

# Technical Report

# Area Avoidance Routing in Distance-Vector Networks

H.Zlatokrilov                      H.Levy  
[zlatokri@post.tau.ac.il](mailto:zlatokri@post.tau.ac.il)      [hanoch@cs.tau.ac.il](mailto:hanoch@cs.tau.ac.il)  
Tel-Aviv University  
Israel

## Abstract

Network routing may be required, under certain applications, to avoid certain areas (or nodes) These areas can be of potential security threat, possess poor quality or have other undesired characteristics. Thus, protocols that can perform area avoidance routing can be beneficial for many objectives. Such routing is particularly challenging in distance-vector networks, where only the shortest-distance information is available to the nodes. We address this challenge by algorithms that retrieve distance-vector information from other nodes, we refer to as reference nodes, and exploit it for computing guaranteed area-avoiding paths. Having these paths, the source can direct the packets using loose source routing towards the destination.

We first study the conditions that guarantee area avoidance routing for the scenario where routing can be done only through the reference nodes. Then we extend the problem to general area avoidance routing with no restrictions. For both scenarios we propose deterministic algorithms and prove their correctness. Using the routing conditions, we study the problem of dynamically selecting reference nodes (nodes from which the source retrieve distance vector information) that guarantee area avoidance routing. We propose several algorithms and analyze their performance over several structured graphs (e.g. Grid) and evaluate the performance on general networks via extensive simulations. We show that in many cases small amount of reference nodes are required for area avoidance routing.

## 1 Introduction

We consider network settings in which some nodes or network areas are likely to jeopardize packets traversing them. For example, consider nodes that are suspected to be malicious or are of poor quality (e.g. high packet drop) or even, in the case of radio ad-hoc networks, have poor reception and/or energetic qualities. We believe that such information may be provided by network monitoring and analysis utilities or by any other intelligence sources. Under such circumstances it is of high interest to route packets on paths that avoid these obstacles, namely *area avoiding* paths, for enhancing security, improving QoS and other objectives.

For the sake of clarity, throughout the paper we will focus on the setting where avoidance area are nodes suspected to be malicious and the objective is security enhancement. However, one should note that the paper is not focusing solely on security issues but lays a general architecture and mechanisms for many objectives where avoidance area can be of various characteristics. The model and the techniques addressed in this work may be beneficial for any routing technology using distance vector including static and ad hoc networks.

We consider a distributed system that is not controlled by a single entity and hence the avoidance area in the network cannot be eliminated and routing protocols cannot be changed. However, we assume that nodes allow other nodes to query their distance vector information. Using this information the source is required to direct packets along guaranteed *area avoiding* paths. Using terminology from security we will refer to such paths as *safe*. This technique complies with a recent trend in routing research to allow hosts to select routes by themselves to avoid the inefficiencies caused by the shortest path network-layer routing.

The concept of area avoidance routing was introduced in [30]. In that study shortest path and area avoidance routing were addressed by a technique of *virtual-position source-routing*<sup>1</sup>. That technique uses the distances of nodes from each other as a system of coordinates for routing. The basic assumption there was that the forwarding algorithm can be modified and each node follows the forwarding rules. In this paper we address a similar problem where we relax this assumption and rely only on the ability to query nodes for their distance vector information. Though the objectives are similar the techniques introduced in this paper take a different course towards area avoidance routing. In a sense, this paper offers a more practical method since it does not necessarily require the cooperation of all nodes in the network<sup>2</sup> and does not require the modification of the forwarding algorithms in the entire network.

Routing algorithms can be roughly classified into link-state and distance-vector algorithms. In link-state routing algorithms, such as OSPF, nodes obtain a snapshot of the state of the network. Using this snapshot and the intelligence information about avoidance area, the source can use a centralized algorithm to calculate area avoiding paths. The source can then direct the packets using strict source routing. Centralized algorithms for calculating area avoiding paths are relatively simple (e.g. remove the nodes to be avoided from the graph and compute the shortest path) and do not pose a major difficulty. Thus, in link-state routing protocols, the network layer provides all the necessary mechanisms for carrying out area avoidance routing policies set by the sender.

Distance-vector algorithms are distributed implementations of all-pairs shortest path algorithms, based on the Bellman-Ford algorithm. Protocols such as the Routing Information Protocol (RIP) and the Destination-Sequenced Distance-Vector (DSDV) use the property that for every pair of nodes  $s$  and  $t$  and an intermediate node  $v$  (along the

---

<sup>1</sup> For the sake of simplicity we will assume that there is no limit on the number of nodes in the list of source routing. Practically, due to technical reasons, in IP the number of nodes in the source routing list is limited.

<sup>2</sup> In the analysis we assume that all nodes will reply to queries and answer them correctly.

shortest path from  $s$  to  $t$ ), the shortest path between  $s$  to  $t$  is the concatenation of the shortest paths between  $s$  and  $v$ , and  $v$  and  $t$ . This key property lends these algorithms a simple and elegant distributed implementation in which every node exchanges with its adjacent nodes the shortest path to all other nodes. While this property is sufficient for shortest path routing, it poses challenges for routing through other paths.

In particular, as we showed in [30], no algorithm that is based only on the distance-vector information can guarantee routing through area avoiding paths, even if such paths exist, for any *arbitrary network*. In fact, we showed that even if such information exists for all nodes at distance shorter than  $k$ , a node-avoiding path cannot be guaranteed to be found. Nonetheless, networks are typically highly connected, which allows the design of effective area avoiding algorithms for distance vector networks.

Our focus in this work is on exploring mechanisms for guaranteed area avoidance (*safe*) routing in distance vector networks. The mechanism we study consists of having the source node query other nodes (called *reference nodes*) for their distance vector information. We study the conditions that, based on this information, can guarantee area avoidance routing between pairs of nodes. Based on these conditions we propose algorithms for computing area-avoiding paths. These paths are characterized by a sequence of nodes  $(v_1, \dots, v_k)$  such that the shortest path between every consecutive pair  $(v_i, v_{i+1})$  is safe. This set of nodes is then used for loose source routing.

Our work starts (Section 2) in introducing the model and terminology. We show that there is no deterministic algorithm that can guarantee finding an area avoiding path without querying  $|V|/2-2$  nodes (where  $|V|$  is the number of nodes in the network).. Clearly, querying this number of nodes is impractical. In Section 3, we study the basic model in which the source retrieves distance vector information from a designated set of *reference nodes* and it may direct packets only through these nodes (the reference nodes are the only source-routing relay nodes), yielding what we denote *reference-node paths*. We derive conditions for area avoidance routing between pairs of reference nodes and provide an algorithm for computing area-avoiding path from source to destination. The results of this section are extended in Section 04 where the model allows routing through *any set* of nodes. We show that though the number of candidate paths, under this setting is enormously high, the search for area avoiding paths can be limited to a small set of paths which are generalizations of the reference-node paths. This highly reduces the complexity of path searching under this setting. Next, in section 5, we address the problem of dynamically selecting reference nodes in a way that will increase the likelihood of finding area-avoiding path with minimal number of queries. We study several selection algorithms and analyze their performance on structured graphs (e.g. grid). In addition, we evaluate these algorithms on general graph settings by investigating a large number of graphs via simulation. The results show that the algorithms are very useful in finding such paths and that on practical graphs it is likely to find good area avoiding paths. In Section 6 we address a centralized problem in which the objective is to find a minimal set of reference nodes so that a safe path will be found (using the algorithms for finding safe paths studied in previous sections) between every pair of nodes if such safe path exists. We show that this problem is NP-Complete and provide a

greedy algorithm addressing this problem .Lastly, concluding remarks are given in Section 07.

## 1.1 Related work

In this study we address the problem of area avoidance routing. The benefits of such routing can be derived from many studies in the field of non-shortest path routing such as load-balancing and efficiency [9][25], reliability [7][11][18] and QoS enhancement [15] [19][24]. In addition, various security objectives such as DoS etc. were addressed by multipath routing in many studies such as [2][14][21][27].

Centralized approach for computing paths other than the shortest path with various objectives was studied in [2][19] and [26]. Similar ideas can be applied for finding area-avoiding paths subject to various objectives. In distance vector networks, a random approach for selecting intermediate nodes together with source routing was introduced in [1][3][9]. A distributed distance vector algorithm for multipath routing that modifies the topology exchange mechanism was studied in [8] and [23]. A general routing methodology that uses distances from several nodes in the network for routing in distance vector networks was introduced in [30]. One of the applications of that methodology was area avoidance routing. In this study we address similar objective by a different method that leaves the forwarding algorithm intact and uses only standard network services (i.e. source routing and distance vector information available in the routing table). This property lends the algorithms in this work chances for practical implementation in various networks.

The algorithms presented in this paper are applicable in almost any network in which nodes possess only their shortest distances to other nodes (distance vector algorithms such as RIP) and do not construct a “map” of the network (i.e. in link state protocols such as OSPF protocol). For example, ad-hoc networks using distance vector protocols such as AODV [20] or networks using geographical distances in the field of geographical routing (e.g. [5][17]) may take advantage of these algorithms. The algorithms can also be used for area avoidance in peer-to-peer networks where distances, in terms of latency or bandwidth, are measured for routing enhancing [10].

## 2 Model and Assumptions

We model the network as an undirected connected graph  $G(V,E)$ . The nodes and the edges are denoted  $v_1, \dots, v_{|V|}$  and  $e_1, \dots, e_{|E|}$ , respectively. Each edge  $e$  is associated with a length parameter  $c_e > 0$ . We assume that the triangular inequality holds in the network<sup>3</sup>.

The routing algorithm in the network is distance-vector based and the topology exchange algorithm and forwarding mechanism cannot be changed. That is, each node  $v_i$  contains in its routing table the distances to all destinations, denoted by *distance vector* of  $v_i$ ,  $T(v_i)$ . Namely,  $T(v_i) = \{d(v_i, v_1), d(v_i, v_2), \dots, d(v_i, v_{|V|})\}$  where  $d(v_i, v_j)$  denotes the shortest distance from node  $v_i$  to  $v_j$ ; due to the symmetry of the graph  $d(v_i, v_j) = d(v_j, v_i)$ . Typically, for

---

<sup>3</sup> In the Internet, though the BGP is based on distance vector, the triangular inequality is not always preserved and the routing, in many cases, is policy based rather than shortest path oriented.

shortest path routing,  $v_i$  uses this information to construct a routing table containing the shortest path and the next hop to each destination.

Path  $p$  (also referred to as route) is a sequence of nodes  $(v_1, \dots, v_k)$  and the links between them  $(e_1, \dots, e_{k-1})$ . If  $v_i (e_j)$  is in path  $p$  we say that  $v_i \in p (e_j \in p)$ . The length of path  $p$ ,  $L(p)$ , is measured as the sum of link weights, that is  $L(p) = \sum_{e \in p} c_e$ . Throughout the paper we will use  $s$  to denote the source node and  $t$  to denote the destination node. The notation  $v_i \text{---} v_j$  is used to denote a shortest path between  $v_i$  and  $v_j$  (there might be several such paths and  $v_i \text{---} v_j$  can be any one of them).

We assume that the network supports source routing. In this technique the source may dictate the full path (strict source routing), or provide a list of intermediate (relay) nodes the packet should traverse through (loose source routing) on its way to the destination. Shortest path routing is implemented between pairs of consecutive intermediate nodes. In this paper we focus on the problem of calculating the list of intermediate nodes such that area avoidance routing is guaranteed. For the purpose of this study we assume that the list of relay nodes in the source routing header is unlimited (though technically the list of nodes in the IP header is limited).

In the routing techniques discussed in this paper, the source queries other nodes for their distance vector information and aims at directing packets over area avoiding paths<sup>4</sup>. The paths can be computed per packet (resulting in high query cost) per session, periodically or according to any other policy. For the sake of simplicity we will assume that between queries the distance vector information in the network remains static (in the worst-case packets transmitted between queries might be jeopardize). One possible method to enhance such mechanism would be to ask for updates on specific routing information in the cooperating nodes as they occur. We also assume that topology changes are reflected immediately and simultaneously in all nodes.

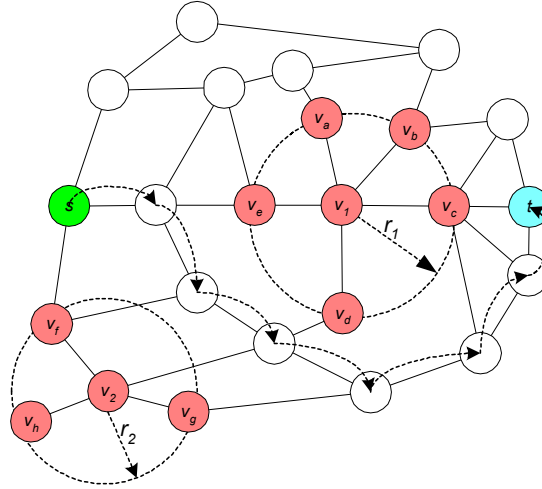
The formal definition of the *area avoidance* routing (also referred to as *safe routing*<sup>5</sup>) is as follows: Let  $v_i$  be a node (to be denoted the *area center node*) and let  $AA(v_i, r) = \{v_j \in V \mid d(i, j) \leq r\}$  denote the set of nodes at distance smaller than or equal to  $r$  from  $v_i$  (the *avoidance area*). The objective is to route packets via *area avoiding (safe) path*  $p$  such that  $p \cap AA = \emptyset$ . The distance  $r$  can be counted in hop-count, in which case  $c_e=1$  for all edges, or using the distance metric, in which case the link lengths,  $c_e$ , are used for distances. In the case  $r=0$ , the objective is to avoid passing through  $v_i$  itself.

This problem can be extended to deal with several avoidance areas,  $AA_i$ ,  $0 < i \leq K$ . In that case let  $U = \{v_1, v_2, \dots, v_k\}$  denote a set of area center nodes and  $R = \{r_1, r_2, \dots, r_k\}$  denote the set of the corresponding distances from the center nodes. The set of all avoidance areas  $AA(U, R) = AA_1 \cup \dots \cup AA_K$  denotes the set of avoidance areas, we will use the term  $AA$  to denote this set.  $AA_i$  is called *avoidance area i*.

<sup>4</sup> Practically, in distance vector algorithms, nodes receive the routing information from their adjacent nodes. In our model we assume that an implicit query is required for receiving distance vector information from adjacent nodes.

<sup>5</sup> Since our focus is on security objectives, we will use the terms *safe path* and *safe routing* to denote area-avoiding path and area avoidance routing.

An illustration of area avoiding path is depicted in Figure 1 where node  $v_1$  and  $v_2$  are the area center nodes and the avoidance area contains nodes in distance  $r_1=r_2=1$  from  $v_1$  and  $v_2$  respectively. That is  $AA_1=AA(v_1,r_1)=\{v_a,v_b,v_c,v_d,v_e\}$  and  $AA_2=AA(v_2,r_2)=\{v_f,v_g,v_h\}$ . An area-avoiding path is plotted by the dotted line.



**Figure 1** Area avoiding path.

In [30] we showed that there is no deterministic memoryless distance-vector forwarding algorithm that can guarantee, for every *arbitrary network*, routing over a path that avoids a given node  $v_i$ , even if such path exists. This stands even if the source uses the distance vector information of all neighbors in distance  $k$ . Thus, heuristic algorithms are to be used for finding area avoidance paths.

In the following sections, we propose and analyze several algorithms for area avoidance routing. All the algorithms share the property of querying nodes for their distance vector information and using it for finding area-avoiding paths. We will refer to the nodes queried by the source as *reference nodes*. We assume that reference nodes can and will provide all their shortest distance information ( $T(v_i)$ ) when being queried.

**Proposition 1:** There is no deterministic algorithm that can guarantee finding an area-avoidance path, by querying less than  $|V|/2-2$  reference nodes, even if such path exists.

**Proof:** To prove this property it is enough to show that there is a network where the source must query at least  $|V|/2-2$  reference nodes to guarantee that a path is area avoiding. To this end consider the network in Figure 2 where the area avoidance is  $AA(v_{|V|},r=0)$ . The source,  $v_1$ , must query at least  $|V|/2-2$  nodes (every other node from  $v_1$  to  $v_{|V|-1}$ ) for identifying the existence of the only area avoiding path ( $v_2,v_3,v_4,\dots,v_{|V-2},v_{|V-1}$ ) to  $v_{|V|-1}$ . In case the source queries less than  $|V|/2-2$  nodes, there are two nodes,  $v_{i+1}$  and  $v_{i+2}$ , that the source did not query. Now, one can verify that the shortest distance between every pair of nodes on the graph (except between  $v_{i+1}$  and  $v_{i+2}$ ) is not affected by the existence of the edge ( $v_{i+1}, v_{i+2}$ ), and thus the question whether this edge exists (and a safe path exists) cannot be derived unless at least one of these two distance vectors is queried. Altogether, the source  $v_1$ , must query every other node ( $v_3,v_5,\dots,v_{|V|-1}$ ) (there is

no need to query  $v_{|V|}$ , and  $v_l$  has its own distance vector). Altogether, at least  $|V|/2-2$  nodes must be queried.

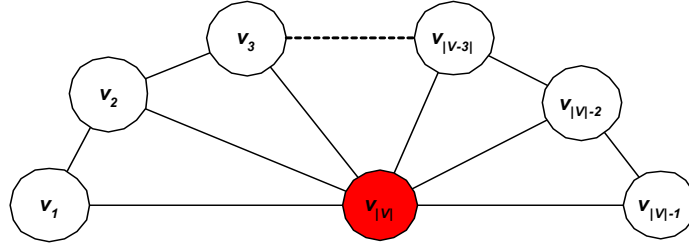


Figure 2 A network that requires the distance vector information from at least  $|V|/2-2$  nodes.

### 3 Hub Based Area Avoidance Routing

In this section we address the basic problem in which the network contains a set  $RN=\{v_1, v_2, \dots, v_k\}$  of designated reference nodes and the source retrieves the distance vector information only from this set. That is,  $s$  has  $T(s)$ ,  $T(v_1)$ ,  $T(v_2), \dots, T(v_k)$  and  $T(t)$ . We assume that  $s$  may direct packets using loose source routing only through nodes in  $RN$ . In this case the reference nodes become also the only possible *hubs* (relays). In the following section we will extend this model and remove the latter restriction, but for the sake of clarity and understanding of the problem we start with this basic model.

We start by deriving a set of conditions for area avoidance routing between a pair of references  $v_i$  and  $v_j$ . Using these conditions we provide an algorithm that finds the set of hubs such that using them in loose source routing (and forwarding the packet from one hub to the next one via a shortest path) guarantees area avoiding path between  $s$  and  $t$ . In case that the algorithm does not find a path, it means that from the available distance vector information the source cannot guarantee area-avoiding path. However, it does not imply that no such path exists. To enhance this, we offer a heuristic approach based on the conditions for area avoidance routing.

In what follows we consider two arbitrary reference nodes  $v_i$ ,  $v_j$  and find conditions for guaranteeing that  $v_i-v_j$  avoids area  $AA_k=AA(v_k, r_k)$ . These conditions are the basis for finding area-avoiding paths. In the conditions we use the following trivial observation:

**Observation 1:** Node  $v_q$  is on a  $v_i-v_j$  if and only if the following trivial condition holds:

$$d(v_i, v_j) = d(v_i, v_q) + d(v_j, v_q). \quad (1)$$

**Claim 1:** If the following holds then every node  $v_q$  on a  $v_i-v_j$  obeys  $v_q \notin AA_k$ .

$$d(v_i, v_j) < d(v_i, v_k) - r_k + d(v_j, v_k) - r_k = d(v_i, v_k) + d(v_j, v_k) - 2r_k. \quad (2)$$

**Proof:** By a way of contradiction assume that (2) holds and there exists node  $v_q$  on  $v_i-v_j$  and  $v_q \in AA_k$ . Thus, using the triangular inequality and  $d(v_q, v_k) \leq r_k$  we have (see Figure 3):

$$i) \quad d(v_i, v_q) \geq d(v_i, v_k) - r_k$$

$$ii) \quad d(v_j, v_q) \geq d(v_j, v_k) - r_k$$

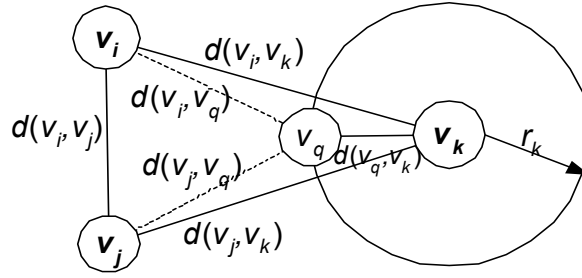


Figure 3 Condition for area avoidance routing.

Since  $v_q$  is on  $v_i-v_j$ :  $d(v_i, v_j) = d(v_i, v_q) + d(v_j, v_q)$  leading to  $d(v_i, v_j) \geq d(v_i, v_k) + d(v_j, v_k) - 2r_k$  and thus to a contradiction with (2). ■

**Observation 2:** Note that if (2) holds:  $d(v_i, v_k) > r_k$  and  $d(v_j, v_k) > r_k$ , namely,  $v_i$  and  $v_j$ , are not in  $AA_k$ . This observation comes immediately from putting the triangular inequality,  $d(v_j, v_k) \leq d(v_i, v_j) + d(v_i, v_k)$ , into (2) (we have:  $d(v_i, v_j) < d(v_i, v_k) + d(v_i, v_j) + d(v_i, v_k) - 2r_k$  that leads to  $d(v_i, v_k) > r_k$ ). Also, if (2) holds  $d(v_i, v_j) < d(v_i, v_k) - r_k$  or  $d(v_i, v_j) < d(v_i, v_k) - r_k$ .

**Corollary 1:** If Claim 1 holds for all  $AA_k$  ( $\forall k: 1..K$ ) then a packet routed from reference node  $v_i$  to reference node  $v_j$  will traverse a safe path.

Corollary 1 follows immediately from the definition of AA.

Next we present additional method for finding a safe path between two reference nodes. This is based on the property that if all nodes on the shortest path are ruled out of being in  $AA_k$ , the shortest path between the reference nodes is safe.

Using the following claim any reference node,  $v_r$ , can *testify* using its own distance vector, that node  $v_q$  can be concluded not to be in  $AA_k$ :

**Claim 2:** For node  $v_q$ , if there exists a reference node  $v_r$  for which:

$$d(v_r, v_q) < d(v_r, v_k) - r_k \quad \text{OR} \quad (3)$$

$$d(v_r, v_q) > d(v_r, v_k) + r_k, \quad (4)$$

then it is guaranteed that  $v_q \notin AA_k$ .

The proof of this claim is trivial.

If Claim 2 holds we will say that  $v_q$  is an  $AA_k$  safe node ( $v_q \notin AA_k$ ) by the testimony of  $v_r$ .

Let  $S_k^r$  denote the set of  $AA_k$  safe nodes from the testimony of  $v_r$ . We define the set of *safe nodes with respect to  $AA_k$*  as  $S_k = S_k^1 \cup S_k^2 \cup \dots \cup S_k^N$ , where  $N$  is the number of reference nodes. We define the set of *safe nodes* (with respect to the entire network) as  $S = S_1 \cap S_2 \cap \dots \cap S_K$ . Note that there may be many other nodes that are in neither of the  $AA$

areas but from the available distance vector information only the nodes in  $S$  are concluded (and denoted) to be safe nodes.

**Corollary 2:** If for every node  $v_q$  on  $v_i \rightarrow v_j$ ,  $v_q \in S$  then a packet routed from  $v_i$  to  $v_j$  will traverse a safe path (The Corollary follows immediately from the definition of  $S$  and  $AA$ ).

For illustration, consider Figure 4, where  $s$ ,  $v_1$  and  $v_2$  are reference nodes and  $v_k$  is the area center node of  $AA_k$  (with  $r_k=1$ ). In (a) we illustrate  $S_k^s$ ,  $S_k^1$  and  $S_k^2$  respectively (above each node we marked the references for which **Error! Reference source not found.** holds). In (b), the safe nodes,  $S_k = S_k^s \cup S_k^1 \cup S_k^2$ , are marked in white. Using this information with the set of nodes on the shortest path (found by **Error! Reference source not found.**), it is clear that  $s \rightarrow v_2$  is safe. Note that there can be two shortest paths for  $s \rightarrow v_2$  and both are safe.

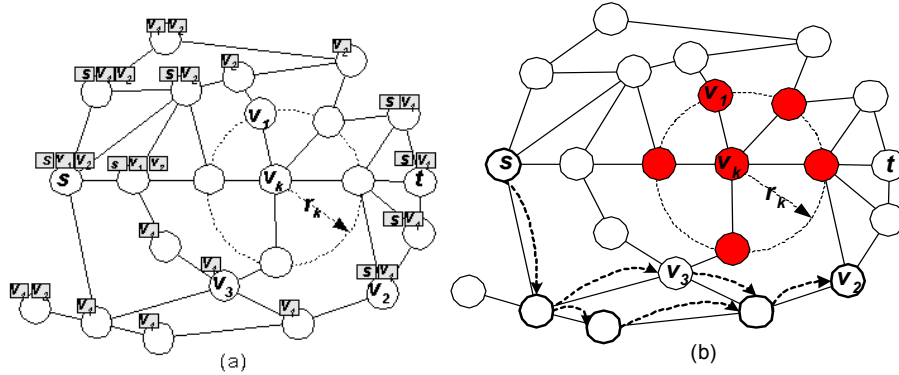
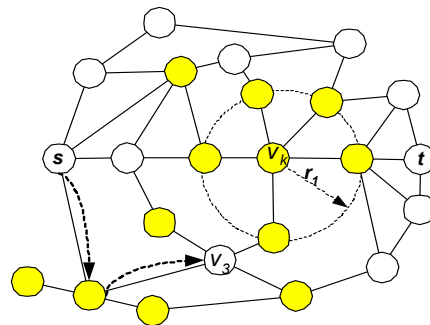


Figure 4 (a) illustrates the nodes in  $S_k^s$ ,  $S_k^1$  and  $S_k^2$ . In (b), nodes in  $AA$  are marked in red and nodes on the safe path  $s \rightarrow v_2$  appear in bold.



Corollary 1 and Corollary 2 are complimentary to each other. For example, consider the network in Figure 4. In case that the references are  $s$ ,  $v_1$  and  $v_2$ , only from Corollary 2 we infer that the path  $s \rightarrow v_2$  is avoiding  $AA_k$ . One cannot infer from Corollary 1 that the path  $s \rightarrow v_2$  is safe. On the other hand, in case that the references are  $s$  and  $v_3$ , as illustrated in Figure 5 (the nodes in  $S_k = S_k^s \cup S_k^3$  are marked in white), we can infer that the path  $s \rightarrow v_3$  is avoiding  $AA_k$  only from Corollary 1.



**Figure 5 Routing from  $s$  to  $v_3$  (also the reference nodes) is safe only due to Corollary 1. The nodes in**

$$S_k = S_k^s \cup S_k^3 \text{ are marked in white.}$$

Based on Corollary 1 and Corollary 2 the following algorithm for dictating area-avoiding path through a given set of reference nodes,  $RN$ , using loose source routing can be used:

#### ***HubBasedAreaAvoidanceRouting***

1. Construct  $G'(V',E')$ , where  $V'=\{s,t\}$  and  $E'=\{\phi\}$
2.  $\forall v_i : v_i \in RN$  add  $v_i$  to  $V'$
3. For every pair of reference nodes  $v_i$  and  $v_j$  in  $V'$ 
  - a. If Corollary 1 or Corollary 2 holds<sup>6</sup> for  $v_i$  and  $v_j$ 
    - i.  $E' \leftarrow E' \cup \{e(v_i, v_j)\}$ ,  $c_e = d(v_i, v_j)$
4. Find shortest path,  $SP$ , between  $s$  and  $t$  in  $G'(V',E')$
5. Direct the packet via the set of nodes in  $SP$  using loose source routing

Note that in the case that  $s$  and  $t$  are reference nodes and the shortest path between them is area avoiding, then this path will be used.

**Proof of correctness and complexity analysis:** Graph  $G'(V',E')$  contains only  $s$ ,  $t$  and the reference nodes that are area avoiding. A link connecting nodes in  $G'$  is added in step 3.a.i only if  $v_i-v_j$  is avoiding  $AA$  according to Corollary 1 or Corollary 2. Thus, routing over such link guarantees area avoidance routing in  $G$ . Since every edge in  $E'$  is safe, so is the concatenation of links to the shortest path in Step 4. Clearly, the algorithm provides the shortest area-avoidance path through the hubs.

The complexity is as follows: Validating Corollary 1 for all pairs over all avoidance area requires  $O(K|RN|^2)$  operations, where  $|RN|$  is the number of reference nodes and  $K$  is the number of avoidance areas. For validating Corollary 2 we have: *i.* Finding the nodes in  $S$  requires  $O(K|RN||V|)$ . *ii.* Finding the set of nodes on the shortest path between pair of references requires  $O(|V|)$ . *iii.* For every pair of references, computing whether all nodes on the shortest path are in  $S$  requires  $O(|V|\log(|V|))$  (sort the sets and go over the ordered sets). Altogether, for checking Corollary 2 for all pairs of reference nodes over all avoidance areas requires  $O(|RN|^2|V|\log(|V|))$ . In the last step (4) we calculate the shortest path over  $|RN|$  nodes in  $G'$  is  $O(|RN|\log(|RN|))$ . The overall complexity of the algorithm is then:  $O(|RN|^2|V|\log(|V|))$ . ■

### **3.1 Heuristic Approach Based on the Claims for Area Avoiding Paths**

In case that the *HubBasedAreaAvoidanceRouting* algorithm comes with an answer it is guaranteed that the path found is the shortest area avoiding path using the intermediate hub nodes. Otherwise, an area-avoiding path that uses these hubs might still exist, but it is not guaranteed. Next we briefly sketch a generalization of the algorithm, that deals with this situation, and whose details are left for further study.

<sup>6</sup> For checking Claim 4 the algorithm has to find the nodes in  $S$ .

The idea is to use a heuristic approach that tries to find area-avoiding path with high attendance. It is done by assigning lengths (grades) for paths between reference nodes in  $G$  which translated to links in  $G'$ . The lengths reflect the *relative potential* of the path to share nodes in  $AA$ . Area avoiding paths in  $G$  are translated to links in  $G'$  having length  $c_e=1$ . All other links between reference nodes will be added to  $G'$  with length  $c_e$  that is proportional to the relative potential of sharing nodes in  $AA$ . These values are set in step 3.a.i in the algorithm.

The estimation of the relative potential for links in  $G'$  to share nodes in  $AA$  relatively to other links in  $G'$  is left for further study. One possible direction is to evaluate the potential on some well-known graphs, such as grid, and use it as an estimation in the construction of  $G'$ . For example, when checking Corollary 1, if  $d(v_i, v_j) - X = d(v_i, v_k) + d(v_j, v_k) - 2r_k$ , (note that if the path between  $v_i$  and  $v_j$  is area avoiding  $X < 0$ ), the length of the link between  $v_i$  and  $v_j$  will be a function of  $X$ . The intuition is that with the growth in  $X$  the likelihood for the path between  $v_i$  and  $v_j$  to share nodes with  $AA$  increases. Analytic analysis and simulations on network models can be used for finding the suitable function that ties between  $X$  and the growth in likelihood. Similar technique can be applied when checking Corollary 2.

Such algorithm does not take into account the path actual path lengths in  $G$  the packet will traverse. However, one can use the two metrics of both relative potential and link length in parallel and compute the shortest path with highest area avoiding likelihood using shortest path with constrains algorithms such as [19][26].

## 4 General Reference-Based Area Avoidance Routing

In the previous section only reference nodes were permitted to be intermediate nodes in the loose source routing. In this section we extend the model and allow all nodes to serve as hubs. This model allows greater routing flexibility that may result in finding area-avoiding paths that were not allowed in the previous model. It can also result in finding shorter paths. However, the complexity of the resulting algorithm is higher.

Similarly to the previous model the source retrieves the distance vector information only from a set reference nodes  $RN = \{v_1, v_2, \dots, v_k\}$ . In that model the algorithm used the exact distances between reference nodes, taken from the distance vector information of these nodes,  $T(v_i)$ . However, this distance vector information hides additional conditions for safe routing between pairs of arbitrary nodes (not necessarily reference nodes). We start by deriving conditions that guarantee area avoidance routing between any pair of nodes. Then, using these conditions we describe two algorithms for area avoidance routing and analyze their performance.

The first algorithm is a greedy algorithm exhaustively checks all possible combinations for finding safe paths between pairs of nodes. By doing so, the algorithm guarantees that if an area-avoiding path between  $s$  and  $t$  can be found with the given distance vector information, the algorithm will find it. However, the complexity of the greedy algorithm is extremely high. The second restricted algorithm offers significantly lower complexity by accounting for a smaller set of paths that are generalizations of the reference-node paths studied in the previous section. We show that if the greedy algorithm finds a safe path the restricted algorithm will be able to find the same path. This is a significant result since it offers a low complexity algorithm having the same qualities as the greedy

exhaustive algorithm. However, the restricted algorithm may result in potentially longer paths.

**Observation 3:** The distance between  $v_i$  and  $v_j$  is bounded by:

$$\max_{v_q \in RN} (|d(v_i, v_q) - d(v_j, v_q)|) \leq d(v_i, v_j) \leq \min_{v_w \in RN} (d(v_i, v_w) + d(v_j, v_w)) \quad (5)$$

Equation (5) results from the triangular inequality. Note that (5) is valid for any  $v_i, v_j, v_q$ , and  $v_w$  in  $V$ , but since there is no knowledge about the distances between arbitrary nodes, Observation 3 should be used only for  $\forall v_q, v_w : v_q, v_w \in RN$ . For example, in the network

illustrated in Figure 6 one can estimate the distance between  $v_i$  and  $v_j$ , using (5):

$$7 \leq d(v_i, v_j) \leq 9$$

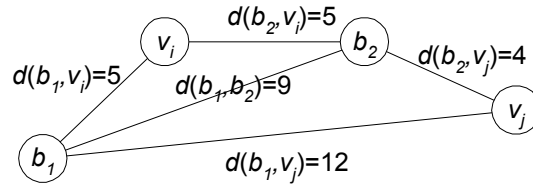


Figure 6 Estimation of minimal and maximal distance between nodes.

We will use  $\lceil d(v_i, v_j) \rceil$  and  $\lfloor d(v_i, v_j) \rfloor$  to denote the upper and lower bounds of the distances between  $v_i$  and  $v_j$ . That is:  $\lceil d(v_i, v_j) \rceil = \min_{v_w \in RN} (d(v_i, v_w) + d(v_j, v_w))$  and  $\lfloor d(v_i, v_j) \rfloor = \max_{v_q \in RN} (|d(v_i, v_q) - d(v_j, v_q)|)$ . We will use the term *witness* for a reference node,  $v_w$ , that *testifies* for the upper or lower bounds on the distances between two nodes. That is,  $v_w$  is a witness for  $\lceil d(v_i, v_j) \rceil$  if  $\lceil d(v_i, v_j) \rceil = d(v_i, v_w) + d(v_j, v_w)$  and a witness for  $\lfloor d(v_i, v_j) \rfloor$  if  $\lfloor d(v_i, v_j) \rfloor = |d(v_i, v_w) - d(v_j, v_w)|$ .

Using the upper and lower bounds of the distances between nodes we can extend Claim 1. In what follows we consider two arbitrary nodes  $v_i, v_j$  and find conditions for guaranteeing that the shortest path between them avoids area  $AA_k$ .

**Claim 3:** If one of the following conditions holds

$$\lceil d(v_i, v_j) \rceil < \lfloor d(v_i, v_k) \rfloor - r_k, \quad (6)$$

$$\lceil d(v_i, v_j) \rceil < \lfloor d(v_j, v_k) \rfloor - r_k, \quad (7)$$

$$\lceil d(v_i, v_j) \rceil < \lfloor d(v_i, v_k) \rfloor + \lfloor d(v_j, v_k) \rfloor - 2r_k, \quad (8)$$

then every node  $v_q$  on a  $v_i$ — $v_j$  obeys  $v_q \notin AA_k$ .

**Proof:** If (6) (or (7)) holds, then for every node  $v_q$  on  $v_i$ — $v_j$ :  $d(v_i, v_q) < \lceil d(v_i, v_j) \rceil < \lfloor d(v_i, v_k) \rfloor - r_k < d(v_i, v_k) - r_k$ , where the last inequality is due to Observation

3 (or  $d(v_j, v_q) < d(v_j, v_k) - r_k$ ). Thus, according to the definition of  $AA_k$ ,  $v_q$  is area avoiding. For example, in Figure 7 routing between  $v_3$  and  $v_4$  is safe due to condition (6) where  $v_{r1}$  is the witness for  $\lceil d(v_3, v_4) \rceil$  and  $\lfloor d(v_3, v_k) \rfloor$  ( $v_k$  is the area center node)..

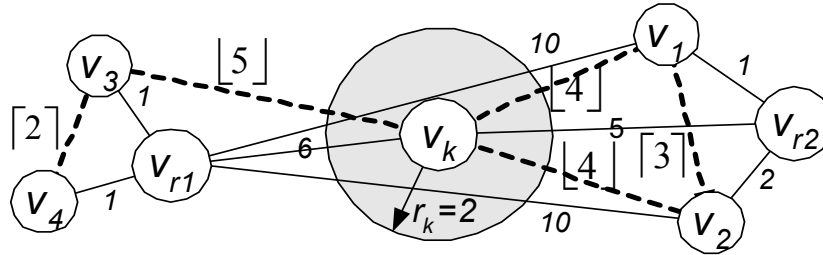


Figure 7 Conditions area avoidance routing from  $v_i$  to  $v_j$ .

If condition (8) holds, according to Observation 3 we have  $d(v_i, v_q) < d(v_i, v_k) + d(v_j, v_k) - 2r_k$  which leads to the conclusion that  $v_q \notin AA_k$  based on Claim 1. For example, in Figure 7 routing between  $v_1$  and  $v_2$  is safe due to condition (8) where  $v_{r1}$  is the witness for  $\lfloor d(v_1, v_k) \rfloor$  and  $\lfloor d(v_2, v_k) \rfloor$  and  $v_{r2}$  is the witness for  $\lceil d(v_1, v_2) \rceil$  ■

One should note that conditions (6), (7) and (8) may hold independently. In the particular case where  $v_i = v_j$ , all three conditions reduce to:  $\lfloor d(v_i, v_k) \rfloor > r_k$ . If this holds, it is clear that the node  $v_i$  is not in  $AA_k$ . Also, note that these conditions are an extension of (2) (in Claim 1) via the use of the upper and lower bounds on the distances between nodes. Thus, in the case that either  $v_i$  (or  $v_j$ ) and  $v_k$  or  $v_i$  and  $v_j$  are reference nodes the exact distances between these nodes are known and Claim 3 reduces to Claim 1.

An illustration of the use of Claim 3 is presented in Figure 8. The single avoidance area is marked by the red nodes where the area center node is  $e$  and  $r=1$  ( $AA(e,1)$ ). According to (6), routing from  $s$  to any node  $x$  obeying  $d(s,x) - r < 2$  (yellow nodes in the figure) is safe. Using node  $t$  as reference in (8), we can see that routing to any of the blue nodes from  $s$  is safe as well. For example, consider routing from  $s$  to node  $v$  we have:  $\lceil d(s,v) \rceil < \lfloor d(s,e) \rfloor + \lfloor d(v,e) \rfloor - 2r$  where  $d(s,v)=3$ ,  $d(s,e)=4$  and  $\lfloor d(v,e) \rfloor = \lceil 6-3 \rceil = 3$  leading to:  $3 < 4+3-2$ . Since (8) holds the path  $s-v$  is safe.

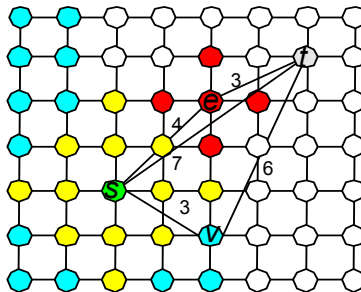


Figure 8 Area avoidance routing from  $s$  using Claim 3.

**Corollary 3:** If Claim 3 holds for all  $AA_k$  ( $\forall k:1..K$ ) then a packet between  $v_i$  and  $v_j$  will traverse a safe path.

Further, we will show two algorithms for area avoidance routing. The second algorithm we present relies on the following fact that ties the safe routing between pair of nodes and the safe routing between the witnesses for  $\lceil d(v_i, v_j) \rceil$  :

**Fact 1:** If Corollary 3 holds there must exist a witness  $v_w$  ( $v_w \in RN$ ) for  $\lceil d(v_i, v_j) \rceil$  such that  $v_w \rightarrow v_i$  and  $v_w \rightarrow v_j$  are safe. That is, if the path  $v_i \rightarrow v_j$  is safe, there must exist  $v_w$  ( $v_w \in RN$ ) such that the paths  $v_i \rightarrow v_w$  and  $v_w \rightarrow v_j$  are safe.

Fact 1 comes immediately from the observation that the witness for  $\lceil d(v_i, v_j) \rceil, v_w$ , testifies for the upper bound of the distance under the assumption that the shortest path may traverse through it. Since  $d(v_i, v_w) \leq \lceil d(v_i, v_j) \rceil$  (and  $d(v_j, v_w) \leq \lceil d(v_i, v_j) \rceil$ ), if (6), (7) or (8) held for  $\lceil d(v_i, v_j) \rceil$  the condition will hold for  $d(v_i, v_w)$  (and  $d(v_j, v_w)$ ) as well.

One should note that if  $v_i \rightarrow v_w \rightarrow v_j$  is safe it is not guaranteed that the path  $v_i \rightarrow v_j$  is safe.

In comparison to Corollary 1, Corollary 3 extends the ability of finding paths through nodes that are not necessarily reference nodes as demonstrated in Figure 9. In that example, no area avoidance path that uses the reference nodes ( $s, v$  and  $t$ ) as hubs can be found. However, the path  $s \rightarrow a \rightarrow b \rightarrow t$  can be found. Using witness  $t$  for  $\lfloor d(v_a, v_e) \rfloor$  in (8) testifies for the path  $s \rightarrow a$ , witness  $v$  testifies for the segment  $a \rightarrow b$  in (6) and witness  $s$  testifies for  $\lfloor d(v_b, t) \rfloor$  in (8) for the segment  $b \rightarrow t$ .

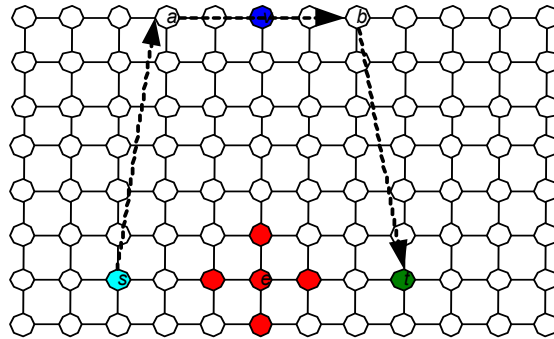


Figure 9 Area avoiding path through arbitrary nodes.

Using Corollary 3 one can verify whether a path is area avoiding based on algebraic calculations of distances. Another method to find area-avoiding paths is similar to the one introduced in Corollary 2. The basic idea is that we can compute the list of safe nodes. Also, we can find the set of nodes that are not ruled out from being on the shortest path between arbitrary pair of nodes. In case the latter nodes are all safe nodes, it is guaranteed that the path between the pair of nodes is safe.

Following the definitions in Section 3 (used in Corollary 2), let  $S$  denote the set of safe nodes in the network.

Let  $v_i$  and  $v_j$  denote two arbitrary nodes and let  $v_{r1}$  and  $v_{r2}$  denote two reference nodes. Let  $U_{v_{r1}v_{r2}}(v_i, v_j)$  denote the set of nodes that are not ruled out from being on  $v_i-v_j$  from the point of view of the reference nodes  $v_{r1}$  and  $v_{r2}$ .

**Claim 4:** An arbitrary node  $v_q$  is in  $U_{v_{r1}v_{r2}}(v_i, v_j)$  ( $v_q \in U_{v_{r1}v_{r2}}(v_i, v_j)$ ) only if:

$$d(v_{r1}, v_q) + d(v_{r2}, v_q) \leq \lceil d(v_i, v_j) \rceil + d(v_{r1}, v_i) + d(v_{r2}, v_j), \tag{9}$$

where  $d(v_{r1}, v_i) + d(v_{r2}, v_j) \leq d(v_{r1}, v_j) + d(v_{r2}, v_i)$ .

**Proof:** According to Observation 1, node  $v_q$  is on one of the shortest paths between  $v_i$  and  $v_j$  only if  $d(v_i, v_j) = d(v_i, v_q) + d(v_j, v_q)$ . Using the triangular inequality we have:

- 1)  $d(v_{r1}, v_q) \leq d(v_{r1}, v_i) + d(v_i, v_q)$
- 2)  $d(v_{r2}, v_q) \leq d(v_{r2}, v_j) + d(v_j, v_q)$

Putting 1) and 2) together we have:

$$3) \quad d(v_{r1}, v_q) + d(v_{r2}, v_q) \leq d(v_{r1}, v_i) + d(v_i, v_q) + d(v_{r2}, v_j) + d(v_j, v_q).$$

Since  $v_q$  is on  $v_i-v_j$  and they are not references, we have to use the distance estimation between  $v_i$  and  $v_j$ :

$$4) \quad d(v_j, v_q) + d(v_i, v_q) \leq \lceil d(v_i, v_j) \rceil.$$

Putting 3) and 4) together we have get (9).

An illustration of this claim is shown in Figure 10. In that figure nodes  $r1$  and  $r2$  are the reference nodes. Node  $v_q$  is potentially on the  $v_i-v_j$  only if (9) holds. That is, for  $v_q \in U_{v_{r1}v_{r2}}(v_i, v_j)$  the following must hold:  $d(v_{r1}, v_q) + d(v_{r2}, v_q) \leq 17 + 3 + 4 = 24$  (note that  $\lceil d(v_i, v_j) \rceil = 17$ ). For any  $v_q$  on  $v_i-v_j$  having  $d(v_i, v_q) = x$  (and  $d(v_j, v_q) = 17 - x$ ) we have  $d(v_{r1}, v_q) + d(v_{r2}, v_q) \leq (17 - x + 4) + (x + 3) = 24$  and thus  $v_q \in U_{v_{r1}v_{r2}}(v_i, v_j)$ . It is clear that  $v_q \in U_{v_{r1}v_{r2}}(v_i, v_j)$  for any  $v_q$  having  $d(v_{r1}, v_q) + d(v_{r2}, v_q) < 24$  since the actual shortest path between  $v_i$  and  $v_j$  can be smaller than the estimated ( $\lceil d(v_i, v_j) \rceil = 17$ ).

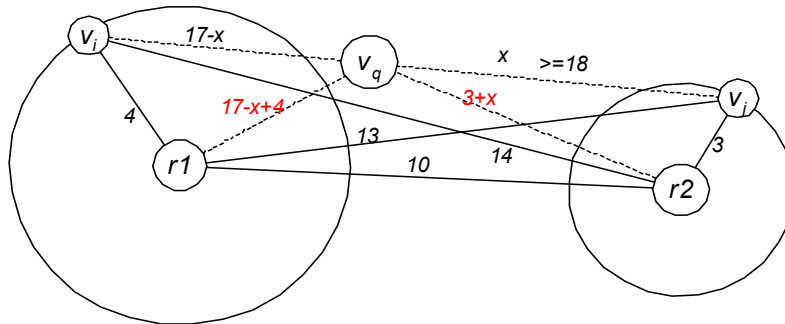


Figure 10 Bounds on the distances from  $v_{r1}$  and  $v_{r2}$  on  $v_i-v_j$ .



Note that even if (9) holds it does not an indication for  $v_q$  of being on  $v_i-v_j$ . But, it is guaranteed that all the nodes for which (9) does not hold ( $\forall v_q : v_q \notin U_{v_i, v_j}(v_i, v_j)$ ) are ruled out of being on  $v_i-v_j$ .

**Claim 5:** If  $\exists v_{r1}, v_{r2} : U_{v_i, v_j}(v_i, v_j) \subset S$ , the path between  $v_i$  and  $v_j$  is avoiding  $AA$ .

**Proof:** According to Claim 4,  $v_i-v_j$  may contain only nodes in  $U_{v_i, v_j}(v_i, v_j)$ . The set  $S$  contains only safe nodes. When  $U_{v_i, v_j}(v_i, v_j) \subset S$  the shortest path between the nodes may not contain any nodes in  $AA$  and thus, according to definition it is safe. ■

The following fact ties between the safe routing between pair of nodes for which Claim 5 holds and the safe routing between the nodes and the references:

**Fact 2:** If Claim 5 holds for  $v_{r1}, v_{r2}, v_i, v_j$  the path  $v_i-v_{r1}-v_{r2}-v_j$  is also area avoiding.

This fact comes immediately from (7) (or (6)). In case  $v_i-v_{r1}$  or  $v_{r1}-v_{r2}$  or  $v_{r2}-v_j$  are not area avoiding there exists node  $v_q$  such that  $v_q \notin S$  and  $v_q \in U_{v_i, v_j}(v_i, v_j)$ . Thus,  $U_{v_i, v_j}(v_i, v_j) \not\subset S$  in contradiction with Claim 5.

The example in Figure 11 illustrates the use of Claim 5. Nodes  $v_{r1}$  and  $v_{r2}$  are reference nodes and  $v_e$  is the area center node with  $r=1$ . The question is whether routing from  $v_{r1}$  to  $v_i$  is avoiding  $AA$ . One cannot infer from Corollary 3 that path  $v_{r1}-v_i$  is area avoiding. However, using Claim 5, any node  $v_q$  having  $d(v_{r1}, v_q) + d(v_{r2}, v_q) \leq [d(v_{r1}, v_i)] + d(v_{r1}, v_{r1}) + d(v_{r2}, v_i) = 4 + 0 + 1 = 5$ , is not ruled out from being on the shortest path. The set of nodes is then:  $U_{v_i, v_j}(v_i, v_j) = \{v_a, v_b, v_f, v_g, v_h, v_i, v_j\}$  (the value of  $d(v_{r1}, v_q) + d(v_{r2}, v_q)$  is marked above the nodes in the figure). The set of safe nodes is:  $S = \{v_{r1}, v_a, v_b, v_f, v_g, v_h, v_{r2}, v_i, v_j\}$ . Since  $U_{v_i, v_j}(v_i, v_j) \subset S$  the path  $v_{r1}-v_i$  is safe and also the paths  $v_{r1}-v_{r2}$  and  $v_{r2}-v_j$ .

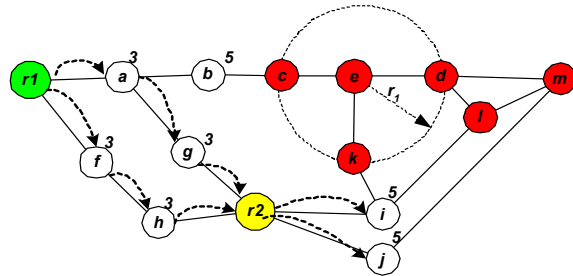


Figure 11 Example of area avoidance routing using Claim 5.

Using Corollary 3 and Claim 5, we will now show two algorithms for enhanced area-avoiding routing. The first greedy algorithm, seeks safe paths by exhaustively checking all available information over Corollary 3 and Claim 5. However, its complexity is very high. The second restricted algorithm seeks only for paths of the form  $v_{r1}-v_{i1}-v_{r2}-v_{i2}-$

...— $v_{rk}$ — $v_{ik}$ , where  $v_{ij}$  is a reference node and  $v_{ij}$  is an arbitrary node (not necessarily a reference). This algorithm is of significantly lower complexity in the expense possibly longer paths. However, it is guaranteed that if the greedy algorithm finds an area-avoiding path the second restricted algorithm will find one as well.

The concept behind the first greedy algorithm is to pre-compute the upper and lower bound of the distances between every pair of nodes in  $V$ . Then, following Corollary 3 and Claim 5 the algorithm validates whether the path between every pair of nodes is avoiding  $AA$ . In case it is, the nodes to a new graph  $G'$  with a link connecting them. The length of the link is set to the upper bound for the path length between the nodes. At the last stage,  $s$ — $t$  is found by looking for the shortest path (Dijkstra algorithm) on  $G'$ . The relay nodes for loose source routing are the nodes on the shortest path found in  $G'$ . Clearly, the algorithm guarantees that if a safe path can be found using the available distance vector information, it will be found.

**Input:**

1.  $RN = \{v_1, \dots, v_k\}$  - Set of reference nodes (including  $s$  and  $t$ )
2.  $T(v_i) \quad \forall v_i \in RN$
3.  $AA_1, \dots, AA_K, K$  - Number of avoidance areas.

**Algorithm GreedyAARouting** ( $s, t, RN, T$ )

1. According to Observation 3 find  $\lceil d(v_i, v_j) \rceil$  and  $\lfloor d(v_i, v_j) \rfloor \quad \forall v_i, v_j : v_i, v_j \in V$
  2. Construct  $G'(V', E')$
  3. For all pairs  $v_i, v_j$  in  $V$ :
    - a. If Corollary 3 holds (here we have to check for all  $AA_k$ ) or if Claim 5 holds for  $v_i, v_j$  (to validate Claim 5,  $S$  has to be pre computed)
      - i.  $V' \leftarrow v_i, v_j$
      - ii.  $E' \leftarrow e(v_i, v_j), c_{e(v_i, v_j)} = \lceil d(v_i, v_j) \rceil$
  4. Find shortest path,  $SP$ , between  $s$  and  $t$  in  $G'(V', E')$
- Direct the packet via the set of nodes in  $SP$  using loose source routing

**Proof of Correctness and complexity analysis:** Corollary 3 and Claim 5 guarantee that the paths between the node pairs in  $G'$  are area avoiding. Thus, any path composed of these nodes is area avoiding.

The complexity of the algorithm is as follows: 1) Calculation of  $\lceil d(v_i, v_j) \rceil$  and  $\lfloor d(v_i, v_j) \rfloor$  requires  $O(|RN||V|^2)$ . 2) Checking Corollary 3 for all node pairs with respect to all avoidance areas:  $O(K|V|^2)$ . 3) For checking Claim 5:  $O(K|RN||V|)$  for computing  $S$  and  $O(|RN|^2|V|^3 \log(|V|))$  for computing  $S \cap U_{v_1, v_2}(v_i, v_j)$  for every pair of nodes using all pairs of reference nodes. 4) Construct  $G'$  and find  $s$ — $t$ :  $O(|E| + |V| \log |V|)$ . Altogether we have the high complexity of:  $O(|RN|^2|V|^3 \log(|V|))$ . ■

The high complexity of the greedy algorithm comes mainly from validation of Claim 5 for all pairs of nodes. Nonetheless, the complexity of checking Corollary 3 for all node pairs is also significant. The following algorithm uses Fact 1 and Fact 2 implying that if the greedy algorithm used the path  $v_i \rightarrow v_j$ , the path  $v_i \rightarrow v_{r1} \rightarrow v_j$  or  $v_i \rightarrow v_{r1} \rightarrow v_{r2} \rightarrow v_j$  is also area avoiding. Finding the latter paths is of lower complexity as will be demonstrated in the following algorithm. But, such paths may be longer (there is no practical bound on the additional length since  $v_i \rightarrow v_{r1} \rightarrow v_j$  might be longer in factor of  $2D$  than  $v_i \rightarrow v_j$ , where  $D$  is the highest distance in the network).

**Local variables:**

$M(v_i) \quad \forall v_i : v_i \in RN$ - Set of nodes for which routing between them and reference node  $v_i$  guarantees area avoidance routing including the node itself (Claim 5 is validates only between reference nodes and not all node in  $V$ ).

**RestrictedAARouting** ( $s, t, RN, T$ )

1. According to Observation 3 find  $\lceil d(v_i, v_k) \rceil$  and  $\lfloor d(v_i, v_k) \rfloor \quad \forall v_i \in V$  and area center node  $v_k$  ( $k=1, \dots, K$ ).
2. Find  $M(v_i) \quad \forall v_i : v_i \in RN$  using Corollary 3
3. Find  $M(v_i) \quad \forall v_i : v_i \in RN$  using Claim 5 between reference nodes only
4. Construct  $G'(V', E')$
5. For every pair of reference nodes in  $RN$ 
  - a. For every  $\forall v_k \in V$ 
    - i. If  $v_k \in M(v_i) \cap M(v_j)$ 
      1. if  $e(v_i, v_j) \notin E'$  then  $E' \leftarrow e(v_i, v_j)$ ,  $c_{e(v_i, v_j)} = \infty$
      2. if  $c_{e(v_i, v_j)} > d(v_i, v_k) + d(v_j, v_k)$ 
        - a.  $c_{e(v_i, v_j)} = d(v_i, v_k) + d(v_j, v_k)$
        - b.  $B(v_i, v_j) = v_k$  //holds the intermediate relay node
6. Find shortest path,  $SP$ , between  $s$  and  $t$  in  $G'(V', E')$
7. For every pair in  $SP$ 
  - a. if  $B(v_i, v_j) = v_k$  add  $B(v_i, v_j)$  between  $v_i$  and  $v_j$
8. Direct the packet via the set of nodes in  $SP$  using loose source routing

**Proof of correctness:** The algorithm relies on the correctness of the greedy algorithm. Based on Fact 1 and Fact 2, in case the greedy algorithm finds an area-avoiding path, this algorithm will also find one. To prove this we show that for every area avoiding link  $v_i \rightarrow v_j$  found by the greedy algorithm using Corollary 3 and Claim 5, the restricted algorithm will also find an area avoiding path between these two nodes. When computing the shortest path, using Dijkstra algorithm, the path between  $v_i$  and  $v_j$  will be found in  $G'$ .

The greedy algorithm computes  $\lceil d(v_i, v_j) \rceil$  and  $\lfloor d(v_i, v_j) \rfloor$ . Using these bounds, it checks between every pair of nodes whether Corollary 3 or Claim 5 hold. In case one of them holds, a link between the pair of nodes is added to  $G'$ . Let assume that a link  $v_i \rightarrow v_j$  was

added to  $G'$  in the greedy algorithm because Corollary 3 holds. In that case, according to Fact 1 either  $v_i, v_j$ , or the witness,  $v_w$ , for  $\lceil d(v_i, v_j) \rceil$  are reference nodes and they are safe. The restricted algorithm will find the links  $v_i-v_w$  and  $v_w-v_j$  as area avoiding according to the conditions in Corollary 3. Thus in  $G'$  of the restricted algorithm the path  $v_i-v_w-v_j$  is safe.

In case the link  $v_i-v_j$  was added to  $G'$  in the greedy algorithm because of Claim 5, according to Fact 2 the path  $v_i-v_{r1}-v_{r2}-v_j$  is also area avoiding. In the restricted algorithm the segment  $v_{r1}-v_{r2}$  will be added to  $G'$  since it must be safe according to (7) (only nodes from  $S$  are on  $v_{r1}-v_{r2}$  otherwise (7) does not hold). The segments  $v_i-v_{r1}$  and  $v_{r2}-v_j$  must be added to  $G'$  in the restricted algorithm according to Corollary 3 (otherwise there is a node  $v_q$  because of which Corollary 3 does not hold and  $U_{v_i, v_j}(v_i, v_j) \not\subset S$  since  $v_q \in U_{v_i, v_j}(v_i, v_j)$  and  $v_q \notin S$ ). Thus, the safe path  $v_i-v_{r1}-v_{r2}-v_j$  will be found in  $G'$  in the restricted algorithm.

Note that at the end of the algorithm the estimated path length of the shortest path is that returned by the Dijkstra algorithm. Practically it may be significantly shorter since the algorithm uses the upper bound for length in links in  $G'$ .

**Complexity analysis:** Since every segment contains at least one reference node, Corollary 3 should be checked only for these nodes (and not for all nodes in  $V$ ) for all  $AA_k$ . Thus, for computing  $\lceil d(v_i, v_k) \rceil$  and  $\lfloor d(v_i, v_k) \rfloor$  between area center nodes and reference nodes (step 1) required  $O(|K||RN|^2)$ . To compute  $M(v_i)$  (step 2) using Claim 3 over all  $AA_k$  requires  $O(|RN|^2|V||K|)$ . Next, to compute  $M(v_i)$  for all reference nodes (step 3) using Claim 5 requires  $O(|RN|^2|V|\log(|V|))$ . Step 3 requires  $O(|V||RN|^3\log(|RN|))$ . Finding the shortest path:  $O(|E|+|RN|\log(|RN|))$ . Altogether we have the time complexity of:  $O(|RN|^2|V|\log(|V|))+O(|RN|^2|V||K|)+O(|V||RN|^3\log(|RN|))$ . If  $|RN|\log(|RN|)<\log(|V|)$  (the number of reference nodes should be a small fraction of  $|V|$ ) and  $|K|<\log(|V|)$ , we have the complexity:  $O(|V||RN|^2\log(|V|)$  similar to the complexity of the *HubBasedAreaAvoidanceRouting* algorithm studied in the previous section).

In comparison to the greedy algorithm the complexity is reduced from  $O(|RN|^2|V|^3\log(|V|))$  to  $O(|V||RN|^2\log(|V|))$  with the cost of possibly longer paths. ■

After finding an area-avoiding path in the restricted algorithm one can try to enhance the result. One way is to seek for shortcuts between relay nodes using Corollary 3. For example, for path  $s-v_{i1}-v_{r1}-v_{i2}-v_{r2}-v_{i3}-t$ , one can verify whether the path  $v_{i1}-v_{i2}$  or  $v_{i2}-v_{r1}-v_{i3}$  is area avoiding. In case it is,  $v_{r1}$  or  $v_{r2}$  may be removed from the list of relays. This will result in shorter path and reduced number of relay nodes, which is also an important objective.

**Remark:** The routing techniques mentioned here can be used as well for shortest path avoidance routing (as mentioned in [30]). Shortest path avoiding is a particular case of area avoidance where the set of nodes in  $AA$  can be computed using Observation 1 by distances from  $s$  and  $t$ .

## 5 Dynamic Reference Selection for Area Avoidance Routing

In the previous sections we studied the problem of finding area-avoiding paths given a set of reference nodes. We assumed that the source obtained the distance vector information from these nodes in some offline or online manner. In this section we address the problem of dynamically selecting reference nodes. That is, how should the source dynamically choose reference nodes for finding a safe path towards the destination. In this model the source does not have a set of known reference nodes neither does it have any distance vector information besides its own. But, it may query all other nodes in the network for their distance vector information. In this model the main objective is to find area-avoiding path using minimal number of queries.

More formally, given a network  $G(V,E)$ , the set of avoidance areas  $AA(U,R)$  and the distance vector information stored in every node  $T(v_i)$ , we are interested in an algorithm for reference node selection that increases the likelihood of finding an area avoidance path with minimal number of queries. Other objective we pay attention to are: *i*) Finding short paths *ii*) Paths having minimal number of relay nodes (relay nodes are inserted into packet header and thus increase per-packet overhead). We assume that retrieving the entire distance vector information from a node is of fixed cost w.l.g is assumed to be 1. Alternative approaches are to have the cost proportional to the number of distances queried, or being fixed for up to  $k$  distances; we leave these two alternatives for future study.

As we showed in Proposition 1, there is no deterministic algorithm can guarantee finding area-avoidance path, by querying less than  $|V|/2-2$  reference nodes, even if such path exists. Clearly, querying  $|V|/2-2$  reference nodes is very expensive and impractical. To address this problem we will introduce two heuristic algorithms for selecting reference nodes. We will focus our analysis on several structured networks, for which we will provide general results: *i*) Manhattan graph (we will refer to as Grid-4) see Figure 12 *ii*) A network in which each node has exactly 8 neighbors (we will refer to as Grid-8), see Figure 19. We will then complement the analytic results of the structured networks with simulations results carried over a wide set of random networks.

Selecting reference nodes in the area avoidance, in particular the are center nodes, sounds like a very good idea since the exact distances from the area center nodes help in getting the exacts distances needed for the conditions in Claim 3. Also, by having the distance vector of the area center node the maximal cardinality set of safe nodes,  $S$ , will be found which increases the chances of finding safe paths using Claim 5. However, since nodes in the avoidance area may be malicious, and provide misleading routing information, the reference node selection algorithms will select nodes only from the set of safe nodes,  $S$ . One should note that in other cases, when there is no threat of getting misleading routing information (e.g. when the thread is only eavesdropping on links), querying none safe nodes may be feasible<sup>7</sup>.

---

<sup>7</sup> E.g. when routing via area avoiding paths for enhancing QoS querying all nodes is acceptable.

The general framework for area avoidance routing algorithm is shown next. The algorithm iteratively selects reference nodes and queries them. Then, using the algorithms presented in the previous section, the algorithm seeks for area avoiding path. In case such path exists the algorithm uses it and stops. Otherwise, the algorithm continues either until going through all nodes or up to a limited number of iterations.

#### **AreaAvoidanceRoutingByQuery( $s, t, AA$ )**

1. If  $d(s,t) < d(s,v_k) - r_k$  for all  $AA_k (\forall k: 1..K)$ 
  - a. Direct packet to  $t$  (the shortest path is safe)
2. Get  $T(t)$  //query the target
3. If the path  $s \rightarrow t$  is safe use the shortest path ( $SP$ ) and return
4.  $RN \leftarrow s, t$  //The  $RN$  is the set of reference nodes selected so far and used for calculation of safe path in *GreedyAARouting()*
5. While (1)
  - a. Find  $S$  (set of safe nodes)
  - b.  $v_i = SelectRefernces(S, RN)$
  - c. Get  $T(v_i)$  //Query node  $v_i$
  - d.  $RN \leftarrow v_i$
  - e. If *GreedyAARouting*( $s, t, RN, T$ ) finds area avoiding path
    - i. Direct packet on the path and exit

The core of the algorithm lies in the procedure of selecting the reference node (*SelectRefernce* procedure). We analyze two such procedures. The first algorithm selects nodes, from the set of safe nodes in a random manner. The idea is that in case the network is well connected small number of queries will suffice.

#### **RandomSelectionOfReferences( $S, RN$ )**

Randomly select a node in  $S$  (that is not in  $RN$ )

The second greedy algorithm selects reference nodes that expand the set of *safe reachable* nodes (nodes that can be reached from  $s$  via safe paths). The set of safe reachable nodes can be compute in a naïve way by checking whether *GreedyAARouting* returns positive result for the path between  $s$  and every other node. An algorithm with lower complexity for this task would find the set of safe reachable reference nodes and for each such reference node the set of non-reference nodes that are reachable from it. Also this set can be computed for new nodes added to  $S$  in every iteration (there is no need to find all safe reachable nodes in every iteration). The expansion of the set of safe reachable nodes is conducted by selecting reference nodes from the set of safe reachable nodes that are as close as possible to  $t$  as far as possible from  $s$ .

#### **GreedySelectionOfReferences( $S, RN$ )**

1. Choose  $v_i$  from  $S$  such that:
  - a.  $v_i$  is not in  $RN$
  - b. The path  $s \rightarrow v_i$  is safe
  - c.  $v_i$  is closest to  $t$  (minimal  $d(v_i, t)$ )

- i. If there are several candidates Maximal  $d(v_i, s)$  (as far as possible from  $s$ )

One should note that the probabilistic approach presented in Section 3.1 can be also adopted here for choosing the path with the highest area avoiding potential.

Next we analyze the performance of the algorithms on the graphs Grid-4 and Grid-8. For the sake of simplicity we will assume that there is only a single avoidance area and the network is of size  $m \times m$ . Also we assume that  $s$  and  $t$  are in the middle of the grid and are not in the avoidance area. We will also assume that  $d(s, t) < \sqrt{m/2}$ . This assumption fits with the spheric shape of the globe, allowing every node to be considered as residing at “the middle of the network”. In what follows we focus on the following problem:

Given  $AA(v_l, r)$  and the distances  $D_1 = d(s, v_l)$  and  $D_2 = d(t, v_l)$ , derive an upper bound on the expected number of queries required for finding a guaranteed safe path (in this setting a safe path always exists).

For the sake analysis tractability, we assume that the area center node can be queried and both the random and greedy selections strategies select it as a second reference node after selecting  $t$ . The nodes selected afterwards follow the random or greedy strategy. As will be demonstrated in the simulation results, the performance of the greedy selection strategy that selects *only safe nodes* results in approximately similar number of queries.

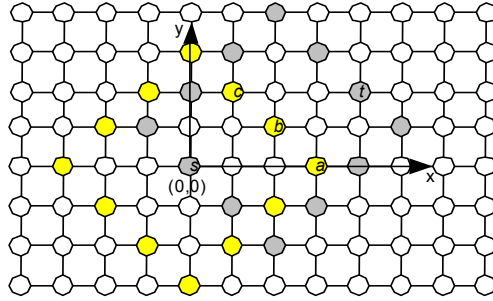
### 5.1 Analysis of the Random Selection on Grid-4

In the following analysis we will assume that  $d(s, v_l) > 2r + 1$  and  $d(t, v_l) > 2r + 1$  ( $s$  and  $t$  are not in the rectangle that wraps the avoidance area)<sup>8</sup>. To calculate the expected number of queries one should first note that there are exactly  $4D_1$  potential area center nodes (yellow nodes in Figure 12) having distance  $D_1$  from  $s$ . For every such area center nodes there are exactly  $4D_2$  potential destinations having distance  $D_2$  from  $v_l$  (the gray nodes in Figure 12 mark potential destinations around node area center node  $b$  having  $d(t, b) = 3$ ). Altogether there are  $16D_1D_2$  possible *configurations* (relative positions in the Grid-4) for  $s$ ,  $v_l$  and  $t$ . For the analysis we set a system of coordinates around  $s$ . That is, let  $s$  possess the coordinate  $(0, 0)$ . Nodes  $v_l$  and  $t$  will possess the coordinate  $(x_{v_l}, y_{v_l})$  and  $(x_t, y_t)$  respectively.

There are four symmetric configurations (each quarter in the coordinate system is symmetric to the other quarters). Thus, for calculating the expected number of queries we account only the positive quarter in which there are  $4D_1D_2$  configurations.

---

<sup>8</sup> This assumption does not pose a major limitation. Even in case this assumption does not hold by at most two queries one can find such reference nodes.



**Figure 12 Configurations in Grid-4.**

In the following analysis we account for all possible configurations of area center nodes and destinations and for each group of configurations we show the upper bound for the number of expected queries. We do the calculations for even values of  $D_1$  and  $D_2$ . The calculations for odd values are almost similar.

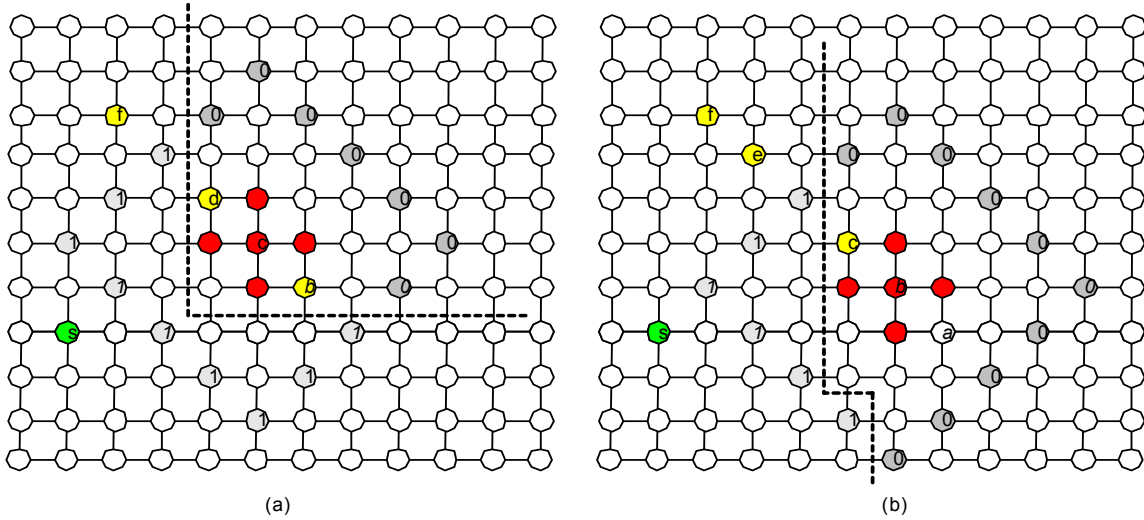
Note that if  $d(s,t) < d(s,v) - r$  there is no need to query  $t$  since from  $T(s)$  it is clear that  $s \rightarrow t$  is safe. Accounting for such configurations is very complicated since there are many cases to account for. For example, in case  $D_2 \geq 2D_1$ ,  $t$  must always be queried, but for  $D_2 < 2D_1$ , for up to  $2(D_1 - r) - 1$  destinations we will have  $d(s,t) < d(s,v) - r$  and thus no queries are needed. For the sake of simplicity, in what follows we will neglect this issue and assume that  $s$  always queries the destination even though it might be useless. Since in the analysis below we show an upper bound to the number of queries this assumption is valid.

We divide the potential area center nodes into two groups: (a) nodes having  $x_v > r$  and  $y_v > r$  and (b) all other nodes (any node,  $v_i$ , such that there is a node from  $AA$  located on  $s \rightarrow v_i$ ). In the first group there are  $D_1 - 2r - 1$  area center nodes while in the second  $2r + 1$  area center nodes.

In group (a), using Claim 1, querying destination nodes having  $x_t < x_v - r$  or  $y_t < y_v - r$  will reveal that the path between  $s$  and  $t$  is safe. Note that for any node,  $v_i$  in group (a) no node from  $AA$  is located on  $s \rightarrow v_i$ . There are exactly  $3D_2 - 2r - 1$  such destinations (out of  $4D_2$ ). For illustration consider the network in Figure 13-(a) where we present a Grid-4 network with  $AA(c,1)$ . All possible destination nodes having distance  $D_2$  from area center node  $c$  are marked in gray. The destinations for which the distance vector of  $s$  and  $t$  is sufficient for finding safe path are marked in '1'.

In group (b),  $T(t)$  of any destination having  $x_t < x_v - r$  (or  $y_t < y_v - r$ ) and  $|y_t| \geq |y_v - r|$  (or  $|x_t| \geq |x_v - r|$ ) exactly for  $2(D_2 - r) - 1 + |y_t|$  (or  $2(D_2 - r) - 1 + |x_t|$ ) will suffice to find safe path using Claim 3. Note that in this case, similarly to nodes in group (a), for any node  $v_i$  there is no node from  $AA$  that is located on  $s \rightarrow v_i$ . Altogether there are exactly  $2((D_2 - r) - 1) + \sum_{i=1..r} i$  such destinations. For illustration consider the network in Figure 13-(b) where we present a Grid-4 network with  $AA(b,1)$ . All possible destination nodes having distance  $D_2$  from area center node  $b$  are marked in gray. Those destinations for which the distance vector of

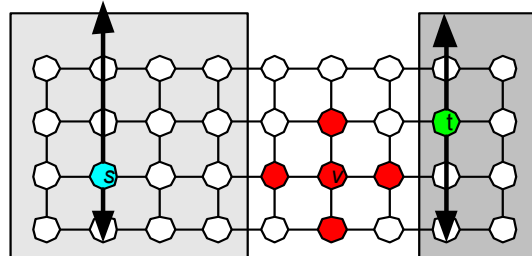
$s$  and  $t$  is sufficient for finding safe path are marked in ‘1’. Those nodes are all nodes from group (a) and the nodes from group (b) for which the above property holds. Summing the configurations, in groups (a) and (b) we have that for  $(D_1-2r-1)(3D_2-2r-1) + (2r+1)(2D_2-2r-2 + \sum_{i=1..r} i)$  out of  $4D_1D_2$  configurations, for which a single query of the destination leads to finding area avoiding path.



**Figure 13 Destinations for which the distance vector information from  $s$  and  $t$  is sufficient for finding safe path are marked in ‘1’.**

For all other configurations additional distance vector information is required for finding safe paths. After querying the destination, the source will query the area center node. We will count all the configurations for which  $T(s)$ ,  $T(v)$  and  $T(t)$  will not sufficient for finding safe path using Claim 3. For the other configurations two queries (querying  $T(t)$  and  $T(v)$ ) will suffice.

Since we assume that  $D_1 > 2r + 1$ , using Claim 3, all nodes that are either on  $(0,x) \forall x : x = 1..m$  or/and on  $(0,y) \forall y : y = 1..m$  are known to have area-avoiding path from  $s$ . Similarly, from  $t$  there is area avoiding path to all nodes with coordinates  $(y_t,x) \forall x : x = 1..m$  or/and on  $(x_t,y) \forall y : y = 1..m$  (see illustration in Figure 14).



**Figure 14 Nodes to which safe routing is guaranteed given the distance vector information of  $s$ ,  $v$  and  $t$ . The gray areas mark the reachable nodes from  $s$  and  $t$ .**

Only for area center nodes from group (b) (where  $x_v > r$  and  $y_v \leq r$  or  $x_t - x_v > r$  and  $y_t - y_v \leq r$  and in the symmetric case where  $y_v > r$  and  $x_v \leq r$  or  $y_t - y_v > r$  and  $x_t - x_v \leq r$ ) see illustration in Figure

14) additional reference nodes will be required. In other words, if one of the shortest paths between  $s$  and  $t$  does not share nodes from  $AA$  a safe path can be found using  $T(s)$ ,  $T(t)$  and  $T(v)$  (see example in Figure 15) and thus for each area center node in group (b) no area-avoiding path will be guaranteed for  $2r+1$  destinations. Altogether we have  $(2r+1)^2$  configurations that require querying nodes other than  $t$  and  $v$ . In all other cases, querying  $t$  and  $v$  will sufficient for finding area-avoiding path.

Given the area center node from group (b), the probability of randomly selecting a reference node  $v_w$  whose coordinates will be in the range of  $|y_w - y_v| \leq r$  (or in the range  $|x_w - x_v| \leq r$  for the case where  $y_v > r, x_v \leq r$ , see illustration in Figure 16) is  $(2r+1)/m$  (a strip in the grid with high  $2r+1$ ) which is the expected number of queries for geometric distribution (derived from random selection) with success probability of  $1 - ((2r+1)/m)$ . Thus, the expected number of random queries until finding safe path in that case is:  $2 + (1 - ((2r+1)/m))^{-1}$ .

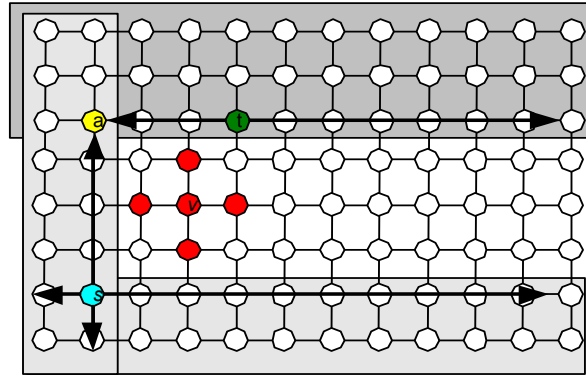


Figure 15 Area avoiding path using  $s$ ,  $v$  and  $t$  as references. The gray areas mark the reachable nodes from  $s$  and  $t$ . Any node in the intersection (e.g. node  $a$ ) is a relay that guarantees area avoiding path.

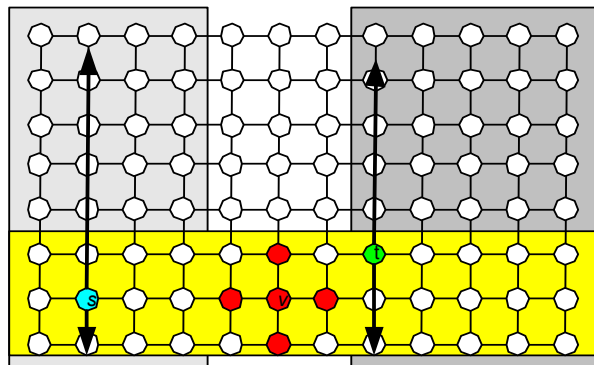


Figure 16 The yellow area marks all nodes that are useless as reference nodes for finding area-avoiding path between  $s$  and  $t$ . Any node out of this area will suffice.

Putting this all together we have  $4D_1D_2$  configurations from which:

- $x_1 \leq (D_1 - 2r - 1)(3D_2 - 2r - 1) + (2r + 1)(2(D_2 - r - 1) + \sum_{i=1..r} i)$  configurations that require a single query.

- $x_2 \leq (D_1 - 2r - 1)(D_2 + 2r + 1) + (2r + 1)(2(D_2 + 1) - \sum_{i=1..r} i)$  configurations that require exactly two queries.
- $x_3 \leq (2r + 1)^2$  configurations require  $2 + (1 - (2r + 1)/m)^{-1}$  queries.

The expected number of queries,  $X$ , in Grid-4 is then:

$$E[X] \leq \frac{x_1 + 2x_2 + x_3(2 + (1 - (2r + 1)/m)^{-1})}{4D_1D_2} \quad (10)$$

One should note that in case  $D_1 \gg r$  and  $D_2 \gg r$ , we have:

$$E[X] \leq \frac{5}{4} + \frac{D_1 + D_2}{4D_1D_2}, \quad (11)$$

which indicates that two queries (querying the destination and the area center node) are enough in most cases.

The path length returned by the algorithm is optimal since the returned path return goes around the avoidance area. The path length is limited by  $d(s,t) + 2r + 1$ .

Another question we can ask is: Given  $D_1$  and  $D_2$  and it is known that the area center node is on  $s-t$ . There are  $D_1D_2$  such configurations in each quarter. For these configurations  $s$ ,  $t$  and  $v$  must be queried. Using the analysis above, for  $(2r+1)^2$  configurations the algorithm will have to select additional references. Thus, we will have that the expected number of queries,  $X'$ , in Grid-4 when the area center node is on  $s-t$  is:

$$E[X'] \leq 2 + \left(1 - \frac{(2r+1)}{m}\right)^{-1} \frac{(2r+1)^2}{4D_1D_2} \quad (12)$$

This result indicates that if  $r \ll D_1$  and  $r \ll D_2$  or if  $r \ll m$  two queries (which is also the minimal number of queries when the area center node is on the shortest path) will suffice. This result agrees with the simulation results presented in Section 5.5.

To get an intuitive explanation of this result consider the network in Figure 17 where  $v$  is the area center node with  $r=1$  and nodes marked in '0', '1', '2' and '3' are potential destinations with  $D_2=4$ . For the destination marked by '0', it is clear from  $T(s)$  that  $s-t$  is area avoiding since  $d(s,t) < d(s,e) - r$ . For destinations marked by '1', then from  $T(s)$ ,  $T(t)$  and Claim 3  $s-t$  is safe. To consider points '2', if  $s-t$  does not share nodes with the  $AA$  a safe path via some intermediate node  $a$  can be found using  $T(s)$ ,  $T(t)$  and  $T(v)$ . Such node  $a$  can be reached safely both from  $s$  and  $t$  using Claim 3. In that case the safe path  $s-a-t$  will be found by *RestrictedAARouting*. The bright gray area in the figure marks the nodes having safe route from  $s$  and the dark gray marks the nodes having safe route from any destination marked by '2'. Only in the case that all shortest paths from  $s$  to  $t$  share nodes with the avoidance area (nodes marked by '3' in Figure 17, all in the yellow region), additional queries are required. In that case, any additional reference node out of the yellow area will suffice for finding safe path between  $s$  and  $t$ .

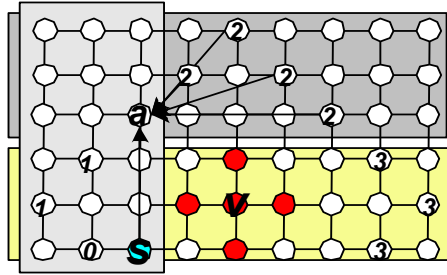


Figure 17 Safe nodes from  $s$  and  $t$  (numbers indicate possible position for  $t$ ).

### 5.2 Analysis of the greedy approach on Grid-4

The upper bound for the expected number of queries for the cases where  $s$ ,  $v$  and  $t$  are sufficient for finding a safe path is similar to the upper bound for the random algorithm. In the other  $(2r+1)^2$  configurations the number of queries is between 3 (querying  $t$ ,  $v$  and another node) and  $2r+3$  since it might require querying up to  $2r+1$  nodes for going around the avoidance area as illustrated in Figure 18. Thus, putting this into (10) (which should replace the coefficient of  $x_3$  in (10)) will produce an upper bound for the number of expected queries using the greedy selection strategy.

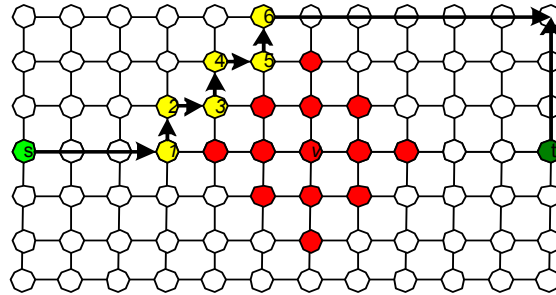


Figure 18 Greedy selection strategy in Grid-4. The nodes marked 1..6 may be queried before the area-avoiding path is found.

As can be seen in Grid-4 the random selection algorithm is superior in average case to the greedy selection algorithm.

### 5.3 Grid -8 with single avoidance area

To calculate an upper bound for the expected number of queries one should note that there are exactly  $8D_1$  potential area center nodes (yellow nodes in Figure 19) having distance  $D_1$  from  $s$ . For every area center node,  $v$ , there are exactly  $8D_2$  potential destinations having distance  $D_2$  from  $v$  (the gray nodes in Figure 19 mark potential destinations around node area center node  $v$  having  $d(t,v)=3$ ). Altogether there are  $64D_1D_2$  possible configurations for  $s$ ,  $v_1$  and  $t$ . For the analysis we set a system of coordinates around  $s$ . That is, let  $s$  possess the coordinate  $(0,0)$ . Nodes  $v_1$  and  $t$  will possess the coordinate  $(x_v,y_v)$  and  $(x_t,y_t)$  respectively.

There are four symmetric configurations (each quarter in the coordinate system is symmetric to the other quarters). Thus, for calculating the expected number of queries we

account only for the positive quarter in which there are  $16D_1D_2$  configurations ( $2D_1$  potential area center nodes and  $8D_2$  destinations with distance  $D_2$  from each area center node).

For the sake of simplicity we will only put an upper bound on the number of configurations that require more than two queries (querying  $t$  and  $v$ ) and count all other cases as if the sources queried  $t$  and  $v$  (two queries). A tighter bound can be found but requires accounting for many scenarios. For instance, for nodes marked in ‘b’ in Figure 20-(b) the distance vector information in  $T(s)$  and  $T(t)$  is sufficient for finding (using Claim 1) that the shortest path between them is safe. For some of these nodes  $T(s)$  by itself shows that the path to destination is safe.

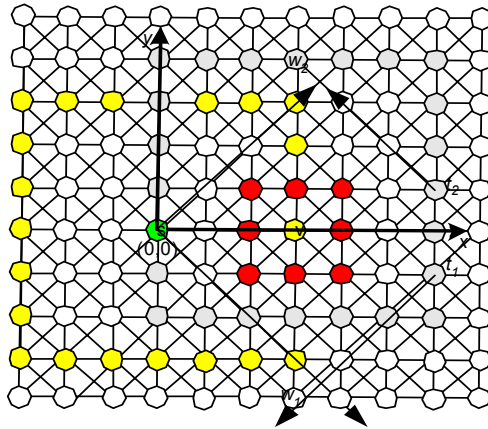
In Grid-8, given  $T(s)$ ,  $T(v)$  and  $T(t)$ , a safe path between  $s$  and  $t$  will not be found when using Claim 3 if and only if the following two conditions hold:

(i) The coordinates of  $v$  are  $|x_v - y_v| < 2r + 1$ . (There are  $2r + 1$  such configurations for area center nodes)

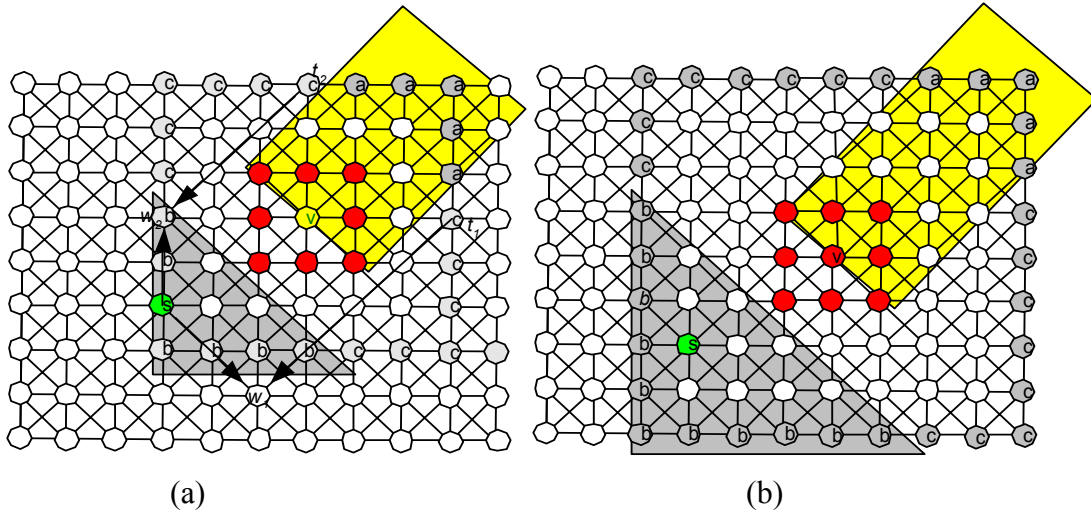
(ii)  $(x_t - y_t) > (x_v - y_v) - (2r + 1)$  and  $(y_t - x_t) > D_2$  or  $(y_t - x_t) > (y_v - x_v) - (2r + 1)$  and  $(x_t - x_v) > D_2$ .

For illustration, consider Figure 19 where area avoidance path can be found between  $s$  and destinations  $t_1$  and  $t_2$ , using only  $T(s)$ ,  $T(v)$  and  $T(t)$ . In that example  $D_1 = D_2 = 3$  and  $r = 1$  (where nodes  $w_1$  and  $w_2$  will be found as intermediate nodes). Another example with different coordinates for  $v$  is shown in Figure 20-(a). As can be seen, area avoiding path cannot be found using only  $T(s)$ ,  $T(v)$  and  $T(t)$  for all destinations in the yellow strip marked in ‘a’, for all other destinations, marked in ‘c’ this information is sufficient for guaranteeing safe routing.). Note that for nodes marked in ‘b’ either no queries are required or querying  $t$  will suffice, but we do not account in our analysis for these cases.

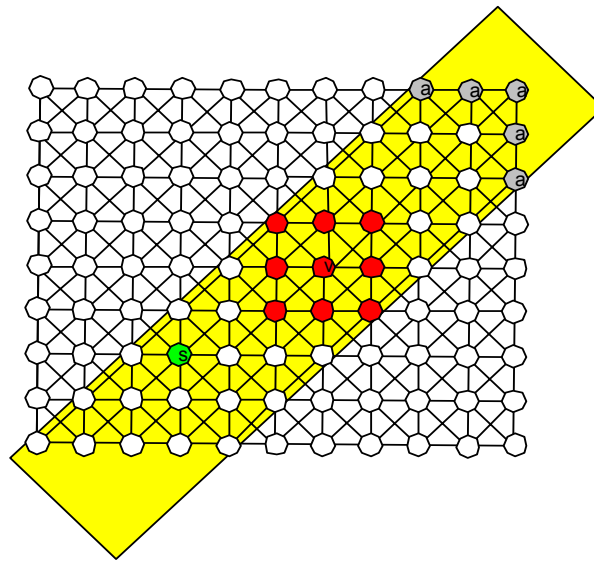
There are  $(2r + 1)^2$  configurations for which (i) and (ii) hold. Assuming the network is  $m \times m$ , querying  $m(2r + 1)/m^2 + \epsilon$  nodes ( $\epsilon$  expresses the tail which is bounded by:  $\epsilon \leq (2r + 1)^2/2$ . Assuming  $m \gg \epsilon$ ,  $\epsilon$  can be neglected) will not help for finding safe path, but all other nodes will do (see illustration in Figure 21 where any reference node from the yellow strip will not help in finding safe path). Thus, we have that the expected number of random selections of reference nodes is  $2 + (1 - ((2r + 1)/m + \epsilon))^{-1}$ , almost similar to the expected number in Grid-4.



**Figure 19** Safe path can be found using references  $s$ ,  $v$  and  $t_1$  or  $t_2$ .



**Figure 20** Destinations that will need one query are marked by  $b'$ , two queries marked by  $c'$  and more than two marked by  $a'$ .



**Figure 21** Useless references in case one of nodes  $a'$  the destination.

Putting this all together, we have  $16D_1D_2$  configurations from which:

- $x_2 = 16D_1D_2 - (2r+1)^2$  configurations require two queries.
- $x_3 = (2r+1)^2$  configurations require  $2 + (1 - ((2r+1)/(m+\epsilon)))^{-1}$  queries (we will omit the  $\epsilon$ ).

The upper bound for the expected number of queries,  $X$ , in Grid-8 is then:

$$E[X] \leq \frac{2x_2 + x_3(2 + (1 - (1 - (2r+1/m)))^{-1})}{16D_1D_2} \tag{13}$$

The path length returned by the algorithm is optimal in the case of Grid-8.

Given  $D_1$  and  $D_2$  and it is known that the area center node is on  $s-t$ . There are  $2D_1(8D_2+1)$  such configurations in each quarter. For these configurations  $s$ ,  $t$  and  $v$  must be queried. Using the analysis above, for  $(2r+1)^2$  configurations the algorithm will have to select additional references. Thus, we will have that the upper bound for the expected number of queries in Grid-8 when the area center node is on  $s-t$ ,  $X^r$ , is:

$$E[X^r] \leq 2 + (2 + (1 - (2r+1/m))^{-1}) \frac{(2r+1)^2}{16D_1 8D_2} \quad (14)$$

#### 5.4 Analysis of the greedy approach on Grid-8

The analysis for the expected number of queries is similar to the analysis for the random algorithm until the point where  $T(s)$ ,  $T(v)$  and  $T(t)$  are not sufficient for finding safe path. There are  $(2r_1+1)^2$  such configurations. In those configurations the upper bound for the number of queried is:  $2r+1$ . Putting this into (13) will provide the upper bound to the number of queries.

Here, as in the case of Grid-4, in the average case the random algorithm is superior to the greedy algorithm.

The analysis for the structured graphs above implies that in most cases querying the destination and the area center is enough to find safe path. Only in a small fraction of configurations additional queries are required and then the number of queries is expected to be small. We saw that the random algorithm in those networks is superior to the greedy algorithm. The intuition is that in these structured networks the concept of expanding the safe area, expressed in the greedy algorithm, is not working well but it guarantees finite number of queries. The random algorithm results in lower average of queries but, in the worst case the number of queries can grow to  $O(|V|)$ .

For the Grid-3 and Grid-6, the expected number of required references is about 2.

#### 5.5 Evaluation with Simulation

In this section, we evaluate the quality and the cost of the reference node selection algorithms for area avoidance routing. We concentrate on three goals. The first is to get the intuition for the number of queries required for finding a safe path, if such path exists. The second goal is to evaluate the resulting path lengths. The third goal is to evaluate the number of relay nodes that appear in packet header for source routing. This parameter is significant to get an intuition of the per-packet overhead required for safe routing.

#### 5.6 Simulation Methodology

We experimented the results of reference selection algorithms three network topologies: i) The Barabasi-Albert model that produces the power law distribution for network connectivity. We used the BRITE [6] generator for 1000 node networks with parameter

$m=2$ . ii) The Waxman model that deals with general random networks. We used the BRITE generator for 1000 node networks using parameters  $m=2$ ,  $\alpha = 0.15$  and  $\alpha = 0.2$ . iii) A simple model for ad-hoc network topologies in which nodes are connected to each other if their Euclidian distance is smaller than a constant parameter. We defined a plane of  $100 \times 100$  on which we randomly distributed  $V$  (a simulation parameter) nodes. Each node had a random location  $(x,y)$ . For each pair of nodes having Euclidian distance less than a parameter  $D$  we added an edge. The cost of the edges in all network models was set to one (which means that distances between nodes in the networks were counted by hop-count). We validated that all tested networks were connected.

For each simulation we generated 30 random networks under the same parameters. In each network we randomly selected 50 source and destination pairs. We used single avoidance area with parameter  $r$ . In order to concentrate on “interesting” scenarios (in which  $T(s)$  and  $T(t)$  are insufficient for finding a safe path), we selected an area center node on the shortest path between the source and the destination having the distance that is half (or as close as possible to half) of the shortest path length. The distances between the source and the area center node and the destination and the area center node was at least  $r+1$ . The number of queries was limited to 20, that is, if the algorithm did not find a safe path after querying 20 reference nodes, the algorithm stopped and returned a negative result.

Figure 22-28 depict the simulation results using ad-hoc network model with parameters  $D=10$  and avoidance area having  $r=2$ . From these figures it is clear that the greedy selection algorithm is superior all perspectives (number of queries, number of relay nodes and the resulting path length) to the random selection. The greedy and the random algorithms returned negative results in roughly 5% and 1% respectively. In roughly 0.5% of the simulations both the greedy and the random algorithms returned negative results. The intuition behind this result is that the greedy algorithm might focus its selections in a limited area while the random selection have wider selection possibilities and thus increase the likelihood of finding safe paths. However, as can be seen in the results, the overall performance of the greedy algorithm is superior to the random selection algorithm. In all figures we removed the simulations that returned negative results either in the random or the greedy algorithm so that these results will not influence the average case.

From Figure 22-23 we can see that the average number of queries, the number of relays and the average path length using the greedy algorithm is significantly lower in comparison to the random algorithm. However, the difference between the algorithms is reduced with the growth of the optimal path length (the optimal area avoiding path length was computed by a centralized algorithm).

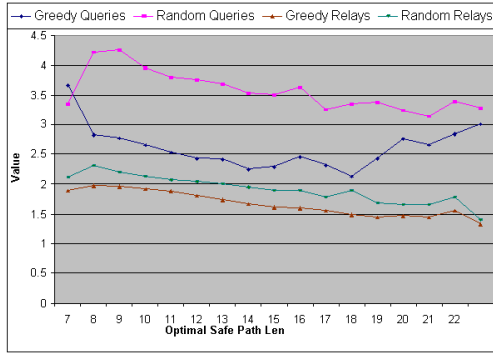


Figure 22 Average number of queries and relays as function of optimal safe path length in ad-hoc network.

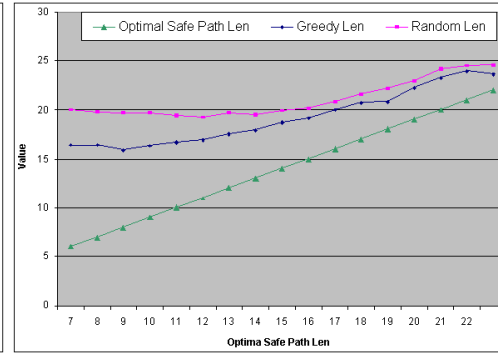


Figure 23 Average path length as function of optimal safe path length in ad-hoc network.

In Figure 24-25 we can see that the number of required queries is increasing with the growth in network degree. This result derives from the fact that with the growth in network average degree the number of nodes in the avoidance area increases. However, the resulting path lengths remain roughly the same, even though the optimal path length is decreasing. This result indicates that the detour around the avoidance area is similar in average regardless of networks' density.

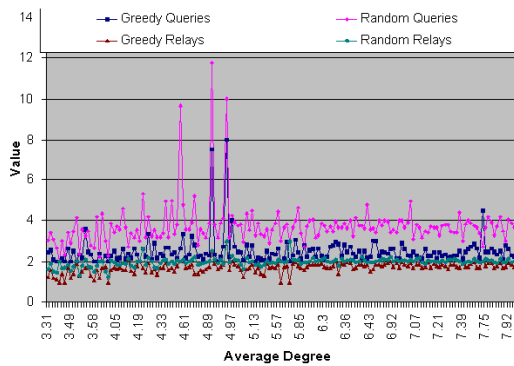


Figure 24 Average number of queries and relays as function of network degree in ad-hoc network.

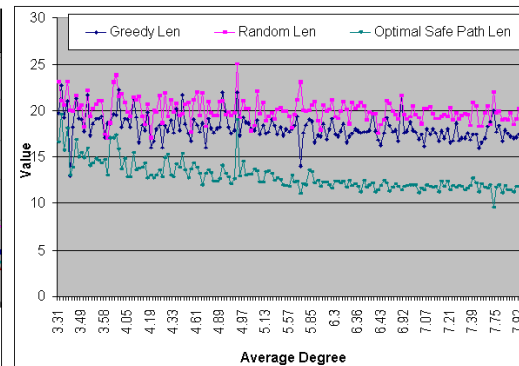


Figure 25 Average path length as function of network degree in ad-hoc network.

Figure 26-28 show that with the growth  $d(s,v)$ , where  $v$  is the area center node, the number of queries and the number of relay nodes decreases. However, the decrease is higher in the case of random reference selection. This is due the fact that when the distance between the source and the area center node (and between the destination and area center node) increases the portion of the nodes in the avoidance area is reduced in comparison to the number of nodes on the shortest. This increases the likelihood of finding area-avoiding paths.

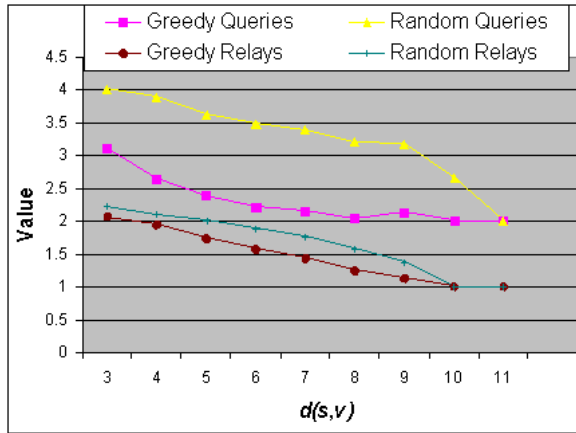


Figure 26 Average number of queries and relays as function of  $d(s,v)$  in ad-hoc network.

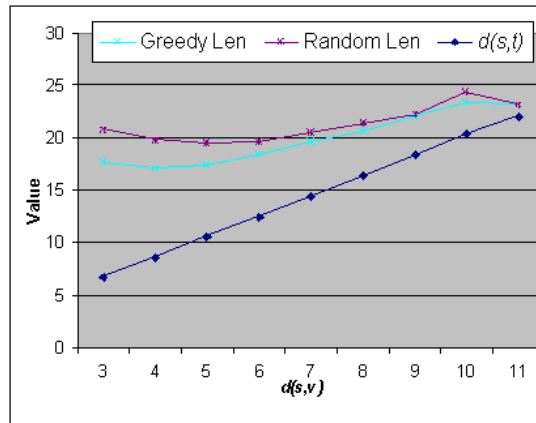


Figure 27 Average path length as function of  $d(s,v)$  in ad-hoc network.

Figure 28-29 depict the simulation results on ad-hoc network model where the plane was of size 250x250 (instead of 100x100 in the previous results). In those networks we used parameter  $r=2$ . The figures depict the results as a function of the optimal path length. As we can see in comparison to figures 22 and 23, the results are roughly similar with superiority of the greedy strategy over the random strategy in all parameters in most cases.

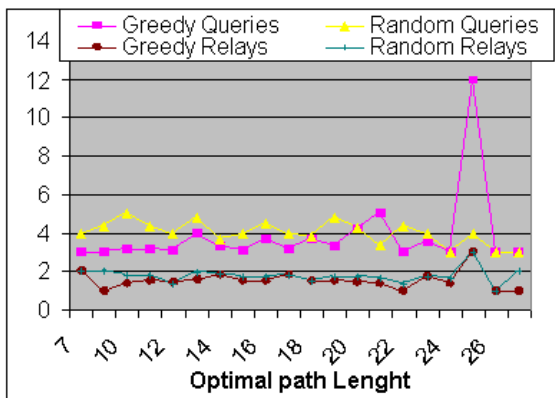


Figure 28 Average number of queries and relays as function of network degree in ad-hoc network in 250x250 plane.

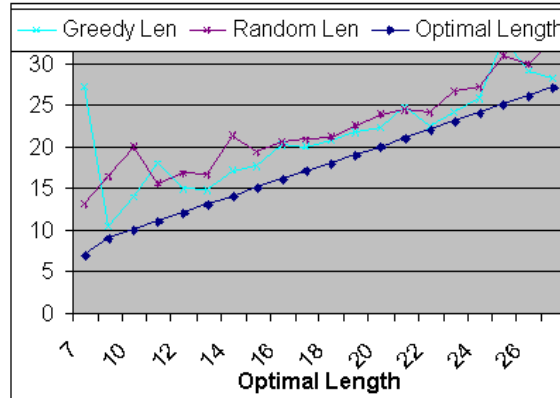
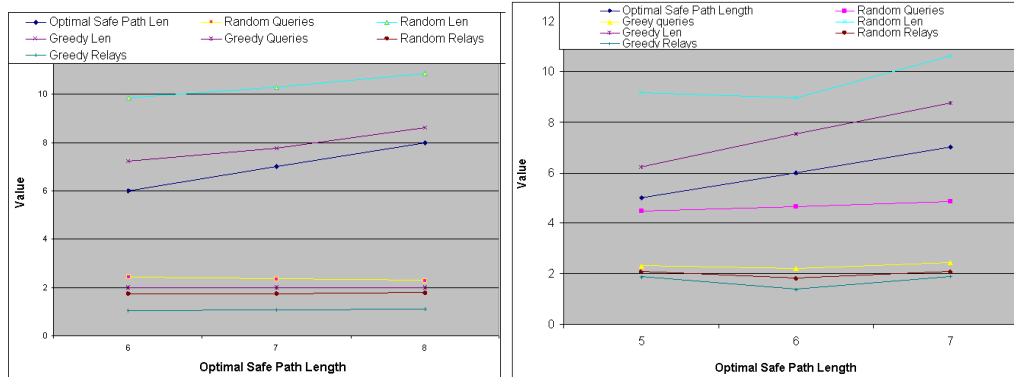


Figure 29 Average path length as function of optimal safe path length in ad-hoc network in 250x250 plane.

Figure 30-31 depict the simulation results on Waxman and Barabassi networks models respectively. In those networks we used parameter  $r=1$ . The network diameter (the highest distance between pairs of nodes in the network) in Waxman networks was between 6 and 8 and in the Barabassi network the diameter was between 5 and 7. Since in the simulation the distance between the  $s$  and  $v$  and the distance between  $t$  and  $v$  was at least  $r+1$ , the minimal optimal length between the source and the destination is 5. In the Waxman model the lowest optimal distance was 6<sup>9</sup>.

<sup>9</sup> In less than 2% of the simulations the optimal safe path length was 5, but the results in these simulations had high variance with respect to number of queries and the resulting path length. Hence, we chose to omit them from the graphs.



**Figure 30 Simulation results on Waxman model based network**

**Figure 31 Simulation results on Barabassi model based network**

One should note that in Section 5.2 we showed that on specific graphs, when the area center node is a reference the random selection algorithm is slightly superior to the greedy reference selection. This result is intuitive since the greedy algorithm selects nodes that are close to the reference node and thus have lower likelihood to help in finding safe paths, once the area center node is already a reference. In the simulations above we used only safe nodes as references that, as expected, result in the superiority of the greedy selection. The intuition is that the greedy selection, which selects safe nodes close to the avoidance area (at least in the first steps), end up with results that are similar to the algorithm that uses area center node as reference.

In addition to these simulations we also tested the influence of the selection of the center node as reference node. The results indicate that the selection of area center node can slightly improve the number of queries but the overall performance is similar to the greedy selection algorithm.

We conclude our simulations with the following insights:

- The greedy selection of area reference nodes is superior in average in all parameters to the random reference node selection.
- The greedy selection results in very low average number of queries in all the scenarios we simulated. The average in the ad-hoc networks was about 2.5 and in the Waxman and Barabassi models the average was 2 (which is also the minimal number of queries under the setups we simulated). This implies that a very small number of queries will suffice in most cases for finding safe paths. This result agrees with the analytic analysis in Section 5.1.
- Selecting the area center node as reference is very beneficial in many cases. For example, in more than 95% of the simulation on Waxman and Barabassi network models, by querying the target and the area center node a safe path was found.
- The average number of relay nodes is very small which means that the overhead, in the packet header, for safe routing that uses weak source routing is very small (in most cases one and two relay nodes will suffice).
- The path length (found by the RestrictedAARouting) are higher than the optimal path lengths. We believe that by additional queries shorter path can be found. We leave this issue for further study.

## 6 Centralized selection of reference nodes

In the previous section we studied the problem of dynamically selecting reference nodes. In this section we turn to a different problem with different objectives. When the entire network topology is available to a central entity, it is of interest to find a minimal set of reference nodes that will guarantee, for every pair of nodes, to find an area-avoiding path, if such a path exists. This set of reference nodes can then be provided to the source node for computing safe paths. We show that the centralized problem is NP-Complete and offer a greedy algorithm addressing the problem.

In the case that a central entity possesses the entire network topology it can find a set of reference nodes and disseminate it to source nodes. Having these reference nodes the sources will be able to autonomously calculate area-avoiding paths. The problem of this central entity is then to find the minimal number of reference nodes such that a safe path can be found for every pair of nodes, if such a path exists. The formal definition of the problem is as follows:

A *reference node cover* of an undirected graph  $G = (V, E)$  is a subset  $V' \subseteq V$  such that for every pair of nodes  $s$  and  $t$  an area-avoiding path will be found, for a given  $AA$ , using the distance vector information of the reference nodes in  $V'$ , if such a path exists. A *minimal-size reference node cover* is a reference node cover of minimal size. The *reference node cover problem* is that of finding a minimal-size reference node cover. It can also be stated as a decision problem:

INSTANCE: A graph  $G(V, E)$ , an avoidance area  $AA$  and an integer  $k$ .

QUESTION: Is there a reference node cover of size  $k$  or less for  $AA$  in  $G$ ?

**Theorem 1:** The reference node cover problem is NP-Complete for and  $K > 0$  (where  $K$  is the number of avoidance areas in  $AA$ ).

**Proof:** It is easy to see that the problem is NP. To show that it is NP-Complete, we will show a polynomial reduction from the *vertex cover* problem, known to be NP-Complete, to the reference node cover problem.

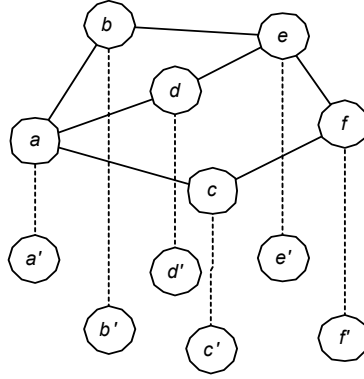
Our reduction will show that given an instance of a vertex cover problem we can construct a graph  $G'(V', E')$  such that one can solve the reference node cover problem, if and only if, it can solve the vertex cover problem. The formal definition of the vertex-cover problem is as follows:

A *vertex cover* of an undirected graph  $G = (V, E)$  is a subset  $V'$  of the vertices of the graph which contains at least one of the two endpoints of each edge:  
 $V' \subseteq V : \forall \{a, b\} \in E : a \in V' \wedge b \in V'$ .

INSTANCE: A graph  $G$  and a positive integer  $k$ .

QUESTION: Is there a vertex cover of size  $k$  or less for  $G$ ?

The construction of  $G'$  is as follows: For every node  $v_i$  in  $G$  add a node  $u_i$  and link  $e(v_i, u_i)$ ,  $c_e=1$ . Let  $u_i$  be a area center node with  $r=0$  in  $AA$ . Clearly, the construction is polynomial. See illustration of the construction in Figure 32. The nodes in  $G$  are  $V=\{a,b,c,d,e,f\}$  and the nodes in  $V'=\{a,b,c,d,e,f,a',b',c',d',e',f'\}$  and  $AA_1=AA(a',0)$ , ...,  $AA_6=AA(f',0)$ .



**Figure 32 Construction of  $G'$  for reference node cover from  $G$ .**

Any solution,  $\hat{V}$ , for the vertex cover problem in  $G$  is also a solution of the reference node cover in  $G'$ . This is trivial because every node in  $V'$ , that is not in an avoidance area (nodes  $v_i$  and not  $u_i$ ) is either in  $\hat{V}$  or adjacent to a node in  $\hat{V}$ . Thus, using Claim 5,  $G'$  is a connected graph of all nodes  $v_i$  (for every link  $v_i-v_j$ : if both  $v_i$  and  $v_j$  are reference nodes, this link is safe, if  $v_i$  is reference and  $v_j$  is not, still the link is safe according to Claim 5). Since  $G'$  is connected, there is a safe path between every pair of nodes which is safe.

Any solution, having  $k$  reference nodes, in  $\tilde{V}$  for the reference node cover in  $G'$  is also a solution with size  $k$  for the vertex cover problem in  $G$ . Note that only the nodes in  $V$  can be chosen for the minimal reference node cover (If  $u_i$  is selected we can always select  $v_i$  instead of it without losing any information). Now we show that for every node in  $V$ , either it is a reference node or it must be an adjacent node of a reference node. Let assume in contradiction that there is a node  $v_q$  in  $V'$  that is not connected to any reference node. In such case, for every reference node,  $v_r$ ,  $d(v_r, v_q) > 1$ . Looking at the closest reference node,  $v_w$ , to  $v_q$ , there is no way to know whether the path is safe or not. The path can be  $v_w-v_r-v_j$  or it might be  $v_w-u_r-v_j$ . Thus, every node  $v_i \in V$  must be in  $\tilde{V}$  or there must be  $v_j$  such that there is edge  $e(v_i, v_j)$  and  $v_j \in V$ . ■

Any approximation to the vertex cover problem can be used for the reference node cover. This can be done by removing all nodes in  $AA$  from  $G$  and using the approximation. Of course, the result might be far from optimal for the reference cover problem. To address this problem the following simple greedy algorithm can be considered:

#### **GreedyReferenceNodeCover()**

1.  $\tilde{V} \leftarrow \{\phi\}$ ,  $C \leftarrow \{\phi\}$

2. Find for every node  $v_j$  the set of nodes  $N(v_j)$  that can be reached from it via area avoiding path (by removing the area avoiding nodes from  $G$ ).
3. Let  $v_i$  be the node for which  $N(v_i)$  is maximal.  $\tilde{V} \leftarrow v_i$  ( $\tilde{V}$  is the set chosen reference nodes)
4.  $C \leftarrow$  Find nodes that can be reached from  $v_i$  via area avoiding path using Claim 6. ( $C$  is the set of nodes that are reachable via safe routing from  $v_i$  up to this step)
5. While  $C \neq V$  //while there are unreachable nodes
  - a. Find  $N(v_j) \forall v_j : v_j \in C$ , where  $N(v_j)$  contains the set of nodes such that  $v_i \notin C$  and can be reached from  $v_j$  via a safe path using references in  $\tilde{V}$  (by validating Claims 6 and 8).
  - b.  $\tilde{V} \leftarrow v_j$ : if  $v_i \notin C$  and  $N(v_j)$  possessing maximal cardinality (adds maximal number of nodes to the set of reachable nodes via safe path)

The performance of this algorithm is left for further study.

## 7 Conclusions and Summary

We studied the problem of area avoidance routing whose goal is to address security, QoS and other vulnerabilities inherited in the traditional shortest path routing. We addressed this problem in the framework of distance-vector networks that were formed originally to yield shortest-path routing *only*. We showed that there is no algorithm that can guarantee area avoidance routing by querying less than  $|V|/2-2$  nodes for their distance vector information.

We introduced a heuristic mechanism in which the source queries other nodes, we refer to as reference nodes, in the network for their distance vector information. Using this information we showed the conditions that guarantee safe routing between pairs of nodes. Using these conditions we proposed algorithms for finding area-avoiding paths. Though the complexity of a greedy exhaustive algorithm is very high we showed an algorithm with significantly lower complexity that guarantees finding area-avoiding path if the greedy algorithm finds one.

Using these algorithms we evaluated two heuristics for selecting reference nodes. We analyzed the performance of these algorithms on structured graphs such as Grid-4 and Grid-8. Through extensive simulation on random graphs, we showed that in most cases querying a very small number of nodes allows safe routing and the per-packet overhead (relay nodes in weak source routing header) is very small as well. These results agree with the analytic analysis.

## 8 References

- [1] S. Bak, and J. Cobb, "Randomized Distance-Vector Routing Protocol", Proceedings of the 1999 ACM symposium on applied computing: 1999.
- [2] A. Bagchi, A. Chaudhary, M. T. Goodrich, S. Xu, "Constructing Disjoint Paths for Secure Communication", DISC 2003: 181-195.
- [3] S. Bahk , M.El Zarki, "Dynamic Multi-path Routing and How it Compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks", in Proceedings of ACM SIGCOMM '92, Aug. 1992.
- [4] S. Bohacek, J. P. Hespanha, K. Obraczka, J. Lee, C. Lim, "Enhancing Security Via Stochastic Routing", In Proc. of the 11th IEEE ICCCN, May 2002.
- [5] S. Capkun, M. Hamdi, J. Hubaux, "GPS-free positioning in mobile ad hoc networks", ICSC, p. 3481-3490, 2001.
- [6] BRITE: Boston university Representative Internet Topology generator, [www.cs.bu.edu/brite/](http://www.cs.bu.edu/brite/)
- [7] M.Chay, S. Moon, Chong-Dae Parkz, and Aman Shaikh, "Placing Relay Nodes for Intra-Domain Path Diversity", IEEE Infocom, April 2006.
- [8] J. Chen, P. Druschel, D. Subramanian, "An Efficient Multipath Forwarding Method", INFOCOM 1998: 1418-1425.
- [9] R. Cohen, G. Nakibli, "On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing", IEEE Infocom 2004.
- [10] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang. "IDMaps: A Global Internet Host Distance Estimation Service", *IEEE/ACM Transactions on Networking*, 9(5):525--540, October 2001.
- [11] Y. Guo, F. Kuipers, P. Van Mieghem, "Link Disjoint Paths for reliable QoS", International Journal of Communication Systems 779-798 NOV 2003.
- [12] J. P. Hespanha and S. Bohacek, "Preliminary results in routing games," in Proc. Of the 2001 American Control Conference, June, 2001.
- [13] B. Karp, H.T Kung, "Greedy Perimeter Stateless Routing for Wireless Networks", in Proceedings MobiCom 2000, Boston, MA, August, 2000, pp. 243-254.
- [14] P. P. C. Lee, Vishal Misra and D. Rubenstein, "Distributed Algorithms for Secure Multipath Routing", IEEE Infocom 2005.
- [15] H. Levy and H. Zlatokrilov, "The Effect of Packet Dispersion on Voice Applications in IP Networks", IEEE/ACM Transactions on Networking, April 2006, Vol. 14.
- [16] H. Lim, J. C. Hau, C. Hoi, "Constructing Internet Coordinate System Based On Delay Measurement", Internet Measurement Conference 2003.
- [17] M. Mauve, J. Widmer, H. Hartenstein, "A Survey on Position-Based Routing in Mobile Ad-Hoc Networks," In IEEE Network. Vol. 15 (2001), Nr. 6, S. 30-39.
- [18] T. Nguyen, A.Zakhor, "Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks", IEEE Infocom 2003.
- [19] A. Orda, A. Spronston, "Efficient Algorithms for Computing Disjoint QoS Paths", IEEE Infocom 2004 Hong-Kong.
- [20] C. E. Perkins, E. M. Belding-Royer, S. Das. "Ad Hoc On Demand Distance Vector (AODV) Routing." *IETF RFC 3561*
- [21] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance", J. Assoc. Comput. Mach., vol. 36, no. 2, pp. 335--348, Apr. 1989.
- [22] S. D. Servetto and G. Barrenechea, "Constrained Random walks on random graphs: Routing algorithms for large scale wireless sensor networks", In Proc 1st ACM Int. Workshop on Wireless Sensor Networks and Applications (WSNA), Sept. 2002.
- [23] D. Sidhuand, R. Nairand, S. Abdallah, "Finding Disjoint Paths in Networks", In Proc. of ACM SIGCOMM' 91, 1991.
- [24] Villamizar, "OSPF optimized multi-path (OSPF-OMP)", draft-ietf-ospf-omp-03.
- [25] J. Wang, "Load Balancing in Hop-by-Hop Routing with and without traffic splitting", Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, October, 2003.
- [26] D. Xu, Y. Chen, Y. Xiong, C. Qiao , X. He, "On Finding Disjoint Paths in Single and Dual Link Cost Network", IEEE Infocom, Mar. 2004, HK.
- [27] H. Zlatokrilov, H. Levy, "Privacy and Reliability by Dispersive Routing", IEEE Infocom 2006, Barcelona.

- [28]H. Zlatokrilov, H. Levy, “Navigating in Distance Vector Spaces and Its Use for Node Avoiding Routing”, Technical Report, available at: <http://www.cs.tau.ac.il/~zlatokri/>.
- [29]H. Zlatokrilov, H. Levy, “Area Avoidance Routing in Distance-Vector Networks”, Technical Report, available at: <http://www.cs.tau.ac.il/~zlatokri/>.
- [30]H. Zlatokrilov, H.Levy, “Navigation in Distance Vector Spaces and Its Use for Node Avoidance Routing”, IEEE Infocom 2007.