Sequence Number Aided Source Routing for Ad-Hoc Networks

Evangelos Papapetrou (epap@eng.auth.gr)
Aristotle University of Thessaloniki

Fotini - Niovi Pavlidou (niovi@eng.auth.gr)
Aristotle University of Thessaloniki

Abstract. Wireless multihop mobile networks, also known as ad hoc networks, are characterized by stochastic topology variations. Random movements of mobile hosts in and out of each other's range encumber smooth system operation and impose limitations on the network performance. Various routing protocols suitable for such networks have been proposed however implementation and performance issues are still considered top research priorities. This paper proposes a new reactive protocol that introduces the use of sequence numbers for evaluating validity of cached routing information when source routing and route caching are used. The new protocol reduces the possibility of using and spreading across the network stale routing information therefore reduces the overhead involved in finding a route. To demonstrate the performance of the proposed protocol we compare it, through a detailed simulation model, with Dynamic Source Routing (DSR) protocol which also uses source routing and route caching. Results prove that the proposed protocol effectively reduces use of stale routing information, improving performance compared to DSR in terms of both delivery ratio and routing overhead.

Keywords: ad hoc networks, routing protocol, on-demand, reactive, source routing, sequence numbers, multihop, route caching

1. Introduction

Technological advances have globally increased the penetration of portable laptops and handheld devices equipped with wireless interfaces. The demand for quick and cost-effective connectivity is growing fast. This latter development has rekindle interest in easily and quickly deployable wireless computer networks, also known as ad hoc networks, which operate without the need of a fixed infrastructure. Since they consist of mobile hosts of a relatively small communication range, ad-hoc networks are prone to link variations because users move randomly in and/or out of each other's range. This kind of behavior produces a stochastically fluctuant network topology, deeply affecting the choice of an appropriate routing technique.

Traditional proactive routing protocols [14],[9],[5],[16],[1] cannot perform efficiently in such an environment [16],[1],[4] since they waste limited system resources to discover routes that probably will not be

used. On-demand routing protocols [16],[1],[4],[11],[15] have been proposed as an effective solution to this problem. In this case, discovery of routes is performed only when there is a request for communication between two network nodes, thus minimizing the incurred bandwidth. Besides this basic characteristic that all on-demand protocols share, several techniques have been proposed so far for discovering routes as well as maintaining them [11],[15],[13],[7],[8]. One of the most attractive approaches is the combined implementation of source routing and caching of discovered routes, since each node may reply to route requests issued by other network nodes by taking advantage of full routing information gathered in a route discovery. This is not the case for most protocols because packets are routed on a hop-by-hop basis and each node is aware of the next hop to a destination instead of the full path. Therefore, even when a node is allowed to reply to route requests, this ability is minimized compared to the case that the full path is known. The incomplete exploitation of routing information available in route discoveries leads to overhead increase as demonstrated in [4] and [19].

The most representative protocol, implementing source routing is Dynamic Source Routing (DSR) protocol [11],[10]. It is based on techniques, used for internetworking IEEE 802 LANs [20]. Although DSR performs efficiently over a significant range of mobility, for high rates of network variation frequent link failures intensify dissemination of stale routing information in network caches. As a result, performance in terms of delivery ratio degrades while routing overhead is aggravated [4], [19]. In an effort to favor use of valid cache content, DSR descendants have been proposed in the literature [2], dealing with the introduction of stability metrics for choosing routes. Although an improvement in terms of delivery ratio is achieved the main disadvantage is the increased incurred overhead, since the introduced metrics follow an additive rule [22], therefore each node is required to propagate all received route request packets.

It is clear that combined use of source routing and caching of routes may be advantageous on condition that spreading and use of stale information is restrained by somehow evaluating the freshness of cached routes without at the same time increasing the required overhead. To this end, sequence numbers emerge as the most appealing and easy to implement solution. Although already used in table driven protocols for eliminating loop formation, sequence numbers have never been used for estimating cached information freshness. In this paper, we will propose a new on-demand routing protocol, called *Sequence number Aided Source Routing* (SASR), that combines the use of source routing, caching of routes and sequence numbers to enhance network perfor-

mance in terms of delivery ratio and routing overhead. SASR relies on sequence numbers to eliminate stale cached routes and successfully deliver packets. Moreover, sequence numbers are further exploited by SASR to minimize the use of source routing, therefore eluding overhead related inefficiencies. The rest of the paper is structured as follows. In Section II the proposed protocol is presented in detail. In Section III a detailed mathematical model is presented for evaluating the routing load incurred by the protocol. Subsequently, in Section IV initially we discuss the simulation framework for evaluating the new protocol. Then we present the results of our simulation study and finally useful conclusions are drawn in Section V.

2. Protocol Description

Similar to almost every routing protocol belonging to the category of on-demand protocols, SASR adopts the organization model comprising of two self-contained mechanisms, namely:

- *route discovery*, and
- *route maintenance*

The first refers to requesting (by flooding a request packet into the network), then building and finally establishing routes by having the destination node reply to the request. Usually the described functionality is referred to as *route request* and *route reply* phases, respectively. Each node n_i maintains a monotonically increasing number, called *request number*, which increases every time a route discovery is performed. This number is carried in route request packets and is used to distinguish successive route requests and prevent propagation of multiple request copies. To this end, node n_i maintains also a table r_{n_i} where it keeps records of all request numbers learned from other network nodes. In the following we will denote the request number of node n_i by $r_{n_i}[n_i]$. Finally, route maintenance holds a key role in dynamic environments as it includes all mechanisms related to evaluation of route validity and availability over time.

SASR utilizes source routing and caching of discovered routes [11],[10] as one of the most appealing routing strategies amongst those proposed so far for ad hoc networks. In this manner mobile nodes are provided with multiple communication alternatives by exploiting all routing information gathered during a route discovery. Enhanced route availability is essential not only for reducing communication requests but also for timely meeting them. The latter property constitutes a fundamental quality of reactive protocols.

However, caching of routes comes with a major disadvantage related to dispersion of stale routing information in high mobility rates, leading to performance degradation. To overcome dissemination and use of invalid routes, SASR introduces the combined use of caching and sequence numbers. In the past, sequence numbers have been extensively used in table driven algorithms both in terrestrial and ad-hoc networks [14],[15],[20], in order not only to eliminate loop formation but also to quantify freshness of routing information collected in different route discoveries. Nevertheless, use of sequence numbers must undergo certain changes to be compatible with cached routes rather than routing tables. SASR makes the observation that it is route replies that announce new topology information to the network. SASR requires each node to include an increasing sequence number on every route reply it sends, and every node that caches information derived from the route reply also caches the sequence number. These sequence numbers can then be used in route discovery to favor fresher information. In the new implementation each node n_i maintains a *sequence number* and a table s_{n_i} where it stores the last known sequence number for every node in the network. Node n_i increases its sequence number, stored in $s_{n_i}[n_i]$, when producing a reply on a route request. Discovered routes are cached together with a sequence number for each node comprising the route. The vector of sequence numbers formed is used for evaluating freshness of not only the cached routes but also of any extracted subroute. In this way SASR can prevent the use of stale routing information in route replies and therefore at the same time can deterred it from suffusing the network and contaminating caches.

In the following subsections, the basic mechanisms comprising SASR, will be described in detail.

2.1. Route discovery

As mentioned previously, SASR is an on-demand protocol. Route discoveries are initiated only when a node needs a route to a destination. So far, use of source routing in the route request phase has been a prerequisite for implementing route caching which is a shortcoming, considering the related overhead. SASR avoids using source routing in the route request phase of a route discovery and limits its use in route reply, since the actual caching of routes is performed at that time. The formation of loops when a node replies with a cached route to a request is avoided by means of sequence numbers contrary to DSR where the same functionality is performed by means of source routing. Thereby, SASR attains significant reduction on the overall induced routing overhead.

Consider the case where a node with address n_s wants to send a data packet to a node with address n_d. Initially, n_s checks its cache for a route to the destination. If such a route doesn't exist, it increases $r_{n_s}[n_s]$ and broadcasts a *route request* packet. This packet has the structure $\langle n_s, n_{prev}, r_{id}, sn_{max}, n_d \rangle$. The field n_{prev} describes the node transmitting the packet in each hop (initially $n_{prev} = n_s$) and is used in the establishment of the reverse path since source routing is not enabled in this phase. In r_{id} the source appends its request number ($r_{id} = r_{n_s}[n_s]$) at the time the request is generated. Finally, sn_{max} is used to record the maximum sequence number known for node n_d in intermediate nodes during the travel of the packet toward the destination (initially $sn_{max} = s_{n_s}[n_d]$, with $s_{n_s}[n_d]$ clearly being the sequence number of node n_d known at node n_s). As it will be explained in the following sn_{max} is used for choosing a suitable cached route and at the same time for eliminating loops.

Each node n_i receiving the request packet first checks numbers n_s, r_{id} to determine whether the received packet is a copy of a request already received. If $r_{id} \leq r_{n_i}[n_s]$ the packet is discarded, otherwise node n_i performs the following actions after updating $r_{n_i}[n_s]$ with value r_{id}:

- if $s_{n_i}[n_d] > sn_{max}$ and a suitable route to n_d is not available the packet field sn_{max} is updated with the value $s_{n_i}[n_d]$.

- node n_i adds in its cache a route entry that contains the data $[n_s, n_{prev}, r_{id}]$, which represents the reversed route on which the packet was received. This route entry is used in the route reply phase for setting-up the discovered route.

- the packet is finally broadcasted.

The described procedure is completed when the request reaches the destination and/or an intermediate node having in its cache a route to destination.

Replying to a request involves sending back to its source a *route reply* packet. This is achieved by forwarding the packet to node n_{prev} and then using values n_s and r_{id} to locate the reversed route entries stored earlier in each intermediate node's cache. These entries are deleted after a period of t_{exp} seconds rather than immediately, to allow multiple replies to reach the originator of the request. Time t_{exp} may be relatively high without affecting the performance of the network in a negative manner.

Contrary to DSR where the discovered route has been constructed in the packet header upon reaching the destination, SASR constructs the route during reply phase. Furthermore, the route is constructed

by recording not only the address of the intermediate nodes but their sequence numbers too. Suppose the reply packet has reached node n_j, coming from node n_i on its way back to node n_s. The structure of the packet can then be represented by $\langle n_d, n_s, r_{id}, cr_{(n_i,n_d)} \rangle$ with $cr_{(n_i,n_d)}$ being the route formed so far:

$$cr_{(n_i,n_d)} = \langle (n_i, sn_{n_i}), \ldots, (n_d, sn_{n_d}) \rangle \tag{1}$$

where $sn_{n_i} \ldots sn_{n_d}$ are the sequence numbers of nodes $n_i \ldots n_d$ respectively. Since the sequence number of the destination can be considered as a token of route creation time, by this methodology we make available this kind of information in any case that a subroute of $cr_{(n_i,n_d)}$ may be used in a reply.

Upon receipt of the request, node n_j updates the numbers $s_{n_j}[n_x], \forall n_x \in cr_{(n_i,n_d)}$ with the values $sn_{n_i} \ldots sn_{n_d}$. Then, it increases its sequence number $s_{n_j}[n_j]$, and adds it to the packet along with its address:

$$cr_{(n_j,n_d)} = (n_j, s_{n_j}[n_j]) \cup cr_{(n_i,n_d)} \tag{2}$$

Finally, n_j adds to its cache the entry $cr_{(n_i,n_d)}$ and forwards the packet using the reverse path entry $[n_s, n_{prev}, r_{id}]$.

It is clear that the structure of Eq. 1 also applies to cached routes. Let us denote by n_r the address of a node replying to a request, whether this is the destination of the request or not and by $cr_{(n_r,n_d)}$ the chosen route. Contrary to DSR, not all nodes having a route in their cache are allowed to reply to a request. As previously described, a route entry contains node addresses as well as their sequence numbers at the time the route was created. Suppose that node n_r receives the first copy of a request originated from node n_s and directed to node n_d. A cached route $cr_{(n_r,n_d)}$ may be used only if:

$$s_{n_r}[n_d] > sn_{max} \tag{3}$$

where sn_{max} carries the maximum sequence number known for the destination n_d. To reveal the rationale of Eq. 3 bear in mind that the route tagged with the value sn_{max} is not available because otherwise there would have been a reply. Thus Eq. 3 is summarized in that there is no point in replying to a request with a route produced earlier than the one tagged with the value sn_{max}. The older routes are obsolete because otherwise the request that produced the value sn_{max} would not have been initiated. In this way SASR manages a passive tracing of stale routes. Furthermore, Eq. 3 provides loop free routes. The latter can be proved by the following rationale. It is clear that:

$$s_{n_k}[n_d] \geq s_{n_r}[n_d], \forall n_k \in cr_{(n_r,n_d)} \tag{4}$$

since the route reply packet that contained in its header the route $cr_{(n_r,n_d)}$ has already traversed all the nodes included in it. Let us suppose that node n_r replies to the request packet using route $cr_{(n_r,n_d)}$ which satisfies Eq. 3. Let us also suppose that a loop is formed because the request packet has already traversed one of the nodes $n_k \in cr_{(n_r,n_d)}$. In this case it should be $sn_{max} \geq s_{n_k}[n_d]$ and based on Eq. 4 $sn_{max} \geq s_{n_r}[n_d]$, which conflicts with Eq. 3.

2.2. Route Maintenance and Packet Forwarding

As previously mentioned SASR uses cached routes of the structure presented in Eq. 1. Since each node receiving a reply packet first increases its sequence number and then appends it to the formed route, it is clear that the vector of sequence numbers $sn_{n_i} \ldots sn_{n_d}$ is unique for each cached route and therefore unique in the cache. SASR takes advantage of this feature to suppress the use of source routing in data packet forwarding and therefore economize on bandwidth requirements. Given a route $cr_{(n_i,n_d)}$ each data packet is required to carry only the sequence number in $cr_{(n_i,n_d)}$ that corresponds to the next hop. Since this number is unique in the cache of the receiving node, the used route can be determined. Another option for SASR concerning data forwarding is to implement source routing and the Flow State Extension proposed in DSR [10] in order to overcome the overhead debit of source routing.

The identification of broken links is made in the same manner as in DSR. When a transmitting host times out (after retransmitting a packet for a number of times), it then sends a *route error* to the source of the data packet using the route stored in the packet header. The error packet contains the addresses of the two nodes constituting the broken link. Upon receipt of the error packet, a host removes from its cache the parts of the routes that contain this broken link and the originator of the data packet additionally initiates a new route request. The maintenance procedure can be implemented even when SASR does not use source routing to forward data packets, as long as the reversed path is accessible. To achieve this, SASR may cache the reverse path entries, used for returning reply packets, as part of the forward route.

Finally, SASR due to its organization model which is directly derived by DSR, is able to take advantage of all optimization procedures proposed for DSR such as salvaging, gratuitous route repair, promiscuous listening, e.t.c. [10],[12].

2.3. Miscellaneous Considerations

Management and maintenance of caches over the network is an important issue due to limited memory resources on one hand and searching time requirements on the other. As will be shown by simulation results, SASR manages to limit cache replies that are responsible for cache size growth. Furthermore, SASR uses DSR like timestamps for erasure of routes that are not used. Another important issue is related to the maintenance of request and sequence numbers in cases that a mobile node crashes and loses records. Since the very essence of their implementation in SASR is not altered, proposals made in the literature [15],[10] may be used. A disadvantage of SASR is that in its current form operates over bidirectional links like a plethora of other protocols. However, most of the proposed link layer protocols for ad hoc networks [21],[3] support only the transmission over bidirectional links, which limits the impact of this disadvantage.

3. Routing Load Analysis

One of the most important parameters of a protocol performance is the incurred routing load since available bandwidth is considered a limiting factor. However, for ad-hoc networks such analysis is almost intractable and is rarely presented in literature [18]-[17]. In this section we will calculate limits on the routing load related to both route request and route set-up phases for SASR and DSR. In order for the analysis to be feasible, it involves only the case that replies from cache are not used. The latter analysis will be presented based on simulation results in the following section.

3.1. Route Discovery Analysis

In each route discovery performed in a non-partitioned network the request packet will be transmitted in the network exactly $N-1$ times. In the case of SASR the packet size (s_{rreq}) is independent of the hops this packet has traversed. Therefore the total routing load involved in a route request for the case of SASR is given by:

$$W_{RD_{(SASR)}} = (N-1)s_{rreq} \tag{5}$$

where s_{rreq} includes next hop address and request and sequence numbers. Even though the address size depends on the maximum number of hosts, usually a fixed value is chosen for implementation reasons.

Therefore s_{rreq} is clearly independent of N. As a result, Eq. 5 proves that:

$$W_{RD_{(SASR)}} = \Theta(N) \tag{6}$$

On the other hand in the case of DSR, since the packet size depends on the number of traversed hops, the corresponding routing load for a route discovery can be calculated by the equation:

$$W_{RD_{(DSR)}} = \sum_{i=1}^{k} [l_i s^i_{rreq}] \tag{7}$$

where l_i is the number of nodes that received the request packet after i hops, k is the maximum number of hops in the network, and s^i_{rreq} the size of a route request packet that has traveled i hops. It is clear that the vector $l = \langle l_1, l_2, \ldots l_k \rangle$ does not represent the actual locations of nodes. As an example, we can describe the situation in which a node n_x, residing two hops away from the source node n_s, may receive a request packet that has already traveled more than two hops. This may happen because the node located between nodes n_s and n_x may be congested. It is easily understood that $N - 1 = \sum_{i=1}^{k} l_i$. Based on Eq. 7 we can write the routing load as:

$$W_{RD_{(DSR)}} = \sum_{i=1}^{k} [l_i (i \cdot s_{aadr} + s_{aux_{(DSR)}})] \tag{8}$$

where s_{addr} is the size used for addresses and $s_{aux_{(DSR)}}$ the size used for carrying other information independent of the number of hops. By underestimating the sum in the right side of the previous equation, we can derive that:

$$W_{RD_{(DSR)}} \geq (s_{addr} + s_{aux_{(DSR)}}) \sum_{i=1}^{k} l_i \tag{9}$$

and therefore:

$$W_{RD_{(DSR)}} = \Omega(W_{RD_{SASR}}) = \Omega(N) \tag{10}$$

that is the routing load involved in a route request of DSR grows with a rate greater than this of SASR, as the network size increases.

3.2. Route Reply Analysis

Let us consider the case that the destination node replying to a request is k hops away from the originator of that request. The load incurred

in the case of SASR is given by:

$$W_{RR_{(SASR)}} = \sum_{i=0}^{k} s_{rrep}^{i} \tag{11}$$

where s_{rrep}^{i} is the size of the route reply message at i hops from the destination, in the direction toward the originator of the request. Equation 11 can be written as:

$$\begin{aligned} W_{RR_{(SASR)}} &= \sum_{i=0}^{k} [i(s_{addr} + s_{seq}) + s_{aux_{(SASR)}}] \\ &= (s_{addr} + s_{seq}) \frac{k^2 - k}{2} + k \cdot s_{aux_{(SASR)}} \end{aligned} \tag{12}$$

where s_{seq} the size of sequence numbers and $s_{aux_{(SASR)}}$ is related to information carried in the reply packet (e.g. request number) which is independent of hops. On the other hand the overhead incurred in the case of DSR is given by:

$$W_{RR_{(DSR)}} = k \cdot s_{rrep}^{k} = k^2 \cdot s_{addr} + k \cdot s_{aux_{(DSR)}} \tag{13}$$

By Eq.12 and 13 it is clear that:

$$\lim_{k \to \infty} \frac{W_{RR_{(SASR)}}}{W_{RR_{(DSR)}}} = \frac{s_{addr} + s_{seq}}{s_{addr}} > 0 \tag{14}$$

therefore:

$$W_{RR_{(DSR)}} = \Theta(W_{RR_{(SASR)}}) \tag{15}$$

The previous equation proves that although SASR carries more information in route reply packets (addresses and sequence numbers), its bandwidth requirements grow with the same rate as that of DSR.

4. Simulation Framework and Results

4.1. Simulation Model

This section is devoted in analyzing the performance of the proposed protocol. Since SASR is based on source routing and caching, we will compare its performance to this of DSR which is the most representative protocol employing the same routing strategy. The simulation results were obtained by a custom simulation tool developed in Java by the authors. Various simulations scenarios were studied by varying

parameters related to geometry, data transmission and mobility of network nodes. The simulation time for each scenario was 900 secs and collection of results was based on ten executions for each scenario. This number was sufficient for the convergences of the results. The models used for simulating each aspect of the network are common in the literature [4],[19],[2],[12]:

1. *Geometry Model*: For the network geometry we used two different cases. The first one concerns 50 nodes moving randomly in an area of 700x700 m^2. In the second one, 100 nodes move randomly in an area 1000x1000 m^2. These two configurations provide us with the possibility of comparing the performance of SASR in networks with different diameters. The nominal communication range of each host was considered equal to 250 m.

2. *Data Transmission Model*: We simulated up to 40 connections, each one emerging from different hosts. The well-known Poisson distribution was used for the generation of data packets with rate 1 packet/sec for each connection. The channel capacity was set to 2 Mbits/s, while the packet size was 512 bytes. The time that a packet is allowed to wait for a route in a buffer was set to 500 msec. The timeout for repeating a route request was set to 1 sec and for non-propagating searches to 30 msec.

3. Mobility Pattern: The movement of hosts is derived according to the Random Waypoint algorithm [4]. Each host chooses its destination location and moves toward that destination with a velocity v_{n_i} . When the destination is reached, the host pauses for a time interval and then chooses another location. The speed of each node is uniformly chosen in the range of 0 to u_{max}. The simulations were run for values of u_{max} ranging from 0 to 20 m/sec while the pause time of a host was set to 0 secs which corresponds to maximum mobility.

4.2. Evaluation Metrics

Besides the qualitative desired characteristics of routing protocols, quantitative metrics are also of interest for evaluating their performance. For the evaluation of SASR we use three different metrics:

– end-to-end packet delivery ratio i.e. the ratio of data packets delivered to the data packets generated,

– normalized routing overhead i.e. the number of routing packets transmitted per data packet delivered and

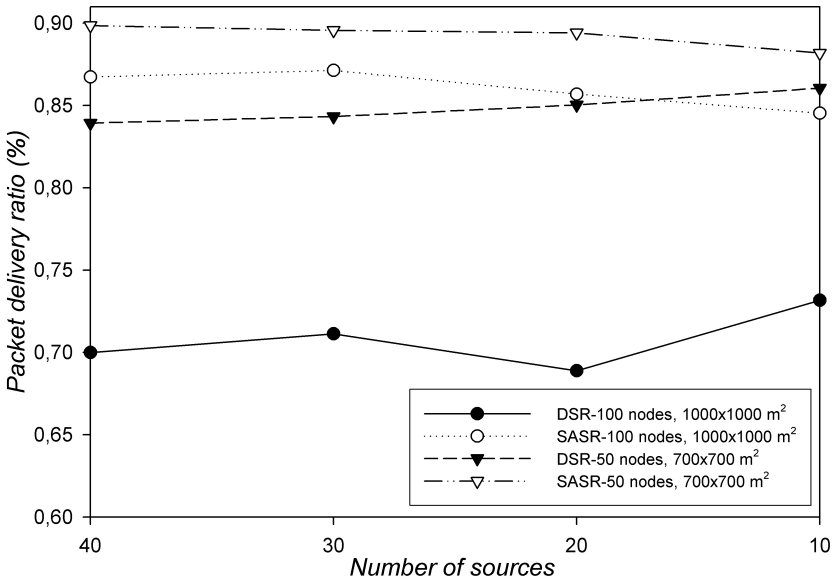

Figure 1. Packet delivery ratio with respect to host speed for the 50 and 100 node models.

– average delay i.e. the time between generation of a packet and successful delivery to its destination.

While the first and the third are considered the basic metrics of the protocol effectiveness, the second evaluates the efficient use of system resources [6]. However, additional metrics for assessing the internal protocol operation are provided in this study. Two different experiments were conducted to measure the performances of the two protocols. The first one assesses the performance in different mobility levels and the second one when different numbers of data sources are used.

4.3. Varying Mobility

The first objective of the study was to compare the performance of SASR under various levels of mobility. In Figure 1, the packet delivery ratio is depicted for various node speeds and for both adopted geometries. It is clear that SASR outperforms DSR for all scenarios and through the entire mobility range. As expected, the performance of both algorithms deteriorates as mobility increases. It must be noted though, that the difference in the performance of the two protocols increases with mobility. This behavior is a clear indication of the efficient cache management that SASR accomplishes. Obsolete routes are

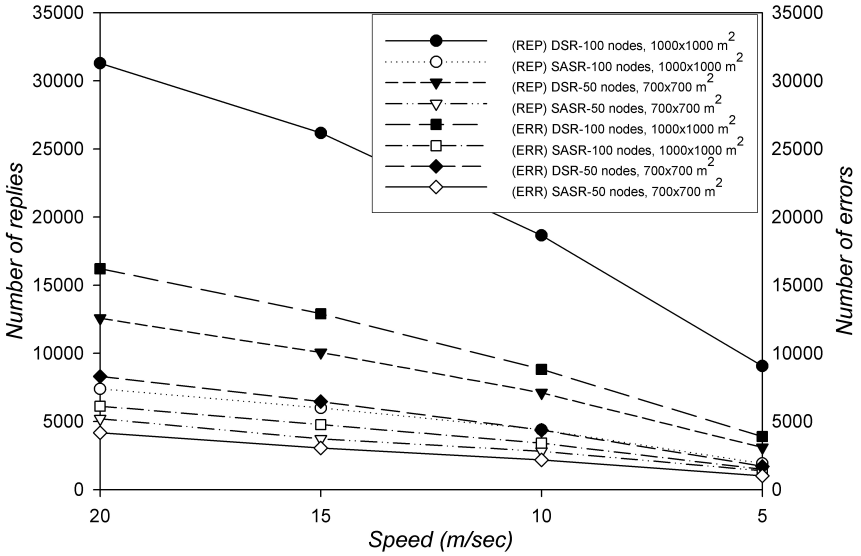

Figure 2. Number of route replies and errors with respect to host speed for the 50 and 100 node models.

excluded from use, leading to fewer errors. This is confirmed by Figure 2 where the total number of route errors is presented. In conditions of increased mobility a growing percentage of cache information becomes stale due to the rapidly changing network topology. However, DSR has no mechanism for identifying stale information therefore it is not able of limiting its spread in the network. This is shown in Figure 2 where the number of route replies is also presented. As can be seen, the contamination of caches takes place in a increasing rate and is more intense in the network of 100 nodes, since the cache capacity of the network is bigger. On the other hand, SASR manages by using sequence numbers, to almost eliminate use of stale information. The confirmation is presented in Figure 3 where we illustrate the percentage of replies that are originated from a cache. SASR produces fewer replies than DSR (Figure 2) and at the same time a smaller percentage of these replies comes from caches. Furthermore, Figure 3 is a confirmation that SASR prevents spreading of false routes to other caches of the network, leading to the reduction of the mean cache size and therefore alleviating memory and search time requirements.

Another useful inference is related to the impact of network size in the performance of the two protocols. Indeed, as it is illustrated, in the case of the 100 nodes scenario, the divergence of the protocol performances increases sharply compared to the 50 node model. It is

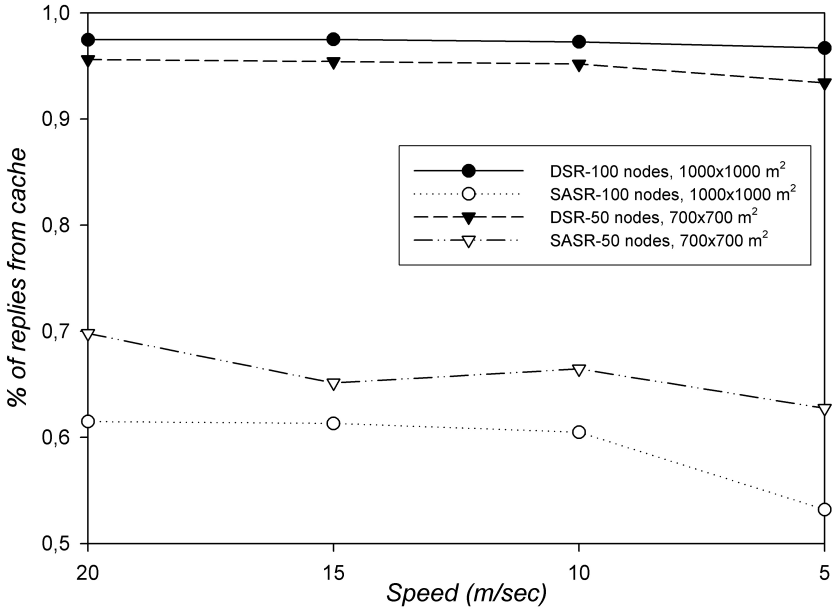

Figure 3. Percentage of replies from cache with respect to host speed for the 50 and 100 node models.

understood that in the case of the 100 node model, discovered routes are longer, since the density of the network is fixed, therefore the possibility of becoming obsolete is greater. At the same time, the cache capacity of the network is increased. The combination of these two facts has the effect of degrading DSR's performance by approximately 14% in the case of the highest mobility while at the same time the performance of SASR is slightly degraded by approximately 3%.

In Figure 4, the normalized routing load is illustrated for different mobility levels. That is the average number of control and data packets per data packet transmitted. This metric captures the protocol's channel access efficiency as the cost of channel access is high in contention-based link layers [6]. Clearly, SASR outperforms DSR over the entire speed range. Again the basic reason for this behavior is the increased number of errors taking place in the case of DSR (Figure 2). SASR on the other hand not only minimizes errors but also limits the number of generated replies (Figure 2). Indeed, diminishing the use of stale cached routing information alleviates the number of route reply packets, improving the overall performance in terms of routing load. It must be noted that the situation is worst for DSR in conditions of high mobility and mainly when the overall number of caches in the network

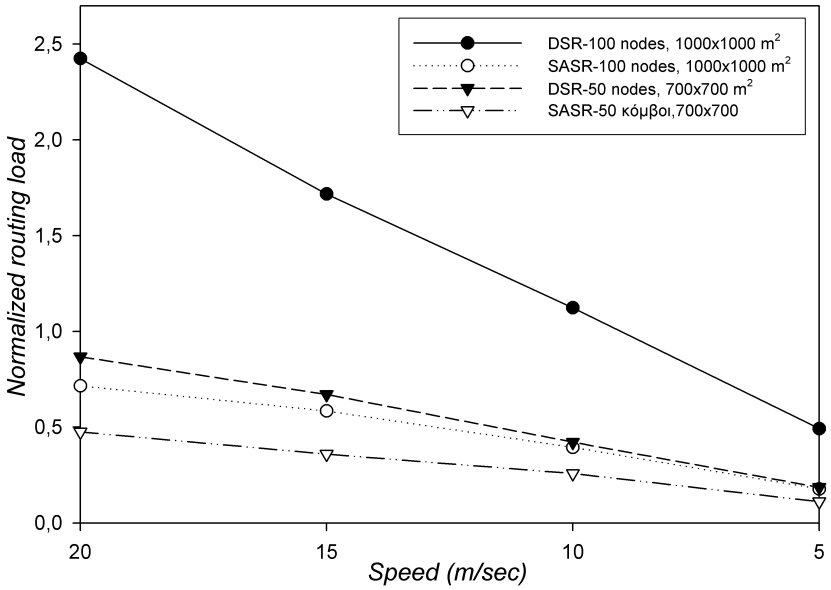

Figure 4. Normalized routing load with respect to node speed for the 50 and 100 node models.

(number of nodes) increases since the effect of contamination with stale routes is more intense.

The impact of network size in a protocol's performance is of great importance. SASR presents an improved performance in this field too. It manages to restrain the increase in routing load incurred by the growth of network size. The presented improvement is owned to the actual number of route requests and replies performed. The advantage of SASR is even bigger if we bear in mind that as indicated in Section 3, SASR uses significantly less bits for constructing request and reply packets.

To complete the performance comparison of the two protocols under different mobility levels, the average delay is presented in Figure 5. Although SASR is characterized by higher average delays compared to DSR, this is clearly not a shortcoming of the algorithm and in general relates to the fact that average delay and delivery ratio are correlated. Two are the basic reasons that can be identified for this behavior. First, SASR manages to deliver more packets therefore queues formed in the network are longer, contributing to average delay. Second, when the delivery ratio is relatively small, it is expected that packets delivered to distant hosts are more likely to be dropped. Therefore, the sample of

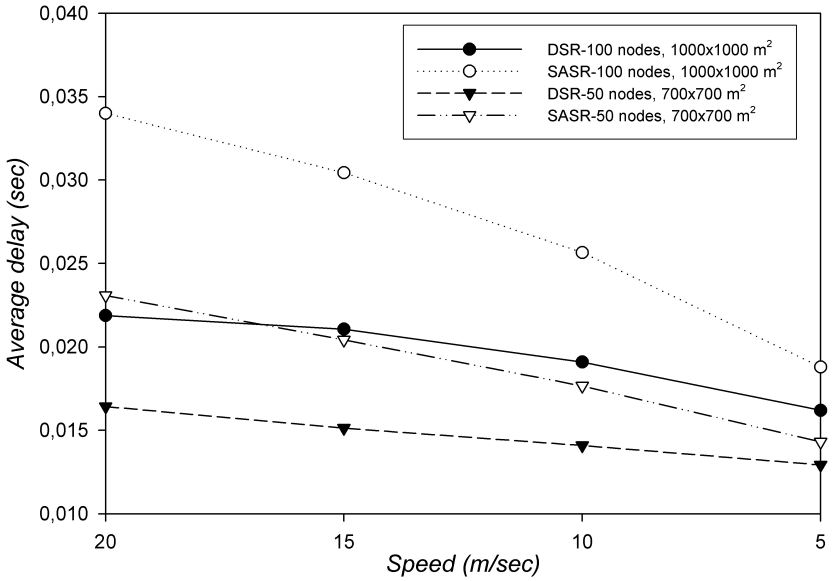

Figure 5. Average delay with respect to node speed for the 50 and 100 node models.

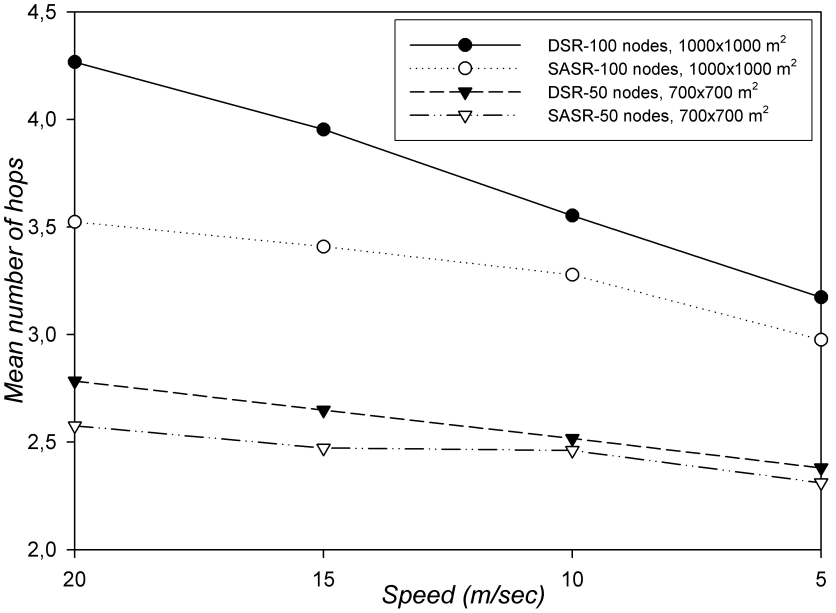

Figure 6. Mean number of hops with respect to node speed for the 50 and 100 node models.

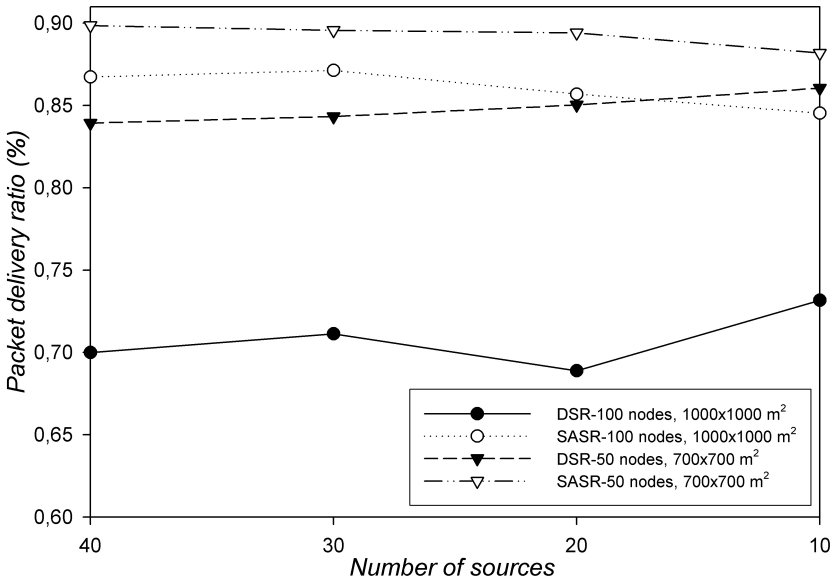

Figure 7. Packet delivery ratio with respect to number of sources for the 50 and 100 node models.

packets used to calculate the average delay in the case of DSR, is biased in favor of those traveling a small number of hops. A confirmation of this is that the difference between SASR and DSR is greater in the 100 node model where the average path length is greater. Another confirmation of the overall reasoning is presented in Figure 6 where the average number of hops taken by all the packets transmitted in the network, regardless of whether they were delivered to the destination or not, is presented. Clearly the increased average delay for SASR is not the result of using longer routes to the destination.

4.4. Varying number of sources

Another interesting aspect to explore is the performance of protocols for different numbers of data sources in the network. In this way we can examine the behavior of both protocols under varying offered load. The average packet delivery ratio is presented in Figure 7. As hinted by the graph, SASR demonstrates better performance than DSR, regardless of the offered load. However, both protocols are almost unaffected by the increase of data sources. The increased number of active connections in the network means that each link is used by more than one connection with a higher probability. As a result each link is monitored by

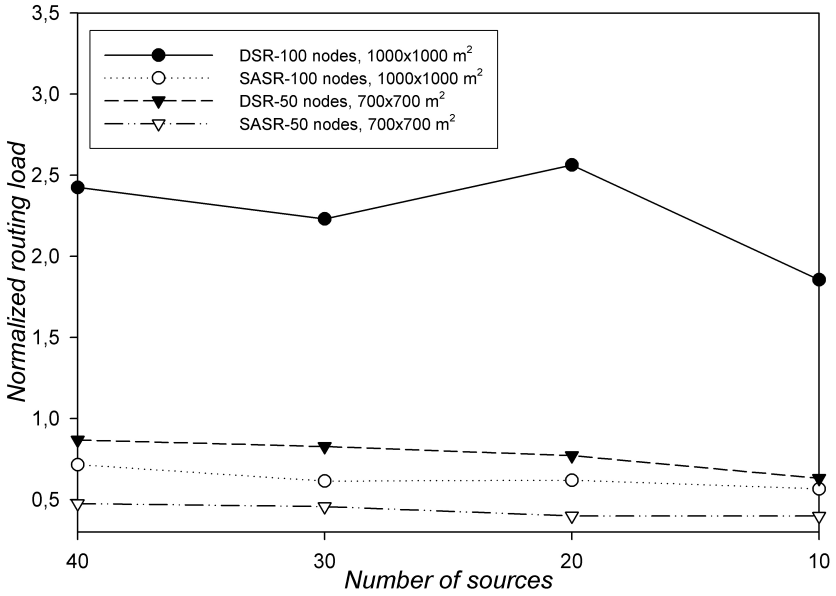

Figure 8. Normalized routing load with respect to number of sources for the 50 and 100 node models.

the maintenance procedure more frequent, therefore leading to timely detections of route errors. Therefore even when an increased routing load is offered (40 sources) the delivery ratio remains unaffected. The case for SASR is not the same since it relies more to sequence numbers than the route maintenance mechanism for avoiding errors. That is why SASR achieves a greater delivery ratio. On the other hand DSR drops packets that are already in their path to destination when an error occurs although packet salvaging is employed.

A confirmation of SASR's behavior is the normalized routing load, presented in Figure 8. SASR produces fewer route error packets since it uses sequence numbers to avoid route errors and not route maintenance to fix them. On the contrary, DSR can only take action after the route error is detected. In the last figure, we present the average delay for the same scenario. In terms of this metric, SASR performs worse than DSR. As mentioned before, this is mainly a result of longer queues in the network due to the increased delivering capability of SASR.

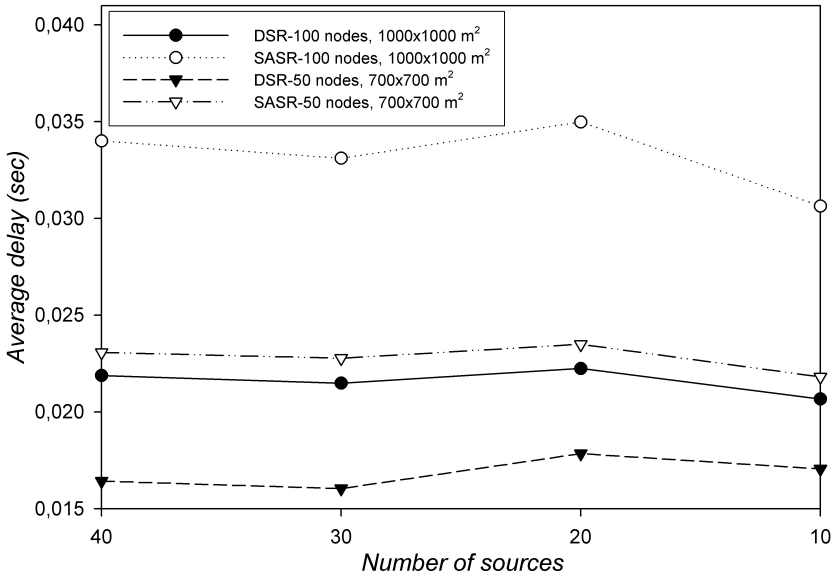

Figure 9. Average delay with respect to number of sources for the 50 and 100 node models.

5. Conclusion

In this paper we presented a new reactive routing protocol for ad hoc networks called SASR. It introduces the combined use of sequence numbers and caching. Although caching of routes may be, to some extend, beneficial for the network, it suffers by the dissemination of stale routing information in network caches which results in performance degradation. SASR takes advantage of sequence numbers to perform a passive detection of obsolete routes and block their use in route replies. As a result, only valid routes are provided for use in route replies leading to increased network performance in terms of both delivery ratio and routing load. Furthermore, by limiting false cache replies, SASR provides caches of smaller size, making feasible their maintenance. Another improvement introduced by SASR is the limited use of source routing only in the route reply phase of the protocol which results in the minimization of its bandwidth requirements. Moreover, SASR may operate without implementing source routing to deliver data packets, further reducing the induced routing overhead. The proposed scheme has been evaluated through extended simulation trials under various metrics and scenarios, proving its improved performance.

References

1. Abolhasan, M., T. Wysocki, and E. Dutkiewicz: 2004, 'A review of routing protocols for mobile ad hoc networks'. *Ad Hoc Networks* **2**, 1–22.
2. Agarwal, S., A. Ahuja, J. P. Singh, and R. Shorey: 2000, 'Route-Lifetime Assessment Based Routing (RABR) protocol for mobile ad-hoc networks'. In: *Proc. IEEE (ICC' 00)*. pp. 1697–1701.
3. Bharghavan, V., A. Demers, S. Shenker, and L. Zhang: 1994, 'MACAW: A Media Access Protocol for Wireless LAN's'. In: *Proc. ACM (ACM SIGCOM' 94)*. pp. 212–225.
4. Broch, J., D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva: 1998, 'A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols'. In: *Proc. ACM/IEEE Fourth Annual International Conference on Mobile Computing and Networking (MobiCom' 98)*. Dallas, USA.
5. Chen, T.-W. and M. Gerla: 1998, 'Global State Routing: a new routing scheme for ad-hoc wireless networks'. In: *Proc. IEEE (ICC' 98)*.
6. Corson, S. and J. Macker: 1999, 'Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations'. RFC 2501.
7. Dube, R., C. Rais, K. Wang, and S. K. Tripathi: 1997, 'Signal Stability based adaptive routing for Ad Hoc Mobile networks'. *IEEE Personal Communications* pp. 36–45.
8. Gunes, M., U. Sorges, and I. Bouazizi: 2002, 'ARA: the ant colony based routing algorithm for manets'. In: *Proc. ICPP Workshop on Ad Hoc Networks (IWAHN' 02)*. pp. 79–85.
9. Iwata, A., C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen: 1999, 'Scalable Routing Strategies for Ad Hoc Wireless Networks'. *IEEE J. Select. Areas Commun.*
10. Johnshon, D. B., D. A. Maltz, and Y.-C. Hu: 2004, 'The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)'. IETF Internet Draft.
11. Johnson, D. and D. Maltz: 1996, *Mobile Computing*, Chapt. Dynamic source routing in ad hoc wireless networks. Eds. Norwell, MA: Kluwer.
12. Maltz, D., J. Broch, J. Jetcheva, and D. Johnson: 1999, 'The effects of on-demand behaviour in routing protocols for multihop wireless ad hoc networks'. *IEEE J. Select. Areas Commun.* **17**(8), 1439–1453.
13. Park, V. and M. Corson: 1997, 'A highly adaptive distributed routing algorithm for mobile wireless networks'. In: *Proc. IEEE (INFOCOM' 97)*.
14. Perkins, C. E. and P. Bhagwat: 1994, 'Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers'. In: *Proc. ACM (ACM SIGCOMM' 94)*. pp. 234–224.
15. Perkins, C. E. and S. R. D. Elizabeth M. Royer: 2003, 'Ad Hoc On-demand Distance Vector (AODV) Routing'. RFC 3561.
16. Royer, E. M. and C.-K. Toh: 1999, 'A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks'. *IEEE Personal Communications* **6**(2), 46–55.
17. Santivanez, C.: 2000, 'Asymptotic Behavior of Mobile Ad Hoc Routing Protocols with respect to Traffic, Mobility and Size'. Technical Report TR-CDSP-00-52, Cender for Communications and Digital Signal Processing (CDSP).

18. Santivanez, C., B. McDonald, I. Stavrakakis, and R. Ramanathan: 2002, 'On the Scalability of Ad Hoc Routing Protocols'. In: *Proc. IEEE (INFOCOM' 02)*.
19. S.R.Das, C.E.Perkins, and E.M.Royer: 2000, 'Performance Comparison of two On-demand Routing Protocols for Ad Hoc Networks'. In: *Proc. IEEE (INFOCOM' 00)*. pp. 3–12.
20. Tanenbaum, A.: 2003, *Computer Networks*. Prentice Hall, fourth edition.
21. IEEE Computer Society LAN MAN Standards Committee: 1997, 'Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications'. IEEE Std 802.11-1997, The Institute of Electrical and Electronics Engineers.
22. Wang, Z. and J. Crowcroft: 1996, 'Quality-of-service routing for supporting multimedia applications'. *IEEE J. Select. Areas Commun.* **14**(7), 1228–1234.