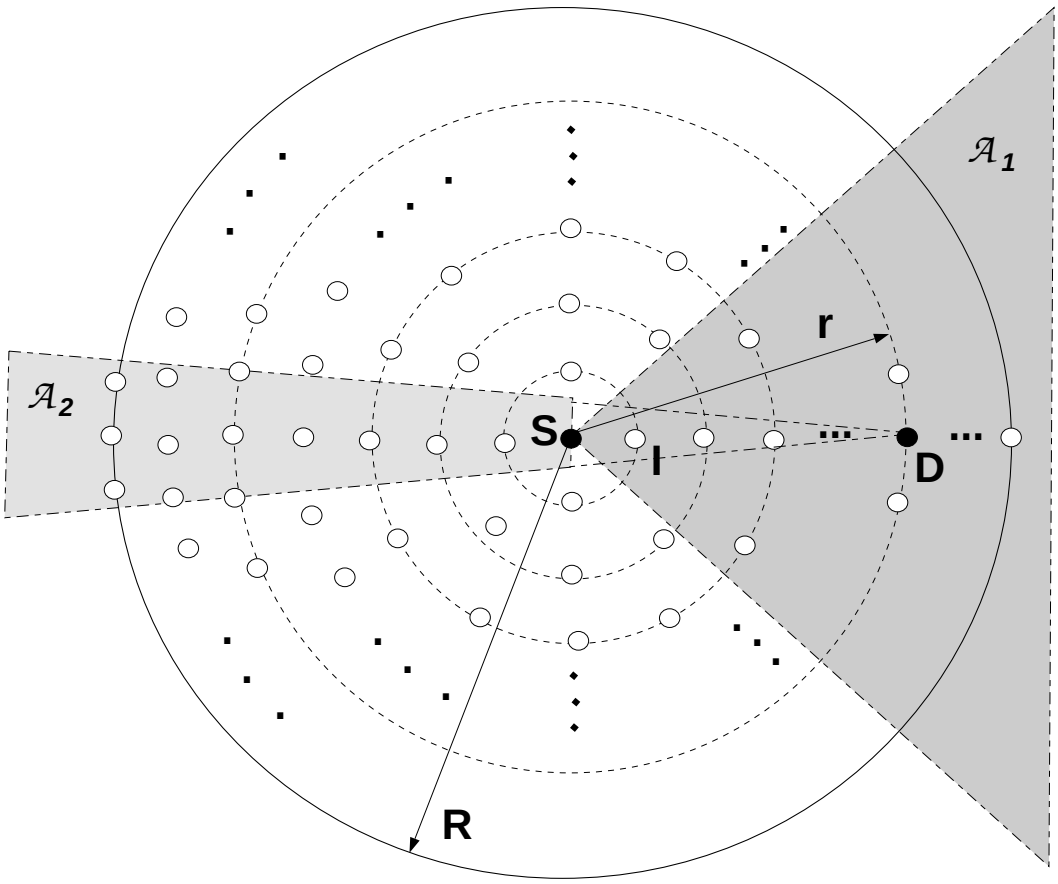

Figure 14: A network snapshot, as seen by node S, running a FSLS algortihm.

Also, loop detection/removal mechanisms are relatively easy to implement. It should be noted, however, that loops are less rare when the packet is close to the destination, since in this region it is topology and not the underlying geography what dominates the routing decision. Thus, it is important that the selected FSLS strategy try to minimize these loops (this topic will be revisited in subsection 6.1).

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

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

Let $N_r(S, t)$ be the set of nodes that, according to node's S view of the network at time t, are exactly at a distance r (in hops) from node S, and let $\mathcal{F}_t(S, D)$ be the set of flows passing through S and ending at D (destination) at time t. Thus :

$$S_{sub}(t) = \frac{\lambda_t}{N} \sum_{r=1}^{R} \sum_{D \in N_r(S,t)} \sum_{x \rightarrow D \in \mathcal{F}_t(S,D)} p(S,x,D,t) \quad (13)$$

$$= \frac{\lambda_t}{N} \sum_{r=1}^{R} \sum_{D \in N_r(S,t)} |\mathcal{F}_t(S,D)| \; p(S,x,D,t) \quad (14)$$

Where the last equality comes from the observation that once a packet reaches node S to be forwarded to another node D, the packet's history is not relevant, and all such packets will be forwarded in the same manner regardless of the packet's origin.

To compute $p(S,x,D,t)$, let's consider Figure 14 again. Node S will forward any data packet destined to node D toward node I, the next hop according to node S's view of the network. In Figure 14 it can be seen that such a decision of forwarding the data packet through node I will be optimal if (and only if) node D is currently inside the area $\mathcal{A}_1$. However, since the figure represents node S's view, and this node (S) may not have received any update from D in the past $T(r)$ seconds (see section 4, on maximum refreshing time), node D's current position may have changed.

Let t_1 be the most recent time when node S received a LSU from node D. Let $p(S,x,D,t,t_1)$ be the probability of node S taking a non-optimal next hop decision when forwarding a packet to D through node I at time t, given that the last LSU update from D was received at time t_1. Since t_1 may be any time less than $T(r)$ seconds ago (in the interval $< t - T(r),\ t >$), [16] then:

$$p(S,x,D,t) = \frac{1}{T(r)} \int_{t-T(r)}^{t} p(S,x,D,t,t_1) \; dt_1 \quad (15)$$

$p(S,x,D,t,t_1)$ is equal to the probability of node D leaving the area $\mathcal{A}_1$ since time t_1 in Figure 14 [17]. This probability depend on the node speed and the mobility model assumed. It is similar to the problem of residence time in cellular networks, where it is common practice to assume an exponential residence time. In our setting, $p(S,x,D,t,t_1)$ is the probability that the residence time inside the area $\mathcal{A}_1$ is less that $t - t_1$. Thus, assuming also an exponential model for the residence time [18], $p(S,x,D,t,t_1) \approx 1 - e^{-\frac{\mathcal{M}(t-t_1)}{r}}$. $\mathcal{M}$ is a constant that depends on the average

[16] At time t_1, node D was r hops away from node S. If node D stays r or less hops away from S, it will 'refresh' node S at most after $T(r)$ seconds. Thus, in this case, if t were greater than $t_1 + T(r)$ node D would have 'refresh' node S about its new location, which would contradict the assumption that t_1 is the most recent time when node S received a LSU from node D. However, strictly speaking, node D may have moved more than r hops apart from node S since time t_1. It will be assumed that node D is still *close* to r hops away, and that the above refreshing time is approximately correct. This assumption is further backed by the fact that function $T(r)$ remains constant under small variations of r (see Figure 4).

[17] Strictly speaking, area $\mathcal{A}_1$ is not necessarily a triangle. Figure 14 just shows our expectation that on average that area will be a fraction of the network. The area of $\mathcal{A}_1$ depends more heavily on the number of neighbors of node S than on the distance from D to S.

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

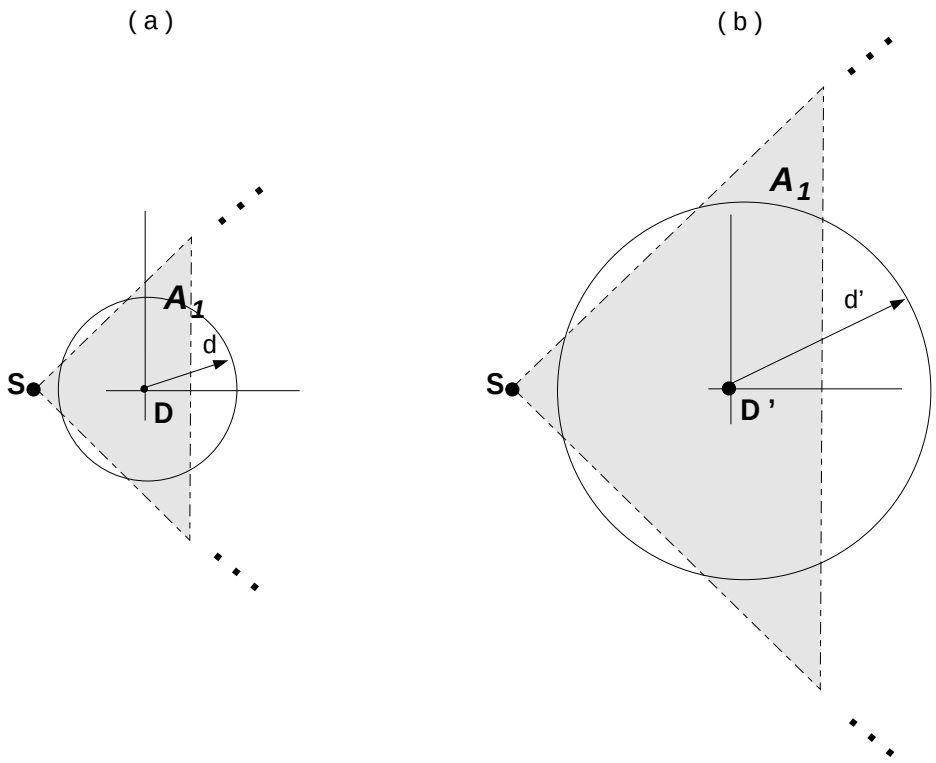

Figure 15: Region $\mathcal{A}_1$ as seen by (a) node D, and (b) node D'.

relative node speed and on the geometry of the area $\mathcal{A}_1$ (which in turn depends on the degree of node S among other factors, but is independent of N or r); $t - t_1$ is the elapsed time since the last LSU was received; r is the distance (at time t_1) in hops between S and D.

The dependence of $p(S, x, D, t, t_1)$ on the ratio $\frac{t-t_1}{r}$ follows from assumption a.8. Consider for example another node D' that is at a distance $2 * r$ from node S, in the same direction as D from S, so that node S also forwards a packet to D' through node I and an optimal next hop decision would be made as long as node D' remains inside the same area $\mathcal{A}_1$. Note however, that node D's view of the area $\mathcal{A}_1$ (regarding itself as the center) will differ from node D''s view. Basically, node D' will view area $\mathcal{A}_1$ (when he is at the center) as being a scaled version of node D's view, where the scale factor is equal to 2. Figure 15 (a) and (b) show node D's and D''s view of the region $\mathcal{A}_1$ respectively.

If the elapsed time (since the last LSU was received from D') is also doubled, the maximum displacement ($maximum\_speed * elapsed\_time$) d' of node D' will also double with respect to node D's maximum displacement d, as also shown in Figure 15. According to assumption a.8 about scaling with time, it would be expected that both probabilities of an optimal and non-optimal next hop decisions (for nodes D and D') be the same. Thus, any simultaneous proportional increment

on $t - t_1$ and r will cancel out. Strictly speaking, let p' and p be the probabilities of an optimal next hop decision for nodes D' and D respectively. Similarly, let $\triangle t'$ and $\triangle t$ be the elapsed time for nodes D' and D respectively. Thus:

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

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

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

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

$$\begin{aligned} &= \int\int_{\mathcal{A}_1} \frac{1}{4(\triangle t)^2} f_{1/0}(\frac{2x}{2\triangle t}, \frac{2y}{2\triangle t})\, 4dx\, dy \\ &= \int\int_{\mathcal{A}_1} \frac{1}{(\triangle t)^2} f_{1/0}(\frac{x}{\triangle t}, \frac{y}{\triangle t})\, dx\, dy \\ &= p \end{aligned}$$

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

Now, replacing the expression for $p(S, x, D, t, t_1)$ in equation 15 we obtain :

$$\begin{aligned} p(S, x, D, t) &= \frac{1}{T(r)} \int_{t-T(r)}^{t} (1 - e^{-\frac{\mathcal{M}(t-t_1)}{r}})dt_1 \\ &= 1 - \left[\frac{r\ e^{-\frac{\mathcal{M}(t-t_1)}{r}}}{\mathcal{M}T(r)} \right]_{t-T(r)}^{t} \\ &= 1 - \frac{r(1 - e^{-\frac{\mathcal{M}T(r)}{r}})}{\mathcal{M}T(r)} \\ &= \frac{1}{2}\frac{\mathcal{M}T(r)}{r} - \frac{1}{6}(\frac{\mathcal{M}T(r)}{r})^2 + \frac{1}{24}(\frac{\mathcal{M}T(r)}{r})^3 + \ldots \end{aligned}$$

where the last expression follows from the series expansion $e^{-x} = 1 - x + \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \ldots$. Thus, a first order approximation for $p(S, x, D, t)$ will be $p(S, x, D, t) \approx \frac{1}{2}\frac{\mathcal{M}T(r)}{r}$. Replacing this expression in equation 14 the following is obtained:

$$S_{sub} = \frac{\lambda_t}{N} \sum_{r=1}^{R} \sum_{D \in N_r(S,t)} |\mathcal{F}_t(S, D)| \frac{1}{2}\frac{\mathcal{M}T(r)}{r} \tag{16}$$

To compute $|\mathcal{F}_t(S, D)|$ consider that if all the nodes to the left of node S in Figure 14 have the same information as node S, the set $\mathcal{F}_t(S, D)$ would be formed by the nodes inside the area

$\mathcal{A}_2$. However, nodes to the left of node S may not have been updated at time t_1 (last time node S was updated), and therefore they may have a different view of the network. Thus, the actual set of nodes that forward packets to D through node S (and therefore I), may be different from the set of nodes inside area $\mathcal{A}_2$. Nevertheless, the *number* of such nodes must be roughly similar. The last statement follows from the observation that if the network in Figure 14 if vertically split (passing by node S), all the nodes on the left half will have to go through some of the nodes on the boundary (node S or some other node along the vertical axis). Let $x_1, x_2, \ldots$ be the nodes on the vertical axis, and let $\mathcal{F}_t(x_1, D), \mathcal{F}_t(x_2, D), \ldots$ be the sets formed by the flows reaching node D through node $x_1, x_2, \ldots$ respectively. Thus, each node in the left half is the source to exactly one flow belonging to exactly one of the aforementioned sets. Thus, the problem of assigning a node (flow) to one set arises. If all the nodes in the left half of the network in Figure 14 have the same view of the network as node S, the number of nodes assigned to $\mathcal{F}_t(S, D)$ is equal to $\sigma * area(\mathcal{A}_2)$. Since node D has been moving, nodes on the left half will have a different (older) view about node D, biasing the assignment. However, since node D's mobility is random, different possibilities cancel out the bias and at the end the proportionality in the assignments is maintained. Thus, the number of nodes assigned to $\mathcal{F}_t(S, D)$ will still be close to $\sigma * area(\mathcal{A}_2)$.

Approximating the area $\mathcal{A}_2$ to the trapezoid inscribed on the circle of radius R, and computing this area as the difference between the two similar triangles with a vertex at D (see Figure 14) the following expression is derived: $area(\mathcal{A}_2) \approx \frac{\alpha}{2} * r * L * \left[(\frac{R+r}{r})^2 - 1\right]$, where α is the base of the smaller of the similar triangles, and is numerically equal to the distance between node S and one of its *close* neighbors in Figure 14. Thus, α is more or less independent from r and R. L is the average distance between a node and its neighbors, so that nodes that are r hops away will be – in average – at a distance $r * L$. [19] Further simplifying the previous expression and inserting it in equation 16, the following is obtained:

$$S_{sub} = \frac{\lambda_t}{N} \sum_{r=1}^{R} \sum_{D \in N_r(S,t)} \frac{1}{4} \alpha \sigma L \mathcal{M} R^2 (1 + \frac{2r}{R}) \frac{T(r)}{r^2} \quad (17)$$

$$= \frac{\lambda_t}{N} \sum_{r=1}^{R} |N_r(S,t)| \frac{1}{4} \alpha \sigma L \mathcal{M} R^2 (1 + \frac{2r}{R}) \frac{T(r)}{r^2} \quad (18)$$

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

[19] It may appear that α and L are the same quantity, but they differ in that α is the distance to the closest nodes and as such it is defined by the geographical node positioning. L in the other hand, is the average distance to all one hop neighbors, and therefore it depends on the node transmission power. If the transmission power is large, L may be significantly greater than α.

$$S_{sub} = \frac{\lambda_t}{N}\frac{\alpha\beta\sigma L}{4}\mathcal{M}R^2\sum_{r=1}^{R}(1+\frac{2r}{R})\frac{T(r)}{r} \quad (19)$$

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

$$S_{sub} \approx \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma L}{4}\mathcal{M}R^2\sum_{r=1}^{R}\frac{T(r)}{r} \quad (20)$$

$$= \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma L}{4}\mathcal{M}R^2\sum_{i=1}^{n}\sum_{r=s_{i-1}+1}^{s_i}\frac{2^{i-1}}{r}t_e \quad (21)$$

$$= \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma L}{4}\mathcal{M}R^2 t_e\sum_{i=1}^{n}2^{i-1}f_i \quad (22)$$

Where $1 < \gamma \leq 3$, and the last 2 equilities follow since $T(r) = 2^{i-1}\,t_e$ for $s_{i-1}+1 \leq r \leq s_i$ (see Figure 4) [20]. And $f_i = \sum_{r=s_{i-1}+1}^{s_i}\frac{1}{r}$. Finally, approximating f_i by using $f_i \approx \int_{s_{i-1}}^{s_i}\frac{dr}{r} = ln(\frac{s_i}{s_{i-1}})$ as an approximation for $i \geq 2$, and $f_1 \approx ln(s_1)$, and recalling that $s_n = R$, the following expression is obtained [21]:

$$S_{sub} \approx \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma L}{4}\mathcal{M}R^2 t_e[ln(s_1)+\sum_{i=2}^{n}2^{i-1}ln(\frac{s_i}{s_{i-1}})] \quad (23)$$

$$= \frac{\lambda_t}{N}\frac{\alpha\beta\gamma\sigma L}{4}\mathcal{M}R^2 t_e[2^{n-1}ln(R)-\sum_{i=1}^{n-1}2^{i-1}ln(s_i)] \quad (24)$$

A.3 HSLS total overhead

FSLS does not induce reactive overhead, thus taking into account the *proactive* and *sub-optimal routing* overheads, FSLS total overhead is obtained:

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

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

[20] $s_0 = 0$ for consistency.

[21] Note that the approximation for f_1 may be loose since the weight (2^1) associated with this term is smaller compared to the next terms. Thus, the highest accuracy of the approximation should be associated with the highest weight term f_n. Fortunately, as i (and s_i) increases, the approximation becomes tighter.

B Control Overhead induced by a node running A-HSLS

In this appendix the control overhead induced by a node, say X, running the A-HSLS algorithm described in subsection 6.4 and Figures 11 and 12 is derived. This derivation is based on the assumption that the time between link failures experienced by a node is a exponentially distributed random variable with mean value $\frac{1}{\lambda_{lc}}$.

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

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

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

HSLS interval, then the expected number of LSU transmissions over the i-th interval is equal to $(1-p_i)N + p_i(1+2f_x)N$.

Regarding the expected length of the i-th interval, let t_i^{elap} be the time elapsed since t_i^{glo} (the time the i-th global LSU was sent) and the time the next link change is detected. Given the memoryless property of the time between link change (exponential distribution), t_i^{elap} will also be exponentially distributed with mean $\frac{1}{\lambda_{lc}}$. If the i-th interval is of SLS type, the interval length will be equal to t_i^{elap}. If the i-th interval is of type HSLS, the interval duration will be equal to t_i^{elap} plus a fixed time equal to $(R_x - 1)t_e$, since $R_x - 1$ time intervals should elapse before *NumEventInt* reaches the value R_x and the algorithm is reinitiated. For simplicity, the above time is rounded and therefore the interval length is approximated by $t_i^{elap} + R_x t_e$. [22] Then, the expected lenght of the i-th interval is equal to $\frac{1}{\lambda_{lc}} + p_i R_x t_e$, that is, it is equal to the expected elapsed time t_i^{elap} plus the term $R_x t_e$ weighted by the probability that the interval was of type HSLS.

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

$$\begin{aligned} X_{A-HSLS}^{control} &= \frac{E\{\text{No. of LSU transmission per interval}\}}{E\{\text{interval lenght}\}} \\ &= \frac{(1-p_i) + p_i(1+2f_x)}{p_i R_x t_e + \frac{1}{\lambda_{lc}}} N \\ &= \frac{1 + 2p_i f_x}{p_i R_x t_e + \frac{1}{\lambda_{lc}}} N \end{aligned} \tag{25}$$

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

$$\begin{aligned} p_i &= P\{t_i^{elap} \leq \frac{R_x t_e}{2}\} \\ &= \int_0^{\frac{R_x t_e}{2}} \lambda_{lc}\, e^{-\lambda_{lc} t}\, dt \\ &= 1 - e^{-\frac{\lambda_{lc} R_x t_e}{2}} \end{aligned} \tag{26}$$

Thus, by applying equation 26 in equation 25, the control overhead induced by a node X, running the A-HSLS algorithm ($X_{A-HSLS}^{control}$) is obtained.

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

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", *IN RIA Research Report 3965, Institut National de Recherche en Informatique et en Automat ique (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