On the performance of the HSLS Routing Protocol for Mobile Ad hoc Networks

G. Koltsidas, G. Dimitriadis and F.-N. Pavlidou
Department of Electrical and Computer Engineering, Telecommunications Division, Aristotle University of Thessaloniki, Thessaloniki, Greece
Email: fractgkb@auth.gr

Abstract. The area of mobile ad hoc networks has recently attracted much scientific interest, as a very appealing research area with many open issues and still unsolved problems. One of the main issues that concerns researchers is the development of routing algorithms that present good performance and face a hostile environment. Many routing protocols have been proposed, attempting to minimize routing overhead, or to reduce the energy consumed by nodes in order to maximize their lifetime. A critical issue, though, is the development of routing protocols that have the ability to maintain their good characteristics at an acceptable level as the network population grows, an ability known as scalability. FSR, ZRP, HierLS and FSLS protocol family are only a sample of scalable algorithms that have been proposed so far. The HSLS protocol is a member of the FSLS family that is proved to scale the best among the algorithms of the FSLS protocol family. In this paper we propose a mechanism to enhance the already good characteristics of the HSLS protocol aiming at the reduction of routing overhead of the original protocol. This new scheme, which we called AFHSLS, exploits the so-called border nodes, in order to deliver routing packets to their destinations. The new algorithm is proved through simulations to significantly reduce routing overhead, with minor or practically no effect on other metrics, such as packet delivery ratio and delay of data packets.

Keywords: ad hoc, routing, scalability, FSLS, HSLS, AFHSLS.

1. Introduction and Related Work

Routing protocols are divided into two general categories: proactive and on-demand. When a proactive algorithm is implemented, each node in the ad hoc network maintains routing information about any other node in the same network. Link state information about a node is retained even if no data packet has to be sent to that node. If the node, at a time instant, receives a packet that has to be forwarded to another node, the path towards which destination is already known, so the data packet suffers very little delay. On the other hand, on-demand routing protocols initiate a route request mechanism only when there is a need, i.e. a data packet is received that has to be relayed to an unknown destination. Hence, routing overhead of on-demand algorithms depends mainly on traffic, while for proactive algorithms routing overhead depends mainly on the size of the network. This is the reason why on-demand protocols are assumed to be more scalable than proactive ones.

A lot of routing protocols have been proposed for ad hoc networks (proactive, on-demand or hybrid), that can scale well as network size grows.

Fisheye State Routing (FSR), presented in [6], is a proactive algorithm based on the idea that the further the source-destination pair is located, the less accurate the routing information has to be. Under FSR, a node sends updated link state information to its neighbours in a periodic way, in order to avoid spreading the information through the whole network and thus increasing routing overhead and reducing network's performance. Moreover, link state information is updated with different frequencies according to the distance from the source node. Hence, as a packet is forwarded, the route to the destination becomes more and more accurate.

ZRP [3] is a combination of proactive and reactive techniques. Every node maintains a zone area, inside which routing information is updated through a proactive scheme. If a packet has to be sent to a destination that does not lie inside the node's zone, the node reactively will initiate a Route Request procedure in order for a path to that destination is found. Mechanisms have also been proposed ([4], [5]) in order for a node to dynamically change the radius of the zone, according to mobility and traffic.

HierLS is an algorithm that is based on clustering. Clusters are established at multiple layers. In the beginning, nodes are organized into clusters, thus forming the 1^{st} layer of clusters. Then, those clusters are considered as nodes in order to form the 2^{nd} level of clusters, and the procedure goes on until the highest layer of clustering is reached. Link state packet transmissions are limited to the nodes of the cluster level. HierLS relies on the Location Management service to inform a source node about the address of the highest level cluster that contains the desired destination and does not contain the source node. Location Management service can be implemented in a proactive, reactive or hybrid way.

The rest of the paper is structured as follows. In Section 2 the FSLS family of protocols is described and particularly the HSLS algorithm, on which our work concentrates. The proposed mechanism for routing overhead reduction is described in Section 3. Simulation parameters and results are presented and analyzed in the following Section. Section 5 concludes the paper.

2. FSLS Protocol Description

2.1. FSLS Protocol Family

The HSLS (Hazy Sighted Link State) protocol is a member of the FSLS (Fuzzy Sighted Link State) protocol family. Therefore, we will first describe the philosophy and the performance of FSLS protocols.

The fundamental routing packet of the FSLS protocols is the Link State Update (LSU). This packet contains the *link state* of the node that generated it, i.e. the list of neighbouring nodes and the status of the link with each neighbour. Every routing packet, thus every LSU, has a Time To Live (TTL) field. The value of the TTL field is initialized by the node that generates the packet (the source node) and each time the packet is received by a node, the TTL value is reduced by one. Hence, LSU's TTL value in fact represents the number of times that the packet can be relayed, in other words, how deep into the network the packet will travel, with regard to the source node. If the TTL value is equal to or greater than the diameter of the network, the packet will practically reach all the nodes in that network. In other words, when a node wishes a specific packet to be received by all nodes, it simply sets the packet's TTL field to a value greater than the diameter of the network, usually a value considered as infinite. Such a packet is called a global LSU, thus an LSU that will reach every node into the network.

FSLS protocols are predicated upon the idea that the greater the distance between two nodes, the less accurate the route between them needs to be. The philosophy is very similar to that of the FSR protocol. The accuracy of the path to a destination node that lies far from the node under consideration is unlikely to have a serious effect on the next hop decision the node will make in order to forward a packet to the destination node. Therefore, nodes that are far away from the source node can be informed less frequently about link state changes than the nodes closer to it. The innovation that the FSLS algorithms introduce is the mechanism of LSU transmissions.

Under FSLS, every node in the network maintains an integer Counter, which is initialized when the node is powered on or after a global LSU. The Counter is increased by 1 every $T = t_e$ seconds, where T is the period of the protocol and is the same for all nodes in the network. Each time the Counter is increased, the protocol seeks for the greatest power of 2 that divides the Counter, i.e. division leaves no residual. So, the Counter's value can be expressed as $Counter = N \cdot 2^{i-1}$, where N is the quotient of the aforementioned division and $p = i - 1$ is the exponent that the algorithm searches for. Afterwards, the algorithm has to decide upon two other issues: whether an LSU should be transmitted and what its TTL value should be. In order to decide whether the node should transmit an LSU, the protocol looks into the recent past of the node. More specifically, the node will transmit an LSU if there has been a link status change in the node's neighbourhood in the past $2^{i-1}t_e$ seconds, or, equivalently, in the past 2^{i-1} periods of the protocol. If an LSU is transmitted, the algorithm sets its TTL value to s_i. The progression of s_i is monotonically increasing, that is $s_i \geq s_j$ for $i \geq j$. Summarizing, each node "wakes up" every t_e seconds, increases its Counter by 1 and transmits an LSU with TTL equal to s_i if there has been a link status change in the past $2^{i-1}t_e$ seconds, or during the last 2^{i-1} periods. However, this procedure does not go on for ever. When the TTL value exceeds the maximum distance into the network that the node running the algorithm is aware of, the TTL is set to infinity and a global LSU is transmitted, if there is a need to do so. After a global LSU, the node reinitializes the Counter and all clocks, and the procedure starts from the beginning. The above mechanism guarantees that nodes which are s_i hops away from the node under consideration will receive an updated LSU after $2^{i-1}t_e$ seconds at most, so they will be informed about a link status change after 2^{i-1} periods of the protocol at most.

Let us explain the protocol's behaviour using the following example. For the sake of simplicity, we will assume a high mobility scenario, so that a link status change takes place during every time interval t_e, and we will focus on a particular node. Initially, Counter is set to zero and so is the clock. After t_e seconds ($t = t_e$), the node "wakes up" and increases the Counter by one, so $Counter = 1$. The greatest exponent p that divides Counter is 0, so $i = 1$ and $Counter = 1 \cdot 2^0$. Therefore, assuming that a link status change always takes place, an LSU will be transmitted with $TTL = s_2$. At time $t = 2t_e$, the node "wakes up" and increases the Counter again, so $Counter = 2$. This time the exponent $p = 1$, so $i = 2$ and an LSU is transmitted with $TTL = s_2$. An interesting phenomenon takes place at the time instant $t = 3t_e$. After the Counter is increased ($Counter = 3$), the algorithm calculates the exponent $p = 0$. Hence, $i = 1$ and an LSU is transmitted with $TTL = s_1$. We mention here that the previous LSU's TTL was $s_2 \geq s_1$. Finally, at $t = 4t_e$, $Counter = 4$, $p = 2$, $i = 3$ and an LSU is transmitted with $TTL = s_3$. The procedure is presented more clearly in Figure 1a, where the horizontal axis presents time (in multiples of t_e) and the vertical axis represents the TTL values of the transmitted LSU's (s_i values presented are indicative). In the previous example, we have assumed that $s_5 = \infty$, i.e. a global LSU is transmitted at $t = 16t_e$ and the scheme is repeated in time.

In order to cover the low mobility scenario cases as well, a soft state protection has been adopted, thus a global LSU is transmitted every t_b

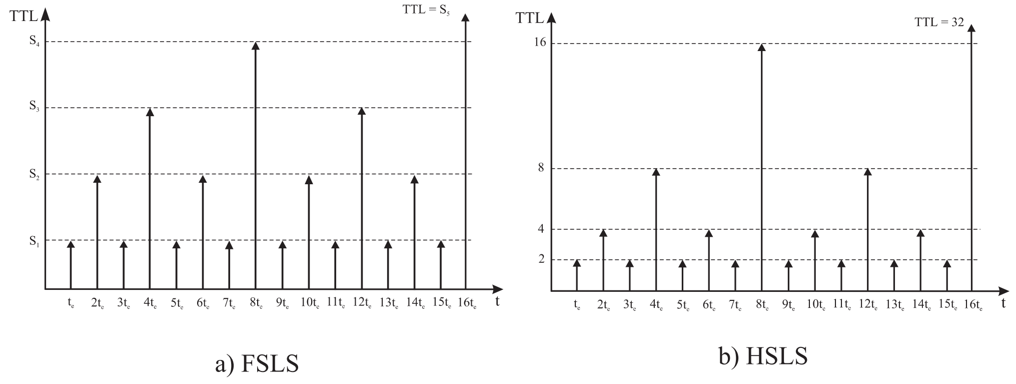


Figure 1. LSU's TTL values and transmission instants under a high mobility scenario (FSLS and HSLS protocols).

seconds when no LSU has been transmitted during this time period. At the extreme case of completely stationary nodes, every node transmits a global LSU every t_b seconds. The value of t_b is usually selected to be much greater than t_e.

2.2. The HSLS Protocol

In this section we will describe the HSLS protocol. Santivanez and Ramanathan in [1] provided approximate expressions of the proactive, suboptimal and total overhead of the FSLS protocols. According to that work, the proactive overhead of a protocol is defined as the amount of bandwidth consumed by the protocol in order to propagate route information before this information is needed. In addition, the sub-optimal overhead is defined as the difference between the bandwidth consumed by all sources when transmitting data using the routes that the protocol provides and the bandwidth that would be consumed if the optimal routes were used to propagate the data. For any proactive protocol, such as FSLS protocols, the total overhead is the sum of the proactive and the sub-optimal overhead. In their effort to minimize the total overhead of the FSLS protocols with respect to the s_i values, Santivanez and Ramanathan were led to the HSLS (Hazy Sighted Link State) protocol, which is an integer solution of the previous mentioned minimization problem. Thus, HSLS is the algorithm with the minimum total overhead among all the FSLS protocols. Under HSLS, the s_i values follow a geometric progression:

$s_1 = 2^1 = 2$
$s_2 = 2^2 = 4$

$s_3 = 2^3 = 8$
$s_4 = 2^4 = 16$
$s_5 = 2^5 = 32$

Therefore, the description of the HSLS protocol could be summarized in the following: *Every $2^{i-1}t_e$ seconds (i=1,2,3,4,...) the node "wakes up" and transmits an LSU with $TTL = 2^i$ if a link status change has been detected in the past $2^{i-1}t_e$ seconds.*

Figure 1b presents the previous example that we used when we explained the FSLS, for the case of the HSLS protocol. The difference lies in the TTL values on the vertical axis.

3. AFHSLS Protocol Description

In the previous section we described the HSLS protocol. Before we describe our proposal, a few definitions should be given. We call *source node* the reference node or the node under consideration, when we examine a protocol's behaviour. We will call *k-hop border nodes* of a source node the set of nodes that are exactly k hops away from it. We can easily come to the conclusion that 1-hop border nodes are in fact the source node's neighbours. As an example, let us consider the simple network presented in Figure 2. Assuming that the source node is node A, 1-hop border nodes are nodes B, C, D and E, 2-hop border nodes are nodes F, G, H, I, J, K, L, M and N and node P is the only 3-hop border node. The description of the proposed mechanism follows.

If we examine the HSLS protocol carefully we will notice that in a number of cases a particular node receives identical LSUs multiple times, even though this is not necessary. This happens, for instance, when the source node transmits an LSU with $TTL = 2^i$ at time $t_1 = m \cdot t_e$ seconds (m: integer) after a global LSU and for the next $t' = i \cdot t_e$ seconds no link status change takes place, so the source node, at time $t_2 = t_1 + t' = (m + i) \cdot t_e$, transmits an LSU with $TTL = 2^j = 2^{i+1}$, which contains exactly the same information with the previous one. However, the first LSU will reach $2^i - hop$ border nodes and the second one $2^{i+1} - hop$ border nodes. Thus, nodes that are less than 2^i hops away will receive two identical LSU's, which indicates useless bandwidth consumption. Alternatively, the $2^i - hop$ border nodes could broadcast the last received LSU instead of the source node, if they were assured that the source node would not transmit any new LSU. Summarizing, the new mechanism could be stated as follows:

Every $2^{i-1}t_e$ seconds (i=1,2,3,4...) the node "wakes up" and transmits an LSU with $TTL = 2^i$, if there has been a link status change in the

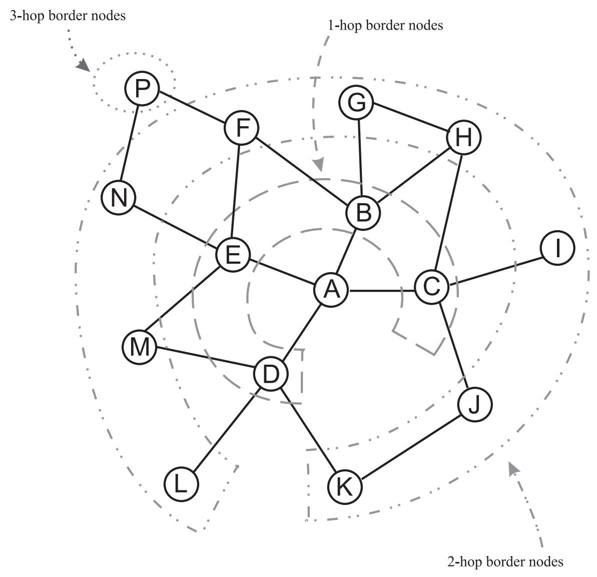


Figure 2. k-hop border nodes.

last t_e seconds. In any other case, the $2^{i'} - hop$ border nodes broadcast the last received LSU, transmitted by the particular node, with TTL value equal to $2^i - 2^{i'}$. i' is the value of i when the last received LSU was transmitted by the source node.

This new version of the HSLS protocol was named AFHSLS (Active Frontier Hazy Sighted Link State). Let us describe the performance of the AFHSLS algorithm by using an example. We assume an ad hoc network that utilises HSLS as the routing protocol. We also assume that a global LSU has just been transmitted by the source node, so timer is set to zero, and that link status changes take place during the 1^{st}, 5^{th}, 7^{th} and 10^{th} time interval t_e for a $16t_e$ period of seconds (low mobility scenario).

At $t = t_e$, an LSU is transmitted with $TTL = 2$, because in the previous t_e seconds a link status change has been detected. Then, the node wakes up at $t = 2t_e$ and transmits an LSU with $TTL = 4$ because in the previous $2t_e$ seconds a link status change took place (the same as before). The node also "wakes up" at $t = 3t_e$, but no LSU is transmitted, as in the previous t_e seconds there has been no link status change in the node's neighbourhood. At time $t = 4t_e$, the node transmits an LSU with $TTL = 8$, because in the previous $4t_e$ seconds a link status change took place. Similarly, the node wakes up at $t = 5t_e, 6t_e, 7t_e, 8t_e, 10t_e$ and $12t_e$ seconds in order to transmit an LSU with $TTL = 2, 3, 2, 8, 4, 8$ and 16 respectively. Figure 3a shows this procedure. The horizontal axis represents time in multiples of t_e. The TTL values of the transmitted LSUs, or, equivalently, the k-hop border nodes that the LSUs will reach, are placed on the vertical axis. This figure will help us compare HSLS with AFHSLS and understand the differences between them and the degree of efficiency in bandwidth utilization we can achieve by adopting AFHSLS.

According to the aforementioned description of AFHSLS, in the previous example the same node would behave as follows: The node "wakes up" at $t = t_e$ and transmits an LSU with $TTL = 2$, as it would act under HSLS. However, at $t = 2t_e$ the source node will *not* transmit any LSU, because in the past t_e seconds no link status change has been detected (the previous link status change took place in the first time interval). On behalf of the source node, all 2-hop border nodes will transmit the last LSU received by the source node with TTL value equal to 2, in order to 4-hop border nodes of the source node will receive the LSU. We should mention here that the information which 4-hop border nodes receive is the same irrespective of the routing protocol we use (HSLS or AFHSLS). However, in the case of AFHSLS, nodes that are 2 or less hops away from the source node will not receive 2 LSUs containing exactly the same information, thus bandwidth is used more efficiently. At $t = 3t_e$, the source node will not transmit an LSU, because no link status change has been detected in the previous t_e seconds. At time $4t_e$, the source node transmits no LSU again, because no link status change has been detected in the past t_e seconds. However, all 4-hop border nodes will transmit the last received LSU by the source node (actually the last received LSU that a 2-hop border node initialized on behalf of the source node) with $TTL = 4$, so that all nodes that are located 8 or less hops away from the source node will be informed. In this case, all transmissions by the nodes that are 4 or less hops far from the source node are avoided. The entire procedure from $t = 0$ to $t = 16t_e$ for the case of AFHSLS is presented in Figure 3b. The arrows now are drawn from the k-hop border nodes that initiate the LSU transmissions on behalf of the source node. Comparing Figures 3a and 3b we can easily notice the significant efficiency that the AFHSLS provides relative to the original HSLS protocol.

In order for the k-hop border nodes to be able to transmit LSUs on behalf of the source node, they should save in a memory all LSUs which were received and not immediately retransmitted (which means that the node was a border node for the source node of the specific LSU). However, this is not compulsory. As the received LSU induces changes on the routing table of the node, the node could reconstruct it, looking into its own routing table, if that LSU should be retransmitted.

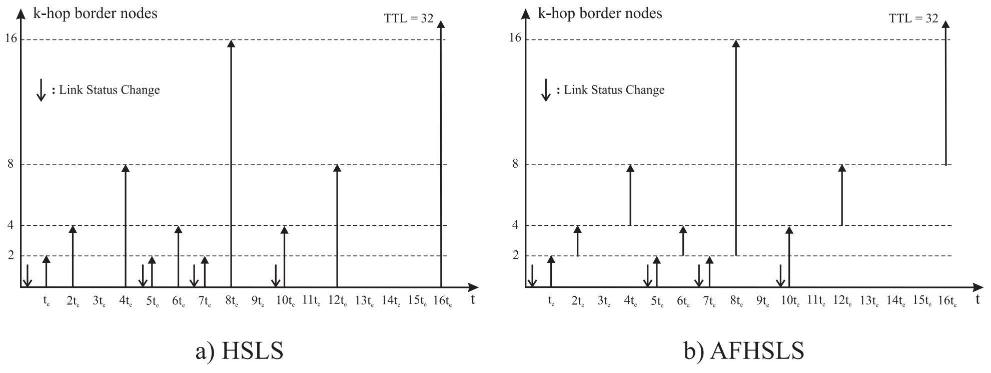


Figure 3. LSU transmissions and TTL values for a low mobility example when the routing protocol is a) HSLS and b) AFHSLS.

So, AFHSLS does not imply that additional memory should be used in order for the nodes to save the received LSU's.

We mentioned earlier that, when the source node detects no link status change for a period of time, $2^i - hop$ border nodes ($i = 1, 2, 3, 4$) will transmit the LSUs, but they should be assured that the source node will not transmit any new LSU. It would not be wise for the source node to send a small packet to the appropriate border nodes to inform them that it will not transmit, because this would lessen AFHSLS's efficiency. However, border nodes can be assured just by delaying the transmissions of the new LSUs for a time interval approximately equal to the time that an LSU would need to travel from the source node to the border nodes. This mechanism will bring about a slight delay in the delivery of the LSUs, but we expect that this delay will barely influence data packet delivery, because the philosophy of HSLS (and AFHSLS) is that nodes that are far from each other do not need to know each other's exact location, but only the direction in which they should forward the message, in order to make a correct next hop decision. We discuss that issue in the next section where we refer to the simulation results.

Node mobility, however, has an impact on our scheme that could not be overlooked. Under AFHSLS, k-hop border nodes are responsible for the delivery of LSU, in case the source node experiences no link status changes for some time. However, k-hop border nodes are moving. This means that, when they should transmit the last received LSU on behalf of the source node, they may not be located k-hops away from the source node. They may either get closer to the source node, or move away from it. In the first case, the LSU transmission from these nodes do not cause any problems. In the other case, though, the nodes are moving in the direction of the LSU propagation. So, these nodes may mistakenly assume that the source node did not transmit any LSU, because the time interval they wait for an LSU transmission by the source node is not enough in order for the LSU from the source node can reach them before they transmit the last received LSU. This may result in the transmission of both LSU (the previous one and the new LSU transmitted by the source node), thus consuming bandwidth unnecessarily. This phenomenon takes place usually far away from the source node, because LSU receiving times cannot be exactly calculated since they have relatively high jitter with respect to the receiving time from nodes near the source node. However, we can use an upper bound in the time the k-hop border nodes will wait before retransmitting the last received LSU, and based on the fact that LSU transmissions with high TTL are not so often and nodes movement makes these phenomena short-living, they do not have a great impact on AFHSLS's performance.

Due to the nodes' mobility, AFHSLS cannot guarantee the reception of an LSU from all the nodes that should receive that LSU. Let us provide an example. Consider a source node transmitting an LSU with TTL value equal to 4. This LSU will reach all 4-hop border nodes. Now consider a specific node which is located 5-hops away from the source node when 4-hop border nodes receive the LSU. Also assume that this node moves towards the source node relatively fast. If no link status change is detected by the source node for some time, 4-hop border nodes will transmit the previously received LSU up to 8-hop border nodes. If the fast moving node at that time is 2 hops away from any 4-hop border node, it will not receive the LSU. This phenomenon is common to nodes moving fast in the opposite direction of the LSU propagation direction. AFHSLS cannot guarantee LSU receptions in situations similar to the above one. This is another drawback of the proposed mechanism, however, node mobility and LSU retransmissions make these phenomena last for short time. These cases are the reason why AFHSLS seams to achieve smaller packet delivery ratios than HSLS, as described in the next section.

By the description of AFHSLS it is obvious that it performs identical to HSLS under very high mobility, because link status changes take place during every time interval t_e. Notwithstanding, AFHSLS outperforms HSLS under middle and low mobility scenarios. The authors in [1] also proposed a mechanism for routing overhead reduction under low mobility scenarios. They named the new protocol Adaptive-HSLS (A-HSLS). A-HSLS switches between HSLS and SLS mode according to the experienced link status change ratio. More specifically, after a global LSU, the node is in the Undecided mode. If t_{th} seconds elapse and no link status change takes place, the node switches to SLS mode, assuming that link status changes rarely happen. When a link status change takes place, the node will transmit a global LSU and will switch back to the Undecided mode. If a link status change takes place before t_{th} seconds elapse, the node switches to HSLS mode and the normal procedure goes on, as the node assumes high mobility conditions. We remind here that under SLS, a node transmits a global LSU whenever a link status change takes place. The previously mentioned threshold t_{th} was calculated by authors to be $t = \frac{R_x t_e}{2}$, where $R_x t_e$ is the instant when a global LSU is to be transmitted. Below this threshold, HSLS's overhead exceeds SLS's overhead, so it would be better if the node followed the SLS algorithm. In our paper, though, we considered the original HSLS protocol, because in our simulated scenarios A-HSLS showed very small differences relative to HSLS. One could expect that, under very low mobility scenarios, A-HSLS and AFHSLS show similar behaviour, regarding routing overhead.

4. Performance Evaluation

In order to evaluate the proposed mechanism we conducted a number of simulations using the NS-2 simulation tool. Simulation parameters are summarized in Table I. Two types of networks were simulated: a "small" network comprising 50 nodes moving inside a 1200x400m^2 area and a "large" network of 100 nodes moving inside a 1500x640m^2 area. The areas' dimensions were selected so that the same area corresponds to every node in both scenarios (9600m^2 per node).

For the purpose of neighbour discovery, we used the method of "Hello" packets. Each node broadcasts a small "Hello" packet every 0.5 sec, so that all neighbouring nodes become aware of its existence. If no "Hello" packet is received by a neighbouring node for 3 continuous times, this node is no longer considered as a neighbour. In order to take an average view of the performance of the AFHSLS algorithm, we ran 6 different scenarios for every pause time and then averaged the values of the performance metrics. We computed the most common metrics used when evaluating the performance of routing protocols in ad hoc networks:

– **Packet Delivery Ratio (PDR)**. It is defined as the percentage of the data packets produced by the sources, that were finally delivered correctly to their destinations.

Table I. Simulation Parameters.

Area	1200x400 / 1500x640 m^2
Number of Nodes	50 / 100
Simulation Duration	900 seconds
Mobility Scenario	Random Waypoint Model (RWP)
Pause Times	0, 5, 10, 20, 40, 80, 150, 300, 600, 900 sec
Speeds	Uniform in [0m/s, 20m/s]
Radius	180 m
MAC Speed	2 Mbps
RTC/CTS Mechanism	ON
t_e	6 seconds
t_b	48 seconds
Number of sources	19 / 31
Number of Data Flows	25 / 50
Traffic Type	Constant Bit Rate (CBR)
Data packet Length	512 bytes
Traffic initialization time	100 seconds after simulation started

– **Average Packet Delay**. It is defined as the average time the delivered data packets needed to travel from the source node to the destination node.

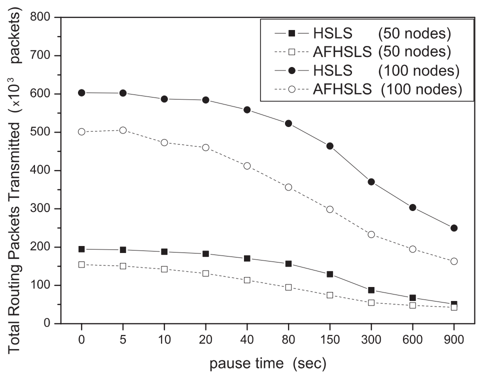
– **Average Number of Hops**. This metric refers to the average number of hops that the delivered data packets needed to reach the destination node.

– **Routing Overhead**. It is defined as the total number of routing packets transmitted by all the nodes during the simulation.

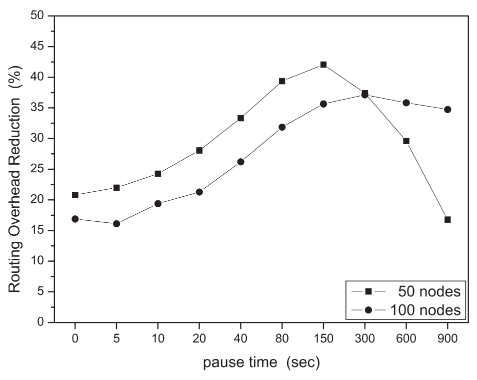
Usually "Hello" packets are ignored because they cannot be avoided by any routing protocol, as they are a fundamental mechanism for neighbour discovery. Therefore, we did not include "Hello" packets in the last metric.

Figure 4a presents the Routing Overhead for both HSLS and AFH-SLS algorithms versus pause time and for both the "small" and "large" network case. First of all, we can observe that routing overhead is a monotonically decreasing function of pause time. This is expected, because as pause time increases, the node's mobility is reduced, so link status changes are less frequent. This leads both algorithms to transmit less LSUs. The interesting fact is that there is a clear difference in routing packet transmissions between HSLS and AFHSLS. For the "small" network, there is a difference up to 60,000 routing packets and for the "large" network the difference reaches 160,000 packets. In order to obtain a more spherical view of the AFHSLS's performance, though, we should take into consideration the total packets transmitted, so we calculated the relative overhead reduction that AFHSLS introduces with respect to HSLS. The results are presented in Figure 4b. We notice that, as pause time increases, overhead reduction also increases, which means that AFHSLS sends less routing messages than HSLS, so it performs better in medium and low mobility scenarios. This is also expected, as under very high mobility scenarios the two protocols perform almost identically. A peak of approximately 42% is reached for the "small" network, while for the "large" network case a 37% reduction in routing packet transmissions is achieved. However, even under high mobility scenarios (pause time = 0 or 5 seconds) AFHSLS consumes about 20% and 16% less bandwidth than the original HSLS for the case of "small" and "large" network respectively. For pause time greater than 300 sec, overhead reduction decreases. This is due to the fact that nodes rarely move, so soft state updates are transmitted and little use of the proposed mechanism is made. Finally, when pause time is 900 seconds, nodes are stationary for the entire duration of the simulation. In this case, routing packet transmissions benefit from the proposed mechanism only during the transitional state at the beginning of the simulation, as a steady state is rapidly reached and only soft updates are then broadcasted. Additionally, the total number of routing packets transmitted is small and proportional to the number of nodes. So, the percentage of the packets the AFHSLS protocol saves is lower for the "small" network than for the "large" network case.

In order to evaluate the performance of the proposed scheme, we should also examine it's impact on the other metrics: Packet Delivery Ratio, Average Number of Hops and Average Packet Delay. Figure 5a presents Packet Delivery Ratio as a function of pause time. At first glance, there is practically no difference between the two protocols. An observable difference can be noticed for pause times 80, 150 and 300 seconds, but this difference never exceeds 5%. However, for the previously mentioned pause times, according to Figure 4b, AFHSLS achieves a reduction greater than 30% in routing overhead. In the case of high mobility of the nodes, PDR of both protocols is almost identical. Since AFHSLS reduces routing overhead, one would expect that AFHSLS would achieve higher packet delivery ratios than HSLS. However, simulation results show that PDR is reduced. This is due to phenomena described in the previous section: There are cases under AFHSLS when both the previous one and the new LSU are transmitted

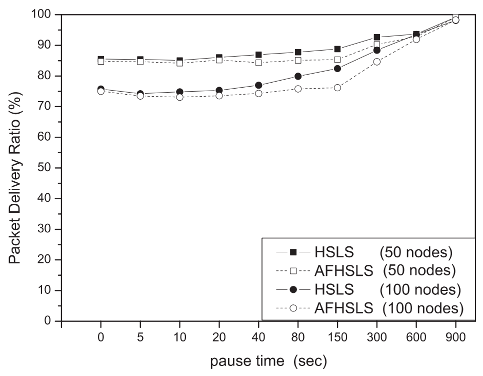


a)

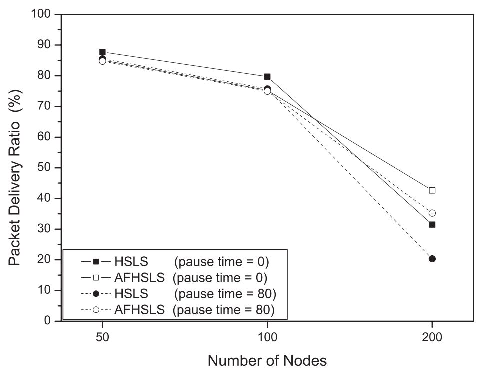


b)

Figure 4. (a) Routing Packets and (b) Overhead Reduction.



a)



b)

Figure 5. Packet Delivery Ratio.

and when particular nodes do not receive an LSU due to their mobility, however HSLS would ensure the LSU delivery to that nodes. This is why we also simulated 2 scenarios with 200 nodes in a $2400x800m^2$ region, pause times equal to 0 and 80 seconds and the same traffic pattern used in the case of the "large" network. Figure 5b shows the Packet Delivery Ratio versus the Number of Nodes in the network for pause times equal to 0 and 80 seconds. One can clearly figure out that when network size increases beyond 200 nodes, AFHSLS benefits from the reduced routing overhead and attains a greater packet delivery ratio than HSLS.

As described in the previous section, AFHSLS delivers exactly the same LSUs as HSLS with a small time delay. This means that, gener-

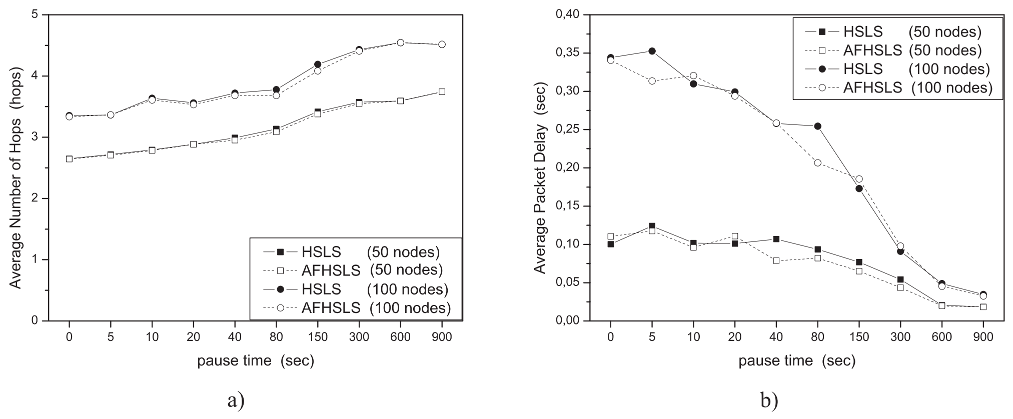


Figure 6. a) Average Number of Hops and b) Average packet delay.

ally speaking, nodes should maintain almost the same routing tables regardless of the protocol adopted. So, focusing on the Average Number of Hops metric, no difference should be observed between the original HSLS protocol and the AFHSLS protocol. In Figure 6a the Average Number of Hops is presented versus pause time. Obviously, our expectations agree with the simulation results.

Finally, Figure 6b shows the Average Packet Delay of data packets over pause time. At first glance, we notice that delay decreases as pause time increases, as we expected. Low mobility translates into less bandwidth consumption, so the common medium is free for the data packets transmissions, thus reaching their destinations faster than they would do if nodes were constantly moving and routing messages should also be transmitted consuming bandwidth and delaying data packets. The most surprising in that figure is that the two curves do not follow a clear order. Generally, AFHSLS tends to deliver data packets in practically the same time or faster than HSLS. As described before, when AFHSLS is adopted as the routing protocol, k-hop border nodes have to wait for some time in order to be assured that the source node did not transmit any LSU. Such a delay may result in a delay in informing other nodes about shorter paths or broken links, so we could expect that AFHSLS should present a higher Average Delay. On the other hand, though, since AFHSLS reduces the number of routing packets transmitted, faster packet forwarding could be expected by the nodes. Simulations show that generally packets are delivered faster by the AFHSLS that by the HSLS protocol.

Simulations also confirm that HSLS is a scalable algorithm and results show that AFHSLS scales better than HSLS.

5. Conclusions

In this paper we presented AFHSLS, a modified version of the HSLS protocol that consumes less bandwidth than HSLS while achieving practically the same throughput and data packet delay as HSLS. This is accomplished by simply enforcing k-hop border nodes to transmit previously received LSU's on behalf of the source node itself. Simulation results show that the proposed algorithm, needs from 16% (for high mobility scenarios) up to 35% (at medium and low mobility scenarios) less bandwidth resources for routing purposes, while practically it does not affect data packet delivery, thus improving HSLS's scalability. It would be of great interest to evaluate AFHSLS performance in larger networks and under mobility profiles other than Random Waypoint Model.

References

1. C. Santivanez, R. Ramanathan, 'Hazy sighted Link State (HSLS) Routing: A scalable Link State Algorithm,' BBN technical memo BBN-TM-1301, BBN Technologies, Cambridge, MA, August 2001 (Rev. March 2003).
2. C. Santivanez, R. Ramanathan, I. Stavrakakis, 'Making Link State routing Scale for Ad Hoc Networks,' in Proc. MobiHOC '01, Long Beach, CA, October 2001.
3. Z.J. Haas and M.R. Pearlman, Prince Samar, 'The Zone Routing Protocol (ZRP) for Ad Hoc Networks,' Internet Draft, draft-ietf-manet-zone-zrp-04.txt, July 2002.
4. M. Pearlman and Z. Haas, 'Determining the Optimal Configuration for the Zone Routing Protocol,' IEEE Journal on Selected Areas in Communications, vol 17, no. 8, pp. 1395-1414, August 1999.
5. T. Thongpook and T. Thumthawatworn, 'Adaptive Zone Routing Technique for Wireless Ad hoc Network', in Proc. ITC-CSCC 2002, Phuket, Thailand, , July 16-19, pp. 1839-1842.
6. G. Pei, M. Gerla, Tsu-Wei Chen, 'Fisheye State Routing in Mobile Ad Hoc Networks,' IEEE ICC 2000, New Orleans, USA, 2000, vol 1, pp. 70-74.
7. X. Hong, K. Xu, M. Gerla, 'Scalable Routing Protocols for mobile ad hoc networks,' IEEE Network magazine, vol 16, no. 4, pp. 11-21, July-August 2002.
8. C. Perkins, 'Ad Hoc Networking, Addison Wesley, 2001.
9. Network Simulator 2 (ver. 2.27), www.isi.edu/nsnam/ns.