Center for Communications and Digital Signal Processing (CDSP)

TECHNICAL REPORT

TR-CDSP-99-50

SOAP : a Self-Organizing, Adaptive Protocol for routing in large, highly mobile ad-hoc networks.

C. Santiváñez and I. Stavrakakis

Department of Electrical & Computer Engineering
Northeastern University, Boston, MA 02115.

(email: cesar(ioannis)@cdsp.neu.edu)

June 15, 1999

SOAP : a Self-Organizing, Adaptive Protocol for routing in large, highly mobile ad-hoc networks.[*]

César Santiváñez   Ioannis Stavrakakis

Department of Electrical & Computer Engineering
Northeastern University
Boston, MA 02115, USA

Abstract

Traditional routing protocols designed for mobile ad-hoc network (MANET) make no attempt to learn and exploit the instantaneous network structure, mobility and traffic patterns. As a consequence they fail to adapt to varying network conditions. Here, a self-organizing protocol that attempts to extract and utilize the current network structure for routing in large, highly mobile MANET networks is presented . Some extensions to heterogeneous scenarios (e.g. fixed-mobile integration) are also presented. Future work will be directed toward designing a multi-mode routing protocol that adapts to a wider range of scenarios from almost static to highly mobile, from homogeneous to heterogeneous networks.

[*]This research is supported in part by the National Science Foundation and the GTE Corporation

1 Introduction

For the past years, routing in mobile networks has been mainly based on the Mobile IP protocol [1], which assumes a hub-based architecture with nodes organized around fixed access points that execute all the routing functions. Traditional Mobile IP relies on the presence of a Home Agent (HA) with knowledge of the Mobile Host (MH) location at any point of time. Packets with a MH as its destination are sent first to the HA which, in turn, forwards them to the MH. Mobile IP may result in great waste of bandwidth since even if the destination MH is close to the source, all the packets have to go through the HA. On the other hand, mobile IP does not require the source to modify its IP kernel nor to be mobile-aware. Recently, a route optimization mechanism has been proposed [2] to improve the efficiency of mobile IP. Such route optimization is achieved by exploiting the binding between a MH and its Care-of-Address. The Care-of-Address of a MH is a router in the foreign network (where the MH is currently located) that agrees to provide service to the mobile host. The association between a MH and its Care-of-Address is called mobility binding or simply binding. In the earlier Mobile IP implementation, the HA intercepts all the packets addressed to a mobile host and then encapsulates and tunnels them to the Care-of-Address. An encapsulated packet will have the address of the Care-of-Address as the IP destination address and the address of the actual destination inside the IP payload. The route optimization extension in [2] allows the current Care-of-Address information to be passed along to the source node using special binding update and binding warning messages. Once the source node has the information of the destination's current Care-of-Address it will encapsulate and tunnel the packet itself, allowing the packet to go through the most cost-effective path toward the destination. The price paid for this performance improvement is that the source (usually a fixed host) should become mobile-aware.

More recently, the study of Mobile Ad-hoc Networks (MANET), has received considerably attention [3]. The more important characteristic of an Ad-hoc Network – where a set of hosts with different degree of mobility and routing capabilities group together – is not the lack of structure but the lack of knowledge regarding the (possibly dynamic) structure at a given point in time. A traditional cellular-based network can be seen as a particular state of an Ad-hoc Network. Other configurations may also be allowed in order, for example, to improve the spatial reuse factor (a campus of a university), or to guarantee the network survivability (a military operation), or when a quick deployment is necessary (disaster recovery, military operation, etc.). There are some proposals in the literature for implementing efficient MANET networks. Among the most interesting of these protocols are the TORA [4] and the ZRP [5] protocols.

TORA is a tracking protocol where each node has an unique height associated with each of the possible destinations. In TORA, the packets' flow direction is always from a higher to a lower node (downstream links). A node reacts to topological changes (lost of its last 'downstream' link) by increasing its own height, leading to incoming link reversal. In TORA the effect of topology changes are kept local, there are no loops, and several alternate routes toward a destination are provided.

Simulation results in [6] show that for medium to high mobility networks TORA outperforms ILS (Ideal Link State). On the other hand, TORA provides information about only the neighboring links not allowing for the computation of a path (succession of links toward a destination) metric or the identification of a best-cost route. TORA overhead and memory requirements increase linearly with the number of nodes in the network.

Other alternatives for routing in MANET Networks have been proposed as for example the Advance On-demand Distance Vector (AODV) [7] protocol, the Dynamic Source Routing [8], etc. They all have scalability problems. It seems that in order to cope with large networks a hierarchical approach needs to be introduced.

ZRP (Zone Routing Protocol) is an example of hierarchical routing for MANET networks. Each node is the center of its own *zone*, which is composed of all its *k-neighbors* (nodes that are at a distance of k hops or less from it). The nodes that are exactly at a distance of k hops from the center of a *zone* are called *border nodes* of that *zone*. All nodes propagate changes in their link states (or their distance vector) within their *zone* according to the IntrAzone Routing Protocol (IARP), keeping changes local. When a source needs to send a packet to a destination, it first checks if the destination is within its *zone*. If the destination is within the *zone* of the source node, the latter will forward the packet to the destination using the information provided by the IARP protocol. If the destination is not within the source's *zone* then the IntErzone Routing Protocol (IERP) is invoked. The IERP relies on a procedure called *bordercast* in which unicast packets are sent to the border nodes only. A source node will *bordercast* a REQUEST to its border nodes. The border nodes will know (thanks to their own IARP protocol) whether the destination is within their *zones*. If a border node has the destination within its *zones*, it will forward the packet to the destination. If the destination is not within a border node's *zone* the border node will propagate the REQUEST to its own border nodes and so on. Some mechanisms are included to prevent looping back or revisiting regions of the network that have already been queried. When a REQUEST reaches a destination, the succession of border nodes traversed is recorded and returned to the source node which stores it in its (IERP) routing table. ZRP only provides one route and if a change occurs, the source has to run a new route discovery procedure. In large mobile networks, it is highly likely that changes along the path will occur during a session, therefore ZRP without a route recovery procedure is not well suited for a large, highly mobile environment.

In a mobile network there are two main issues that need to be considered. One is the cost of choosing a sub-optimal path at a particular point in time (bandwidth waste), the other is the cost of tracking destinations due to mobility. Traditional protocols that address the first issue in a static network with moderate success have a poor performance in a highly dynamic environment, not only because of the instability typically associated with them, but because the bandwidth overhead they require to operate correctly increases significantly with the mobility rate. For highly dynamic networks the bandwidth overhead present in typical Link State or Distance Vector algorithms may consume the greater part of the available bandwidth, also the convergence time of these algorithms may be unacceptable. It is clear that in such cases the availability of optimal routes is not of primary importance if the protocol itself congests the network. An alternative protocol that reduces the network congestion due to flooding of routing information is preferred, even if the routes obtained are not optimal. The degradation in network performance due to sub-optimal routes would be less compared to that due to excessive overhead due to flooding of routing information – proportional to the rate of topological change. It is clear that in highly dynamic networks it is the second issue (the cost of tracking mobility) the single most important factor in the network's performance. Therefore, it is desired that a protocol tries to minimize the cost involved in route discovery, instead of trying to minimize the cost of the route itself.

An efficient routing protocol must take into account the spatial-temporal correlation inherent in any network and should also consider the network behavior and structure, trying to take advantage of them. In the present report a Self-Organizing Adaptive Protocol (SOAP) that addresses these requirements, reducing the cost associated with route discovery is presented. The SOAP protocol is based in part on concepts presented in the ZRP, TORA, and Mobile IP protocols and the concept of 'footprint size' introduced in [9], as will be explained in the next sections.

2 Protocol Overview and Motivation

Suppose that a large network is divided in four regions, each of them associated with one of the four cardinal points (N,S, E, and W). Consider a source node which desires to send a message to a destination node in region W. Furthermore, assume that the source knows that the destination is within region W but does not have any route toward this destination node. In this scenario the source node may initiate a route discovery procedure that will result in a broadcast (a bordercast if using ZRP) of a REQUEST over the entire network.

Alternatively, the source node could begin forwarding the data packets (not the REQUEST) in the "West" direction, hoping that along the way the packets will be heard by a node that has knowledge of some routes toward the destination. In the latter case a broadcast will not be necessary. It is obvious that this second alternative is by far more attractive than the first. To implement this second approach, however, two major problems are encountered. First, the nodes may not know which the West direction is [1]; and second the source does not always know in which region the destination currently is.

The first issue can be addressed if one node is chosen inside each region as a "beacon". These nodes may be for example some nodes in the center of each region. Each node serving as a "beacon" is referred to as *reference node* and the region it belong to is referred as *reference area*. Then, the

[1] Unless some form of geographical routing is being used. In that case, nodes will learn their position from built-in GPS devices.

TORA protocol can be used to track these four reference nodes [2]. Thanks to TORA all the nodes (possible sources) have downstream links toward each reference node/area. For example, let N_W be the reference node of the reference area (region) W. All the nodes (and particularly the nodes outside W) has downstream links toward N_W and therefore they have downstream links toward the reference area W. These downstream links provide each node with a sense of direction toward reference area W. Thus the first issue may be addressed but with the cost of creating reference nodes/areas and tracking them.

The second issue can also be addressed with the inclusion of a location management scheme that takes into account the past history of a node. For example, if a node was in the recent past inside the W region it is unlikely that it is far away from that region. Thus, the location management will initiate the search for a node in the last recorded location and from there will follow the path (along reference areas) followed by the node. After the packet reaches the destination, the destination will inform the source of its new reference area (location) and the subsequent packets will be sent directly to the current reference area.

The previous approach will reduce the number of broadcast (or bordercast) that other protocols produce. In particular, this protocol may be seen as an improvement over ZRP that, with an additional cost, dramatically reduces the number of bordercast in a large, dynamic, highly loaded network. It should be noted, however, that for some nodes – due to their high mobility or low usage – it may not be cost-effective to maintain reference nodes and track their location. For those nodes other alternatives such as bordercasting or even flooding may be considered. The proposed protocol also considers these situation by using ZRP to route packets to these nodes when the extra cost (over ZRP) required to maintain reference nodes to these destinations is greater that the expected cost due to bordercast to these nodes.

In the proposed Self-Organizing Adaptive Protocol (SOAP) each node is running a modified version of the ZRP protocol – as will be explained in the next sections – and therefore each node is the center of its own *zone*. The clustering mechanism presented in the next section will determine that some nodes become reference nodes. The zone – set of k-neighbors – of each reference node will constitute the base of its associated reference area.

The SOAP clustering mechanism assigns every node in the network one of three possible values : *assigned, around, free*. A node is *assigned* if it is inside a reference area. A node is *around* if it was inside a reference area in the past but currently is outside the reference area but less that 2k hops away than that reference area's reference node; i.e. at least one of the reference node's border nodes has the *around* node inside its zone and therefore has up-to-date routing information about it. A node is *free* if it is not inside or around (see above) any reference area. Figure 1 shows an example of a network that at a given point in time has organized itself into three reference node/areas. The

[2] If geographical routing is being used then tracking of the *reference node* is not necessary. It would suffice that each *reference node* advertise their current location after some traveled distance.

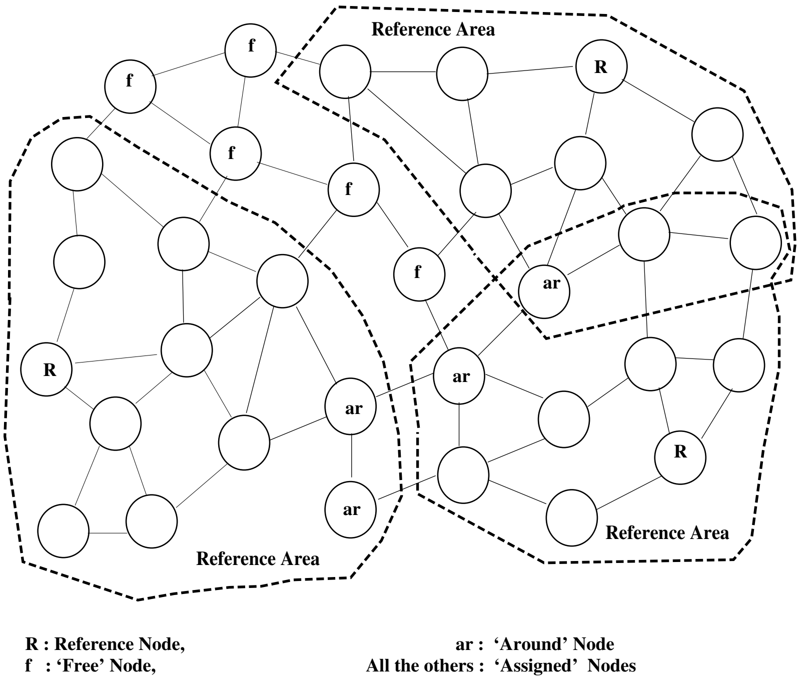

Figure 1: A self-organized network with three reference areas; k is equal to 2.

value of k has been set to 2. There are four nodes with *around* status. These nodes were inside one reference area a while ago. There are also 5 *free* nodes, since it is not cost-effective to group them into a new reference area. It is also shown that the reference areas may be overlapping and that one node determines its reference area not only based on distance but also taking into account the immediate past. It should be pointed out that although each node is the center of its own *zone*, there are only three reference areas each associated with one reference node.

The SOAP protocol also requires that every node has a *destination table* with the addresses of all the reference nodes and has its neighboring links market as 'upstream' or 'downstream' with respect to any of the reference nodes. In Figure 2 the links towards the reference node R are shown. The downstream links form a directed acyclic graph (DAG), with node R as the root. TORA provides a similar DAG for each reference node. The nodes also have a *location table* with an entry for every destination in the network[3]. Each entry in the location table will have the address of the reference node of the last recorded position (reference area) of the node being pointed out for that entry. Different nodes will have different values in the same location table entry, since some nodes may have more up to date information that others. It is also assumed that when a node moves from a reference area to another, the moving node will inform its past reference node (and only it) of its new reference node address. In this way if a packet arrives to a node that has been previously a reference node for the destination node, the reference node will forward the packet to the reference node in the next visited area. This procedure will continue until the packet reaches the current reference area of the node. The previous procedure allows for the sources not to have an up-to-date information on their *location tables* but still be able to reach their destinations. It

[3]The mechanisms to initialize and maintain the *location table* are discussed in section 4.

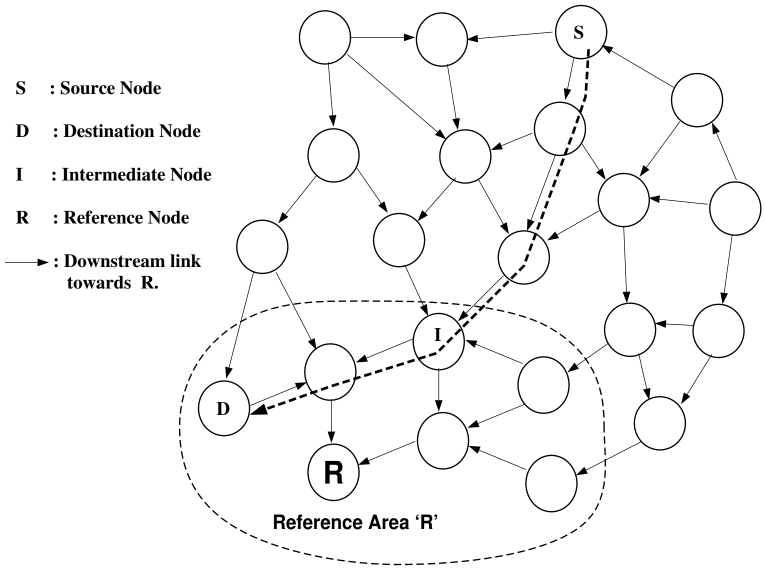

Figure 2: Packet routing using AHRP when location (reference area) information is up-to-date.

should be noted that it is also possible that a node moves away from a reference area and does not enter another reference area. In such a case the last reference node will regard itself as the last reference node visited by the destination node.

If all the information is available, the proposed protocol will work as follows when forwarding packets: when a node requires to send a packet toward a destination it checks first if the node is within its own zone (IARP). If not it looks at its location table for the entry associated with the destination node. If the entry is empty, the source executes the IERP of ZRP, bordercasting a REQUEST through the network. If the location table entry is not empty, the source node sets the D_ENCAPSULATION flag on and encapsulates the packet putting the reference node address (Care-Of-Address) as the destination address and putting the actual destination address inside the IP payload (mobile IP). Also, the source puts its own reference node address as the IP source address and puts its actual address in the IP payload. This way the destination node (and all the nodes along the way) may update their location table entries, making it simple to return packets to the source node. The source node then sends the packet through the network using one of its downstream links associated with the destination address (as in the TORA protocol).

Each intermediate node in the network will look at the set D_ENCAPSULATION flag and will recover the actual destination address from the payload. If the destination is within the intermediate node's zone then the latter will forward the packet directly to the destination, but without decapsulating it. If the destination packet is not within the intermediate node's zone it will forward the packet through one of its downstream links associated with the Care-of-Address (TORA protocol). If the Care-of-Address is not a reference node, it must be within the intermediate node zone (as is the case for the *around* nodes, as explained later); in such cases the intermediate node forwards the packet in the direction of the Care-of-Address as if it were using IARP.

When a reference node receives the packet, it recovers the destination address and checks if the

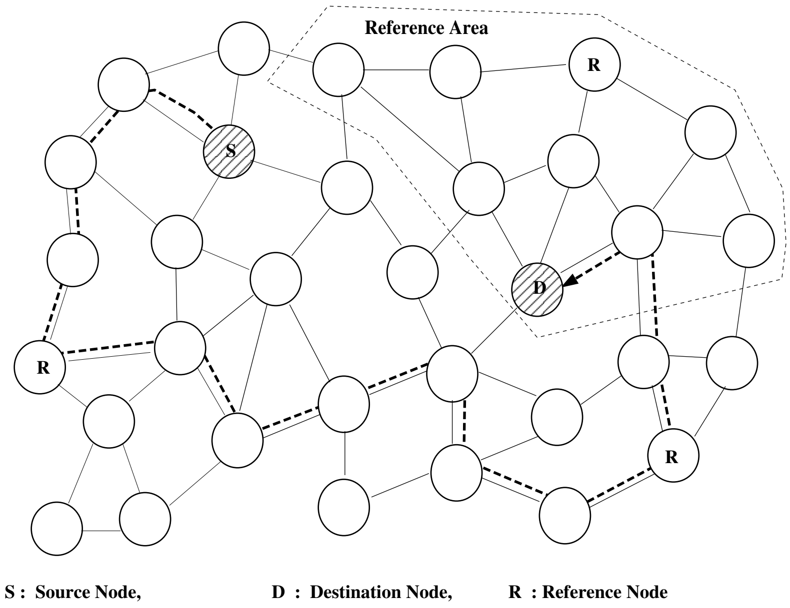

Figure 3: Packet routing using AHRP when location (reference area) information is out-of-date.

node is within its zone. If the node is, then it decapsulates the packet, set the D_ENCAPSULATION flag off and forwards the packet directly to the destination. If the destination node is not within the reference node's zone it checks if the node is reachable (status *around*, i.e., less than 2k-hops away). If the packets is reachable it re-encapsulates the packet putting as Care-of-Address the border node that is closer to the destination node (note that in this case the Care-of-Address is not a reference node). If the destination node is not *around* the reference node, the latter will look at its location table and will forward the packet to the reference node pointed to in the location table, putting the new reference node address as the care-of-address and forwarding the packet to one downstream link. If the reference node pointed out to in the location table is itself, the reference node will assume that the node is "unreachable" and will send a packet to the source node which will update its location table and will bordercast a REQUEST for the destination node.

Once the packet is received by the destination node, it will recover the source's reference node's address and put it in its location table. The destination node will use the new reference node address as Care-of-Address for any subsequent packet sent to the source node. Also any subsequent packet sent by the destination to the source will have the current destination's reference node's address and therefore the source may update its location table (Route Optimization procedure). Therefore, after the initial packets has been interchanged both source and destination will have knowledge of the most up to date location information of each other. Figure 2 shows an example of the path followed by a packet when up-to-date location information is available (the middle of a session). Figure 3 shows the path followed by a packet in a case in which the location information is out-of-date. The packet will visit the past destination's reference nodes until reaching the current reference area. This long path is expected to be used only at the beginning of a session, since the destination node may include its current location information in any packet (e.g. acknowledgments) it sends back to the source, allowing subsequent packets to follow a more direct path.

From the above discussion, it may be seen that a bordercast is no longer necessary for the *assigned* and *around* nodes, therefore significantly reducing the number of bordercasts in the network.

It should also be pointed out that during normal communication, a packet addressed to an *assigned* node does not necessarily have to go through the destination's reference node, but any node in the destination zone may forward it. Since it is expected that the majority of the nodes in the network will have status *assigned* it can be said that the proposed protocol attempts to distribute the load within the network. However, it should be noted that packets addressed to *around* nodes are more likely to pass through the destination's reference node; and also packets with outdated Care-of-Address will require to pass through a sequence of reference nodes before reaching destination. These two operations will result in an increase in the traffic going through the reference node's links somehow creating an unfair traffic load (burden) on the reference nodes. Nevertheless, this effect may be reduced (or eliminated) if non-reference nodes forwarding packets to *assigned* nodes within their zones (the most of the cases) are instructed to choose paths that avoid interfering with reference node's links, whenever possible. In this way, the reference nodes are 'relieved' of some traffic burden and the network maintains some good balance.

The mechanism to allow this protocol to work properly are described in the next sections. The creation and maintenance of the "reference" nodes is discussed in Section 3. The location management procedure is explained in Section 4. The protocol is presented in section 5. Some applications and modifications for particular situations are discussed in Section 6, and finally some conclusion and new research directions are treated in Section 7.

3 Clustering

The Clustering algorithm is responsible for choosing the reference nodes and for assigning to every node some of the following values {*assigned, around, free*} with respect to a reference node. A node is *assigned* if it is inside a reference node's zone, i.e. it belongs to a reference area. A node is *around* a reference node if it was *assigned* to the reference node but now it is outside the reference area but less than $2k$ hops away, so that at least one of the reference node's border node is inside the node's zone. A node is *free* if it is not associated with any reference node, probably because it is not worthy tracking this node location. In Figure 1 an example of a clustering is shown.

As an initial point for this discussion let's consider a network that is running the ZRP protocol. From the discussion above (Section 2), it is clear that there is a benefit in grouping all the nodes *assigned* with a reference node in a reference area, mainly because the number of bordercasts needed is significantly reduced with respect to ZRP since bordercast are needed to reach *free* nodes only. The objective of the clustering algorithm is to choose the nodes which are the best candidates for being a reference node in the sense that they maximize a 'Gain' metric. This Gain will be compared with a Cost metric (extra bandwidth overhead with respect to ZRP) of maintaining a reference area.

In the following subsections the Gain and Cost functions are presented together with an heuristic for (suboptimal) clustering of the network.

3.1 The Gain Function

In [9] the concept of 'footprint size' has been introduced. The footprint size is equal to the number of neighbors of a node. Nodes may be classified based on their footprint size. The larger a node's footprint size the larger the region (in number of nodes) that the node covers and the less likely that a random node will leave that region in the near future. Therefore, a node with higher footprint size would be a better candidate to be a reference node, since it would serve as a beacon for a greater number of nodes. Here, the concept of footprint size is extended to a gain function that will be employed to determine the nodes which are the most suited to be reference nodes.

The Gain function at a level T and at time t for a reference area is defined to be equal to the expected number of bordercasts saved over the next T seconds. A bordercast will be saved if at the beginning of a new *session* the destination node is either *assigned* or *around* a reference node. Here, the term *session* is applied to a sequence of packets not more than T_s seconds apart. For example two consecutive file transfers to the same destination will be considered to be part of one session whereas in a transaction in which the packet interarrival time exceeds T_s each packet will be considered to form a new session.

In the ZRP protocol, a bordercast is needed when a route is lost due to mobility (two nodes that were k-hops away get out of reach), whereas in the proposed protocol there will still be routes available. Therefore the gain function will be equal to the number of new sessions with destinations inside the candidate reference area plus the number of sessions currently in progress with broken path over T.

Let A be a potential reference node and let $V(A,t)$ be its zone (reference area) at time t; let $G_T(A,t)$ be the gain function of node A at time t and at level T. Each node $i \in V(A,t)$ has two parameters : $S_i(A,t)$ and $R_i(t)$. $S_i(A,t)$ is equal to the percentage of the next T seconds that node i will stay inside node A's zone, i.e. within a radius of k from A; $R_i(t)$ is equal to the total expected number of new plus broken sessions – destined to node i – over the next T seconds. Then, $G_T(A,t)$ can be defined as :

$$G_T(A,t) \quad = \sum_{i \in V(A,t)} S_i(A,t) R_i(t) \; x \; (\text{cost of a bordercast}) \qquad (1)$$

Different approaches can be followed to estimate the values of $S_i(A,t)$ and $R_i(t)$, depending on the desired amount of complexity. For $S_i(A,t)$, a first approach could be to set its value to be equal to the inverse of the distance (in hops) between nodes A and i elevated to an integer exponent.

A better choice may be to consider the percentage of the previous T seconds that the node was inside $V(A,t)$ and assume that this will be repeated. A more complex approach (and likely a more successful one) will be to consider the node i distance history over the past T seconds and estimate its trajectory (incoming, outgoing, etc.). Different estimation techniques may be considered to determine the more accurate result for a limited complexity implementation.

For the value of $R_i(t)$, the more simple approach will be to associate a value equal to 1 for nodes already in a session (likely to be broken) and a fixed small value for nodes not in a communication. This approach will result in a gain function closely related to the footprint size [9]; that is, nodes with greater footprint size will have greater gain. A more complex approach should take into consideration different traffic patterns for different classes of users. For example in a client-server architecture, it is likely that the server receives constant requests resulting in a large number of new sessions, however these sessions will be short resulting in almost no broken sessions. On the other hand, if the node is handling voice communications it is likely to have a small number of new sessions, but since their length is greater it is more likely to have broken sessions. Therefore, the class of user has to be taken into account along with past history to estimate node i's traffic characteristics.

Let $n_{i,s}$ be the expected number of bordercasts required by a session. Obviously $n_{i,s}$ will depend on the session duration, the path length (the larger the path the more likely to be broken) and the rate of topological change. An initial estimate for $n_{i,s}$ may be based on the network diameter, rate of topological change, a link's mean time to failure, and mean session length. A more up-to-date estimate can be obtained in the case in which a higher level agent is present that monitors node i's number of new and broken sessions over time; this case will be considered later.

Also, since the reference nodes are required to perform some extra functionalities than the rest of the nodes, it would be convenient to also weight the computational power and available bandwidth. $R_i(t)$ is computed by each node (it is independent of the reference node) and is broadcasted inside a radius k.

It should be noted that the principal objective of the gain function $G_T(A,t)$ is to help identify the most suited node to be the reference node. Once the best candidate is found, its gain is compared against a threshold that is equal to the cost function defined in the next subsection to see if it is worthy to create and maintain a reference area.

3.2 Cost Function

The cost associated with a reference area consist of three components : the cost of creating a reference area, the cost of tracking the reference node, and the cost of continuous link state updates inside a reference area. The last cost is caused by the ZRP and is independent of the existence of a reference area. Thus, in order to decide whether to create a reference area or not, only the first two cost components need to be considered.

The cost of creating a reference node is equal to the cost of sending a broadcast packet announcing the new reference node.

The cost of tracking the reference node increases with time and is dependent on the network characteristics, mainly the average number of nodes involved in a topological change and the rate of topological changes. In general the above cost is difficult to estimate. An approach could be to use history information, that is, review the topological changes (rate and number of nodes involved) over the immediate past; recall that node A has full knowledge of the topology inside its zone thanks to IARP-ZRP [5].

Finally, the total cost will be the sum of a constant term – the cost of one broadcast – and a linearly time-dependent term – the tracking cost.

3.3 Creation of Reference Areas

A reference area should be created only when it is worthy. In the case of a potential reference node A, if there were not a reference area centered in A then the initial packets of a session sent to A (or any node inside node A's zone) should be routed using ZRP (bordercasted). This cost has to be compared with the cost associated with the creation of a reference zone in order to determine if it is worthy to create the reference zone. Therefore, the cost function defined before (Section 3.2) is a threshold against which the gain of the better suited candidate to become a reference node has to be compared.

The values 'cost of broadcast' and 'cost of bordercast' are network dependent and may be assigned a priori and updated infrequently. It is not expected that these values change significantly over time nor that the threshold that they define be critical.

ZRP considers that each node informs its k-neighbors when a change in link state occurs. SOAP considers that a node i sends extra information to its k-neighbors along with the ZRP's information. The information is sent after a change in link status or after a timer expires (as required in ZRP). The extra fields are:

$$\Big(\; Status \quad , \quad R_i(t) \quad , \quad G_T(i,t) \quad , \quad RF_p \quad , \quad RF_1 \quad , \quad \ldots \quad RF_n \; \Big)$$

Where $Status$ has three possible values : *free, around, assigned* . $R_i(t)$ and $G_T(i,t)$ were introduced in Section 3.1. RF_p is the reference node to which node i is assigned (if any) and $RF_1, \ldots, RF_n$ are , in order of distance, all the other reference nodes in whose area node i belongs as well (are node i's k-neighbors).

Additionally, a node has to send at least one update packet every T seconds. Therefore, if for the last T seconds there have not been any link status change the node will have to send an update packet. The value of T should be chosen so that the updates triggered by this time out are infrequent. Therefore T should be chosen to be greater than the mean failure time of a link. In a highly dynamic environment where the changes in link status happen frequently and for moderate values of T, most updates will be triggered by link status changes.

Node A uses the information presented above to constantly update its gain $G_T(A,t)$ considering only the *non-assigned* nodes (i.e. with status *free* or *around*) in equation (1). If $G_T(A,t)$ is not greater than the gain of all the other *non-assigned* nodes inside its zone, then no further action is taken. If $G_T(A,t)$ is greater than the gains of all node A's *non-assigned* k-neighbors, then node A is a candidate to be a reference node. In the case that two or more nodes have the same gain, the node with (for instance) the higher address will be the candidate to be a reference node.

If node A is a candidate to be a reference node then it compares its gain $G(A)$ to the cost threshold presented in Section 3.2. If the gain is greater, then node A becomes a reference node and sends a broadcast packet all over the network. There is a field in the broadcasted packet that contains the cumulative height with respect to the reference node and that is initialized to the current time plus zero, according to the TORA protocol. Node A also includes additional information in this broadcast as will be explained later.

When a node receives the broadcast of A it reads the cumulative height and increases it by one. If this height is lower than its last value (initially in NULL, and non-NULL only after the first reception of the broadcast packet) the node updates its height associated with the reference node and forwards the packets to their neighbors. After all the nodes in the network have received the broadcast from node A they all has at least one downstream link toward node A and the entire network is a directed acyclic graph (DAG) rooted at A.

After the DAG has been formed, the network will continue tracking node A using the route maintenance mechanism of TORA. The main difference with the current TORA implementation is that routes that have been deleted (because of network partitions) are being reestablished as soon as the connectivity is recovered. The links are going to be definitely erased (change to undirected state) only when node A broadcasts another packet telling that it is no longer a reference node, as explained below.

All the nodes within the zone of A change their status to *assigned* and record A as their reference node.

It should be noted that there is at least one node – the one with the global maximum gain – which has the maximum gain among its neighbors. Therefore, if the threshold (cost) is sufficiently low (favoring the creation of reference areas) then at least one reference area is created and the nodes within this area change their status to assigned. Therefore, the set of *non-assigned* nodes is reduced and a different node has the new (smaller) global maximum gain. The latter node becomes a new reference node and so on; the procedure continues until a large portion of the network is grouped in reference areas. The number of nodes left *free* depends of the value of the threshold.

It should also be noted that a node is not selected to be a reference node unless it is willing. If a node does not send the broadcast packet announcing itself as a reference node, then it will not become a reference node even if its gain is greater that its k-neighbors'. Such a node should put a value of zero instead of its actual gain in their next transmission.

The resulting set of reference areas would represent the structure (hierarchy) of the network at a particular moment in time. After this initial setup, reference areas may split or may combine and new reference areas may appear. Every time a node has a gain greater than its neighbor's and also greater than its threshold and it is willing, it will become a reference node. Therefore, it may be interpreted that every time a large set of nodes is left outside the existing reference areas these nodes will tend to form a new reference area.

3.4 Deletion of Reference Areas

All the possible changes in reference areas : creation, splitting, combination, and deletion are the result of two basic procedures, namely creation and deletion. The algorithm employed to create a new reference area has already been explained and is running permanently at each node. In this subsection the algorithm used to delete existing reference area is explained.

The deletion algorithm is based in the computation of what is called the *minimal gain*. The *minimal gain* of the reference node A is the gain of A computed considering only the nodes within the zone of A that are not within the zone of any other reference node. We can see the minimal gain as the actual contribution of node A since the other nodes could be taken care of by the other reference nodes.

For a reference node A to disappear two conditions have to be met : the *minimal gain* of node A has to be lower than the minimal gain of all its neighboring reference nodes[4] and the *minimal gain* has to be lower than the value *threshold_del*. A high value of *threshold_del* will result in frequent reference area deletion/creation and increase bandwidth overhead. Thus, *threshold_del* should be set to be a small fraction of the cost function discussed in subsection 3.2 but taking into consideration that a too small value resulting in infrequent changes in reference area topology may reduce the algorithm ability to adapt (quickly) to varying network conditions.

In order to guarantee that the two conditions are met, a reference node that detects that its minimal gain is lower than *threshold_del* sends a packet to every neighboring reference node informing them that it is going to be deleted. The node will be deleted only after all the neighboring reference nodes acknowledge its packet and agree with it. A neighboring reference node that receives a message from a reference node that is going to be deleted compares that node's minimal gain to its own and acknowledges agreeing only if its minimal gain is greater. In such a case (reference node is going to be deleted), the neighboring reference node attaches to the message its current gain along with a list with its distance in hops to the nodes it wishes to take care of (including any

[4] Two reference nodes are said to be neighbors if there is at least one other node that is within the zone (i.e. k hops away) of both reference nodes. Reference node A knows all its neighboring reference nodes based on the information being sent by all its k-neighbors as explained in previous subsection.

node with status *around*).

Before deleting itself, the reference node decides (based on the distance) which will be the new reference node (if any) for each of its *assigned* or *around* nodes. In case there is no node willing to take care of a particular node, that node will change its status to *free*. After that, it sends a broadcast message to the entire network including a list of its previously associated nodes with their new reference node's addresses. The links toward the deleting node are changed to undirected and the nodes previously associated with the deleted node record the identity of their new reference node (from the broadcast).

3.5 Handoff

Node i originally assigned to reference node A is said to have executed a handoff from reference node A to reference node B if node i decides to have node B as its new reference node.

When a node i originally in node A's reference area moves more than k hops away (outside node's A reference area) it changes its status to *around*, but it is still associated to node A. It will remain associated with node A as long as its distance in hops is less than $2k$, that is, at least one node inside node A reference area is inside node's i area. If the distances in hops between nodes A and i is equal to or exceeds $2k$ then node i status become *free* until it is associated with (less than k-hops away from) another reference node. In the latter case, node i will change its status to *assigned* again and will have to notify node A of its new reference area as required by the location management mechanism explained in the next section.

Nodes in *around* status have to send a packet to their associated reference node – in addition to their k-level broadcast (as required by the ZRP protocol) – indicating their closest k-neighbor that is inside its associated reference node area (i.e. a border node of the reference area). Based on this information, the reference node will always be able to forward packets to the *around* nodes.

If at any moment a node that has status *around* detects that it is within another reference node zone (i.e. less than k-hops away) and that the distance to its previous associated reference node is greater than the distance to the new reference node plus *extra_hops* then this node decides to execute a handoff to the new reference area. The node will change its status to assigned and will inform its previous reference node of its new reference area as required by the location management mechanism explained in the next section. The value of *extra_hops* is initially set to 2, and is intended to provide with some 'capture' effect to prevent a (border) node from being constantly oscillating between neighboring reference areas. Also, no *assigned* node executes a handoff even if the distance to another reference node is smaller than its distance to its current reference node. Thus the overhead caused by unnecessary handoffs is reduced.

4 Location Management

The objective of the location management algorithm is to provide a table, referred to as 'location table', associating any possible destinations with its last known reference node (if any).

4.1 Initialization

Each node has a location table with as many entries as known destinations. Each destination has initially associated the value NULL meaning that no information is available about the location of that destination.

When a new reference area is formed the new reference node sends a broadcast all over the network. In this broadcast the reference node indicates the nodes that belong to its reference area. All the nodes receiving this broadcast will update their location tables putting the new reference node address as the current location of the nodes mentioned in the broadcasted packet.

For example, if node A decides to be a reference node for the nodes B,C,D, and E then node A will send a broadcast packet indicating that nodes A,B,C,D, and E belong to its reference area. If node i receives this broadcast, node i will update the location table entries of nodes A,B,C,D, and E with the address of node A.

After several reference areas have been formed, a number of entries of the location tables of the network nodes will be filled. NULL entries will still be present in the location tables corresponding to *free* nodes.

4.2 Maintenance

There are the following maintenance procedures:

- When a node changes its reference node (handoff) it will inform the previous reference node - only - of its new location (reference node). The previous reference node will update the corresponding location table entry and this node will serve as a pointer toward this destination.
- In any packet sent, the source IP address will correspond to the current reference area (if any) and the actual source node address will be in the payload. A node that receives the packet will extract the source node's reference area information and will update its location table accordingly.
- After a number M of handoffs a node will send a new broadcast informing all the nodes on the network of its current position. The optimal value of M will depend on the network conditions and will be specified later. A small value of M will imply too many updates being propagated (bandwidth waste) and a too large a value of M may result in out-of-date information producing inefficient initial source-destination paths congesting the areas close to the reference nodes and degrading the network performance.

- When a reference node is deleted it sends a broadcast informing the network not only of its deletion but the new reference areas of the nodes previously inside its reference area.

5 Protocol description

The proposed SOAP protocol uses a combination of the mechanisms of the Mobile IP, TORA, and ZRP protocols. SOAP recognizes and uses the REQUEST, REPLY, and FAILURE packets of the ZRP protocol and the QRY (query), UPD (update) and CLR (clear) packets of TORA. Also a *binding update* packet is used to update the nodes' current location when needed. Instead of explaining all the details contained in these protocols, only the differences with the proposed protocol are presented.

Major differences between related functionalities in SOAP and Mobile IP

- The binding database in Mobile IP is replaced by the location table in SOAP.
- Every node in the network has a location table, as opposed to only the Home Agent being equipped with the binding database in Mobile IP.
- In SOAP it is assumed that every packet has a D_ENCAPSULATION and a S_ENCAPSULATION bit flags. The D_ENCAPSULATION (S_ENCAPSULATION) flag is set on if the destination (source) IP address is not the actual address but the address of one reference node. The actual destination (source) address is inside the payload.
- In SOAP, when a node receives a FAILURE packet, it sets the location table entry of the destination referred to by the FAILURE packet to NULL. FAILURE packets were previously used only by the ZRP protocol.

Major differences between related functionalities in SOAP and TORA

In SOAP only the reference nodes are TORA destinations so that the two terms are used equivalently (reference nodes and TORA destinations). The nodes keep a table of the current destinations (*destination table*). When a new physical link is established, the new neighboring nodes exchange among other information their height with respect to the current destinations, therefore assigning a value (upstream or downstream) to the physical link. The following modifications to TORA need to be employed, to account for TORA's role in SOAP.

- In TORA the routes toward a destination are generated after a query from the source (source initiated - on demand) while in the present protocol the initial routes toward a destination (reference node) will be constructed based on the broadcast sent by the reference node at the time of the reference area's creation. When a node first receives the mentioned broadcast it records the original sender's address and the time the reference area was created in its destination table. It will set its height to the value retrieved from the broadcast message.

Finally, the receiver node increases by one the height field received in the broadcast packet and resends it. If the node receives the same broadcast again no action is taken unless the height is lower than the one recorded. In such case, the node proceeds as before updating the height and resending the broadcast. It is clear that the route creation is less costly in SOAP than in the original TORA.

- When a new node enters the network it will receive – from its neighbors – the information about their height with respect to the current reference nodes. This way the new node will be informed of the reference nodes in the network and will set its height (for each reference node) slightly higher than the highest of its neighbors.

- Similarly, when a node that has its height toward a destination with the value NULL (possible after a network partition) establishes connectivity with a node that has a non-null height it will set its height slightly higher than its neighbor's. This means that the network will try to continue tracking a reference node as soon as some connectivity is reestablished following a network partition.

- The nodes stop tracking a destination only if the destination ceases to be a reference node. The broadcast packet sent by the deleted reference node serves as the CLR (clear) packet in the original TORA protocol. Not only the links are set to undirected (changing their height to NULL) but the reference node is removed from the list of destinations; that is, there is no height associated to that destination and the nodes stop broadcasting information related to this destination when new physical links are established. The time the reference node was deleted is recorded.

- When network partitions occur it may be possible that different nodes have different destination tables. When two nodes realize they have different destination tables they send each other their last recorded creation/deletion time. The more recent event takes precedence. The nodes update their destination tables and heights and propagate the information to their neighbors as necessary.

- TORA does not have a mechanism to choose best-cost routes. In SOAP a greedy approach is taken. In SOAP, when a node needs to forward a packet to a destination it will choose the lowest cost link among its downstream links. In this context, the cost of a link is equal to the number of nodes that shared the link over the (current) available bandwidth of the link (the same cost function presented in [9]).

- In SOAP, if a node has to forward a packet to a reference node that is unreachable (maybe because of a network partition) then the node has to send a FAILURE packet to the source node of that packet. TORA does not provide such a mechanism.

- Infrequent updates (broadcasts) need to be considered to compensate for route degradation over time. The original version of TORA leaves the possibility open but does not require it or specify any procedure.

Major differences between related functionalities in SOAP and ZRP

- The amount of information that a node has to send to its k-neighbors is larger in SOAP than in ZRP. The new fields were introduced in section 3.3. These fields provide the nodes with the information necessary to decide to create/delete reference areas.
- In SOAP, the nodes send updates not only when there is a change in their connectivity but also after a period of T seconds. T should be chosen in order to provide for infrequent updates which do not lead to an excessive traffic burden.
- IN SOAP, nodes whose status is *around* have to send the updates not only to their k-neighbors but also to their reference node.
- In SOAP, when a node receives a FAILURE packet, it sets the location table entry associated to the destination to NULL.
- In SOAP, when forwarding packets using IARP the nodes will try to avoid paths that interfere with a reference node's links. ZRP does not consider such a mechanism.

With the modifications outlined above to Mobile IP, TORA, and ZRP, the following actions have to be executed by the nodes involved in the routing of a packet sent by node i to node j in order to induce the protocol behavior presented in Section 2.

5.1 Node i (source node)

- If node j is within node i's zone then the latter forwards the packet using IARP-ZRP (no encapsulation is necessary and the D_ENCAPSULATION and S_ENCAPSULATION flags are set OFF).
- If node j is not within node i's zone, node i looks at its location table for the node j's reference node address (if any) :
  - If an entry is found. Let node A be the recorded reference node for node j. Node i encapsulates the packet putting node A's address as the IP destination address and its own reference node's address (if any) as the IP source address. Inside the payload node i will put node j and its own (node i) addresses. The D_ENCAPSULATION and S_ENCAPSULATION flags are set ON. If node i does not have a reference node, the S_ENCAPSULATION flag is set OFF and the source IP address will be equal to node i address. The packet will be forwarded across the network using the 'downstream' links established by the TORA algorithm.

- If no entry is found. Node i uses the IERP-ZRP protocol. It looks at its IERP table for the route towards node j and sends the packet to the next border node in that route. The destination IP address is equal to node j address and the D_ENCAPSULATION flag is set OFF. Also, the S_ENCAPSULATION flag and the source IP address field will be set depending on whether or not node i has a reference node as in the previous case. If there is no known route towards node j, node i bordercasts a REQUEST. The rest of the (border) nodes receiving a REQUEST react to the REQUEST as specified in the ZRP protocol.

5.2 Node k, (intermediate node)

If an intermediate node receives a REQUEST packet, it will react as specified in the ZRP protocol. Therefore, only the case where a data packet is received is considered here.

When node k receives a data packet (i.e. a packet that is not a REQUEST), it reads the S_ENCAPSULATION flag and updates its location table entry corresponding to the source node. Also, the node reads the D_ENCAPSULATION flag and recovers the destination address. Node k compares the destination address to its own. If they are equal it is implied that node k is the destination, therefore it executes the procedure explained in subsection 5.3. If node k is not the destination node, it checks the D_ENCAPSULATION flag again.

- If the D_ENCAPSULATION flag is OFF, the node tries to forward the packet using the IARP-ZRP protocol. If this is not possible the node sends a FAILURE packet to the source node.
- If the D_ENCAPSULATION flag is ON, the intermediate node recovers the actual destination address (node j in the previous example) and the presumed (maybe out-of-date) reference node's address.
  - If node j is within the intermediate node's zone. The intermediate node forwards the packet directly to node j, using the IARP of ZRP – either Link State (LS) or Distance Vector (DV) algorithm – and bypassing the reference node if possible. The packet is still D_ENCAPSULATED.
  - If node j is not within the intermediate node's zone. Node k compares the presumed reference node's address (IP destination address in the packet) with its own address.
    * If they are different, node k forwards the packet D_ENCAPSULATED along its less costly 'downstream' link.
    * If they are the same. Node k is the presumed reference node. If node k is not a reference node then it sends a FAILURE packet to the source. The other possibility is that node k is a reference node but node j is not within its zone. It may be

because the source location table is out-of-date or because the node j is *around.* Node k checks its location table for node j's reference node address looking for a more up-to-date reference node information and the node status.

- If a more up-to-date reference area is found, node k forwards the packet to the new reference node D_ENCAPSULATED using TORA.
- If node k does not have a more up-to-date reference node and node j status is *assigned* : An error occurred. Node k sends a FAILURE packet to the source.
- If node k does not have a more up-to-date reference node and node j status is *around.* Node k knows which border node is closer to node j. Node k sends the packet to node j using ZRP. The D_ENCAPSULATED flag is OFF and the accumulated route towards j consists of only two terms : the border node closer to node j, and node j itself.
- If node k does not have a more up-to-date reference node and node j status is *free.* Node k generates and sends a limited-depth bordercast (D_ENCAPSULATED OFF) trying to reach node j's current position. After receiving the location update or after a timer expires, node k sends a packet to node i updating node j's location table entry to the new found reference node address (if any), or to NULL (a FAILURE packet). Node i will use the new reference area (or NULL) value for successive packets.
- If node k does not have a more up-to-date reference node and node j status is *assigned* : An error occurred. Node k sends a FAILURE packet to the source.

5.3 node j (destination node)

When the destination (node j) receives the packet it updates its location table with node i's reference node address (if any), removes the overhead and passes the data to the upper layers.

6 Extensions

The present protocol was initially conceived to work in a horizontal highly mobile network. In the present section, the protocol behavior in a different scenarios is investigated.

6.1 Superclusters

In an ad-hoc network, the number of destinations increases with the network size. Beyond a certain point it may be too costly to keep track of all the clusters (reference nodes). Under such conditions it would be effective to group several clusters into superclusters. The nodes would then keep track of all the superclusters in the network as well as the clusters that belong in the same supercluster with that of a particular node. A destination node should be referred to by using its supercluster (reference) node's address, its cluster (reference) node's address, and finally its own address.

6.2 Two level network

In [9] a two level network has been considered. One level is formed by mobile land stations with limited coverage, and the second level (vertical network) is formed by a network of airplanes with both land-to-air and air-to-air interfaces. An airplane's land-to-air radio interface has a greater radio coverage and bandwidth and can serve to directly connect two land stations. The number of users inside an airplane radio coverage area defines the interface's footprint size. The concept of bandwidth pool is introduced to represent the bandwidth available to all the users sharing a common access medium (land-to-land interface or land-to-air interface). The cost of a link is defined to be inversely proportional to the remaining bandwidth in the bandwidth pool and directly proportional to the number of users accessing this pool, or in other words, the number of users that are deprived of accessing the bandwidth pool when a packet is transmitted. The routing protocol at each node has to decide which bandwidth pool to access when forwarding a particular packet.

In this scenario, the horizontal network (land mobiles) implementation of the SOAP protocol may enable a highly efficient method for exploiting the land-to-air interface. Assuming that the horizontal network has implemented the SOAP protocol, choosing reference nodes and forming clusters, a two level protocol may be implemented as follows:

When a source node has initially a packet to transmit (beginning of a session), it will check its location table for the destination node's reference node. The time the last entry (in the location table) was entered is also recovered. The next step is to assign a cost for the packet for any of the interfaces. For the land-to-air interface the cost is equal to the ratio between the footprint size of the air interface and the available Bandwidth in the bandwidth pool. For the land-to-land interface the packet is assigned a cost that is directly proportional to the time elapsed since the last update was entered. This cost will reflect the fact that older data may be out-of-date and may origin a long path towards the destination. The land-to-land interface cost is also proportional to the average number of neighbors a user has and inversely proportional to the network land-to-land interface available bandwidth. If no reference node is found the cost is set to infinity. This means that instead of bordercasting a REQUEST over the horizontal network, the requests are going to be carried out over the land-to-air interface. The resulting behavior in the network will be to minimize the number of bordercasting over the horizontal network. Also, long inefficient paths toward a destination are most likely eliminated, being replaced by broadcasts using the air network support (land-to-air and air-to-air interfaces). The horizontal network will tend to use only up-to-date paths, where the node is almost sure to be inside the recorded reference area.

After the beginning of the session and once the source node has up-to-date information about the destination node's reference area (if any), the source node may update the route's *accumulated* *cost*. For this purpose, each packet traveling over the horizontal network will have both an *accumulated cost* and a *remaining cost* field. The *accumulated cost* field will be initially set to zero, and each node that forwards the packet will add their link's cost computed as the ratio between the number of neighbors sharing the link over the available (remaining) link bandwidth. When the packet arrives at the destination, the *accumulated cost* field will have the current cost associated with the path followed over the horizontal network. This cost is path-dependent, so different packets following different path will collect different values of this cost. However, the observed *accumulated cost* is good enough to provide a rough estimate of the distance between the source and the destination nodes as well as the congestion state of the subset of the network between them.

When a node already in a session has a packet to send, it uses the most up-to-date *accumulated cost* information available as the land-to-land interface cost and compares it to the land-to-air interface cost to decide over which interface to forward the packet. In case the land-to-land interface's cost is lower and the node forwards the packet over the horizontal network, then the intermediate nodes in the network will use the *remaining cost* as the land-to-land interface cost for the remaining of the route toward the destination. The *remaining cost* is initialized by the source node to the value of the last known *accumulated cost*, and each node along the past subtracts its link cost from the remaining cost. It should be noted that the *remaining cost* may be negative or may reach the destination with a nonzero value. Nevertheless, it gives an idea of the approximate cost of the remaining of the route. Additionally, in networks with bidirectional links the *accumulated cost* of a source-destination path may be assumed to be equal to the value in the *accumulated cost* field of the packets received in the reverse direction (destination-source). For asymmetric network, however, the *accumulated cost* in the forward direction has to be piggybacked from the destination back to the source.

Finally, the SOAP protocol with the mentioned modifications will allow for an efficient use of the land-air interface. The horizontal network will be relieved of broadcast, bordercast, and other congestion-causing packets. Packets that need to traverse long paths will be forwarded using the land-air interface whereas packets traveling short portions of the network will be transmitted over the horizontal network.

6.3 Integrated fixed-mobile network

A mixed environment with picocells interconnected with fixed network nodes, and mobile nodes that may want to communicate with other mobiles as well as fixed network nodes is considered here. The nodes may communicate directly with each other if within transmission range. The mobile nodes' transmission range is smaller than the picocells' (due to power constraints) and they use a frequency different from that of their closest picocells, most likely the transmission frequency of another – out of reach – picocell (spatial reuse). The mobile nodes may also rely on the picocells to forward their packets. It is likely that mobile nodes are out of the picocell range and need their packet to be forwarded by another mobile node, in a multihop fashion. Such a scenario may be possible in buildings having fixed antennas in predefined positions but that do not cover all the possibles areas; as for example halls, entrances, some rooms, etc.

In this scenario, it is expected that the SOAP protocol will identify the picocells as reference nodes and will keep track of them [5]. If the mobile nodes are always k hops or less away from a picocell then the picocells will be the only reference nodes. The picocells will always have routes to each other – after the initial broadcast announcing a picocell as reference node – so they will not require to execute the TORA protocol never again (will never get without a 'downstream link'). In this scenario the protocol will present the required behavior as far as no mobile node is chosen as reference node.

But, if the mobile nodes are more than k hops away from a picocell it is likely that some of them will organize in a reference area. In such a case, a picocell would be forced to track down a mobile node using the TORA protocol resulting in eventual performance degradation as shown by the next example. Consider that a mobile node is chosen as reference node, over time it may be possible that the height level of all the mobile nodes with respect to a mobile reference node have been updated to the current time causing the mobile nodes' heights to be greater than the height of their closer picocell. In this situation a picocell would be forced to increase its own height and propagate the new height level all over the fixed network. This action is neither desired nor efficient.

The SOAP need to be revised and modified in this situations. The self-organizing algorithm discussed in Section 3 is still valid, but some alternative to TORA needs to be developed.

An ad-hoc solution may be to run a different algorithm in the picocells, which will behave as gateways as explained below. Routes between picocells and fixed network nodes will be computed in the traditional fashion, as for example, using link state updates (LSU). Mobile nodes will run the SOAP protocol described above. The picocells will run SOAP in their mobile network interface, and will keep track of the (mobile) reference nodes. If a picocell detects that a reference node is reachable[6] it may send an LSU update including a (virtual) link between the reference node and the picocell with a cost equal to the cost of the path between the picocell and the reference node[7]. Similarly, if the picocell detects a network partition that causes a reference node to be unreachable, it will send an LSU notifying the rest of the picocells that the link reference node-picocell is down. Also, when a picocell receives a broadcast from a new reference node, it sends an LSU update to

[5] This behavior may even be forced by the picocells setting their *Gain* field to the maximum value and advertising themselves as new reference nodes

[6] TORA detects if a node is unreachable (network partition) with one bit in the height field.

[7] Since the path between the picocell and the reference node is variable, so is their cost. The cost of the (virtual) reference node-picocell link should reflect the expected value of the path cost in the near future but not necessarily be directly proportional to it.

the fixed network advertising a link to the new reference node[8]. Any picocell that receives such an LSU will convert it in a broadcast similar to the one originated by the reference node (preserving the hop count)[9]. With the above modifications to SOAP, when a mobile node want to send a packet to a node that is unreachable (or too far away) in the mobile network, it would rely on the closest picocell which will use Link State (or other) protocol to forward the packet to the destination (if a fixed node), or to the closest picocell (if a mobile node). In the latter case, the picocell will forward the packet to the destination node's reference node as indicated by the SOAP protocol. In case the destination node is not associated with any reference node, then flooding will have to be used. The above protocol behavior assumes that location information is available. The actual implementation of the location management function depends on the choice between a hierarchical network (the picocells will storage the location management information for nodes that are not reachable if using only the mobile network segment) or flat network (the location management information for nodes unreachable unless using the fixed network segment, may be distributed across the network). The decision rules for such a choice need to be further investigated.

7 Conclusions and future work

In the present report, an algorithm is proposed for allowing a network to self-organize. This self-organizing algorithm together with a location management function allows for an efficient routing protocol for Mobile Ad-hoc Networks. The protocol is presented and explained and some modifications to other scenarios are also covered.

The presented protocol is expected to have a good performance in the scenarios presented. However, as pointed out before, it may not perform as well in different scenarios as for example in a fixed-mobile network. Some extensions need to be included to allow the protocol to perform efficiently under a broad class of scenarios. The self-organizing property need to be further explored to achieve a protocol that really adapts to every network condition, from quasi-static (when route optimization is an issue) to highly mobile (mobility management is the main issue).

References

[1] C. Perkins, editor. *IP Mobility Support.* RFC 2002, October 1996.

[8] Additionally, the picocell has to transmit, in the same or other packet, the location information of the nodes associated with the reference node. Both packets (LSU and location update) packets should not leave the picocell network domain.

[9] Alternatively, the picocells may form a higher level hierarchy recording the reference node info without passing it to its neighboring mobile nodes. In such a hierarchical network, when a mobile source node needs to send a packet to a node that is not reachable or for which no location information is available, the source node will forward the packets the closest picocell, which will behave as its default gateway.

[2] A. Myles, D. B. Johnson and C. Perkins, "*A Mobile Host Protocol Supporting Route Optimization and Authentication.*", IEEE Journal of Selected Areas in Communications, 13(5) pp 839-849, June 1995.

[3] Internet Engineering Task Force (IETF), Mobile Ad Hoc Networking (MANET) Working Group drafts in http://info.internet.isi.edu:80/R284713-291510-1m/in-drafts/id-abstracts.html

[4] V. D. Park, and S. Corson, "*A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks.*", IEEE Infocom '97. Kobe, Japan(1997).

[5] Z. J. Haas, "*The Zone Routing Protocol (ZRP) for Ad Hoc Networks.*" Internet Draft, IETF Mobile Ad Hoc Networking Working Group, November 1997. Work in progress.

[6] V. D. Park, and S. Corson, "*A Performance Comparison of the Temporally-Ordered Routing Algorithm and Ideal Link-State Routing.*", IEEE Infocom '98.

[7] C. Perkins, "*Ad Hoc On Demand Distance Vector (AODV) Routing*". Internet Draft, IETF Mobile Ad Hoc Networking Working Group, 20 November 1997. Work in progress.

[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] I. Stavrakakis and R. Landry, "*Management of Communication Resources in a Hierarchical Mobile Mesh Network.*" Technical Report TR-CDSP-98-49, Communication and Digital Signal Processing Center, ECE Deparment, Northeastern University. Boston, MA. 1998.