

Navigation in Distance Vector Spaces and Its Use for Node Avoidance Routing

Haim Zlatokrilov, Hanoch Levy

Tel-Aviv University, Israel

Abstract—Traditional network routing uses the single (shortest) path paradigm. This paradigm exposes sessions to various attacks along this path, such as eavesdropping, DoS attacks etc. As a result, certain nodes or network regions may form security threats and it is desired to consider node routing schemes which avoid them. The task of node avoidance routing is particularly challenging in distance-vector networks, where only shortest- distance information is available to the nodes. We address this problem by proposing a new routing paradigm in which the forwarding mechanism exploits the distance-vector information towards several nodes and utilizes it to forward network traffic on non-shortest paths routes; in particular on node-avoiding routes aiming at bypassing security-suspected nodes. We study this paradigm, propose a routing algorithm based on it and establish their properties. Extensive evaluation of the algorithm in general situations is conducted via simulation.

Index Terms—distance vector, multipath, routing, node avoidance, beacon, virtual position.

I. INTRODUCTION

Network security is a major concern both in static and ad-hoc networks. One possible attack is to hijack routers in the network. Controlling routers provides the attackers with heavy machinery for inflicting significant damage on all sessions passing through them. By controlling routers, attackers may conduct various attacks such as: DoS (for all or some specific sessions routed through them), eavesdropping attacks, man-in-the-middle attacks etc.

We study a model in which the sender has some intelligence information about *suspicious* nodes in the network. By the term *suspicious* we refer to nodes that are suspected in jeopardizing security. For example, nodes having low security classification due to physical location, nodes that might have been captured by

hostile parties, etc¹. We believe that such information may be provided by network monitoring and analysis utilities or by any other intelligence sources. Accounting for these situations it is of interest to route sessions along routes which avoid such nodes. We focus our discussion on two specific objectives of interest: *shortest-path-avoidance* (addresses the scenario where there is general information about a suspicious node along the shortest path) and *area-avoidance* (addresses the scenario where a specific node and its adjacent nodes are to be avoided).

We consider systems where the source does not have the ability to eliminate suspicious nodes from the network or change the routing protocol, but the forwarding mechanism in all nodes can be extended. In order to take advantage of the intelligence information and reduce the threats, the source node needs to direct the packets along suspicious-node avoiding paths. This necessity complies with a recent trend in routing research to allow hosts to choose routes by themselves to avoid the inefficiencies caused by the shortest path network-layer routing. Thus, we focus on the problem of adding packet-forwarding capabilities to an existing routing protocol that will enable routing via node avoidance paths.

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 suspicious nodes, the source can use a centralized algorithm to calculate node-avoiding paths. The source can then direct the packets using strict source routing. Centralized algorithms for calculating node-avoiding paths are relatively simple (e.g. remove the suspicious nodes from the graph and compute the shortest path) and do not pose major difficulty. Thus, in link-state routing protocols, the network layer provides all the necessary mechanisms for carrying out node-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 Routing Information Protocol (RIP) and Destination-Sequenced Distance-Vector (DSDV) use the property that for every pair of nodes s and t and an intermediate node v (along the 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

¹ Suspicious nodes may be nodes with particular characteristics to be avoided in other contexts, for example nodes that unintentionally reduce QoS due to load or nodes suffering low energy in ad-hoc networks.

through other paths. In particular, as we will show, no algorithm that is based on the distance-vector information only can guarantee routing through node-avoiding paths, even if such paths exist, for any *arbitrary network*. In fact, we show 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 node-avoiding algorithms for distance vector networks.

Our focus in this work is on introducing a new forwarding paradigm designed for distance-vector networks. We study how this paradigm can utilize the full (or part of) distance-vector information for routing on non-shortest paths. In particular we deal with routing through suspicious-node avoiding paths. The basic idea is that the full (or partial) distance vector information available to a node, reflecting its distance from the other nodes, can be used as an approximate indication of the “location” of the node within the network. This is in analogy to latitude-longitude coordinates used in geographical navigation or a set of distances from satellite used in Global Positioning Systems (GPS) for the same purpose. Thus, the distances from several nodes can serve as a “system of coordinates”. Based on this coordinate system, one can now attempt to navigate a packet through a series of locations (virtual points), all expressed in this coordinate system. Our work studies this generalized routing concept.

Our work starts (Section II) with presenting the model and studying basic properties of node-avoidance routing in distance-vector networks. Next (Section III) we present the paradigm for distance-vector navigation and we propose the Virtual-Positioning-Source-Routing (*VPSR*) algorithm that is based on this paradigm. We analyze the properties of the VPSR and show the bounds on path length in uniform and non-uniform networks, analyze various properties and observations etc. Next (Section IV) we evaluate the quality of the VPSR algorithm on a general setting by investigating a large number of cases via simulation. Lastly, concluding remarks are given in Section V.

A. Related Work

In this work we present a routing algorithm for routing via paths other than the shortest path. The benefits of such routing scheme were widely discussed in the area of multi-path routing in the contexts of load-balancing and efficiency [8][23], reliability [6][9][17], QoS enhancement [14][18] [22], security and DoS [2][13][19][25]. Routing via random paths for avoiding enemies was studied in [4] and [10].

The routing methodology we study is based on the concept of using distances from various nodes as a coordinate system for navigation. Similar concept for presenting the location of Internet hosts was studied in [15]. Routing according to geographical location was studied in the area of geographical routing (also

referred to as *position based routing*) [12][16], where the physical locations are used as a coordinate system for forwarding.

One of our objectives is to route packets over deterministic paths. That is, packets carrying equal routing information will be forwarded to the same next hop. This will preserve static routes from the point of view of the session. Different approaches that randomly choose the next hop, were studied in [20]. ECMP (Equal Cost Multipath Protocol) [11] is also a technique that can be used either randomly or deterministically for routing along multiple paths of equal cost.

A centralized approach for computing non-shortest-paths with various objectives were studied in [2][18] and [24]. In distance vector networks, a random approach for selecting intermediate nodes together with source routing was introduced in [1][3] and [8]. A distributed distance vector algorithm for multipath routing that modifies the topology exchange mechanism were studied in [21] and [7]. In contrast, we focus on routing in distance vector networks by modifying the forwarding algorithm without changing the topology exchange protocol.

II. MODEL AND ASSUMPTIONS

We model the network as an undirected graph $G(V,E)$. The nodes and the edges are denoted v_1, \dots, v_n and e_1, \dots, e_m , respectively, where $n=|V|$, $m=|E|$. Each edge e is associated with a length (cost) parameter $c_e > 0$. For the sack of simplicity we assume that the network is stable and routing changes are relatively infrequent. That is, route changes are reflected in the routing information immediately and simultaneously in all nodes.

The routing algorithm in the network is distance-vector based and the topology exchange algorithm cannot be changed. That is, each node v_k receives from its neighbor v_i the vector of its distances to all destinations, denoted by *distance vector* of v_i , $T(v_i)$. $T(v_i)$ contains the minimal length to reach each of the nodes v_j in the network, namely, $T(v_i) = \{d(i,1), d(i,2), \dots, d(i,|V|)\}$ where $d(i,j)$ denotes the shortest distance from node v_i to v_j ; due to the symmetry of the graph $d(i,j) = d(j,i)$. Typically, for shortest path routing, v_k uses this information to construct a routing table containing the shortest path and the next hop to each destination. We will assume that, in addition to constructing and storing this routing table, v_k will store the information received from its neighbors separately (the additional storage required is $g_k|V|$, where g_k is the degree of v_k).

Path p (also referred to as route) is a sequence of nodes (v_1, \dots, v_k) and links (e_1, \dots, e_{k-1}) . If $v_k(e_j)$ is in path p we say that $v_k \in p(e_j \in p)$. 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 routing objective in this model is to increase the likelihood of routing packets via node-avoiding length-limited paths (finite number of routing loops is allowed). Thus, even if the security is jeopardized, packets must not be dropped. We also require *deterministic routing*. That is, every node will consistently choose the same next-hop for packets carrying the same routing information². Such packets will always follow the same path.

Avoidance paths can be used for various objectives and applications, of which we focus on this work on the following:

- *Shortest-path-avoiding path* - Routing via paths that are as disjoint as possible from the shortest path. That is, let sp denotes the set of nodes on the shortest path from s to t ($sp=(s=v_1, \dots, v_k=t)$), then the objective of the routing algorithm is to route on path p so that $(p \cap sp) - \{s, t\}$ is minimized. An illustration of such a path is depicted in Figure 1-(a) where the shortest-path-avoiding path is marked by the dotted line.
- *Area-avoiding path* - Let v_i be a node (to be denoted the *center*) and let x denote distance between nodes. Let $S(v_i, x)$ denote the set of nodes at radius smaller than or equal to x from v_i , that is $S(v_i, x) = \{v_j \mid d(i, j) \leq x\}$. The objective is to route via path p such that $p \cap S$ is minimized. One should note that in the case $x=0$, the objective is to avoid passing through v_i itself. An illustration of an area-avoiding path is depicted in Figure 1-(b) where node v_i is the center node and the area contains nodes in distance $x=1$ from v_i . The area-avoiding path is plotted by the dotted line.

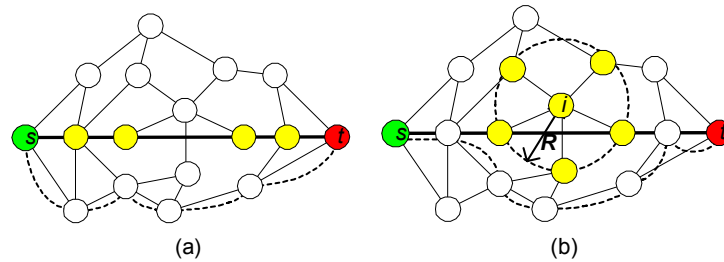


Figure 1: (a) A *Shortest-path-avoiding path*, (b) *Area-avoiding path*

² This requirement comes to maintain stable routing as long as the network remains stable. Such feature is very important for tracing and monitoring paths and enables to query the nodes along the path for other properties.

A. Properties of Node Avoidance Routing in Distance Vector networks

Our first result, presented in the theorem below, is a “negative” one, suggesting that for an *arbitrary network* the vector-distance information is not sufficient for *guaranteeing* that a node-avoidance path can be found.

Definition 1: Let $k \geq 0$. A forwarding algorithm is called *k-local distance-vector forwarding* if at every node v_i it is allowed to use in its decision only the distance vector of v_i and of all nodes v_j such that $d(i, j) \leq k$.

Remark 1: Note that the shortest path routing is a *1-local distance-vector forwarding* algorithm.

Theorem 1: No deterministic memoryless *1-local distance-vector forwarding* algorithm can guarantee, for every *arbitrary network*, routing over a path that avoids a given node v_i , even if such path exists.

Proof: Consider Figure 2-(a) and (b) where s aims at routing to t while avoiding node v_i . In both networks s receives the following distance vector information from a and b : $T(v_a) = \{d(a, b) = 2, d(a, c) = 3, d(a, d) = 1, d(a, i) = 1, d(a, t) = 2\}$ and $T(v_b) = \{d(b, a) = 2, d(b, c) = 1, d(b, d) = 3, d(b, i) = 1, d(b, t) = 2\}$, and, also, the distance-vector of s is identical in both networks. In case there was a deterministic algorithm that could guarantee routing via path avoiding v_i it would choose node a as the next hop for the network in Figure 2-(a), resulting in path $s \rightarrow a \rightarrow d \rightarrow t$. Since the routing information is similar in both networks, a deterministic algorithm should choose node a as the next hop in the network depicted in Figure 2-(b) as well. But, this selection results in not avoiding node i , namely resulting in path $s \rightarrow a \rightarrow i \rightarrow t$.

Remark 2: of course, if the algorithm is not memoryless, it can conduct a distributed BSF and reach destination while avoiding v_i , regardless of the distance vector information (At the expense of very long paths and high complexity) ■

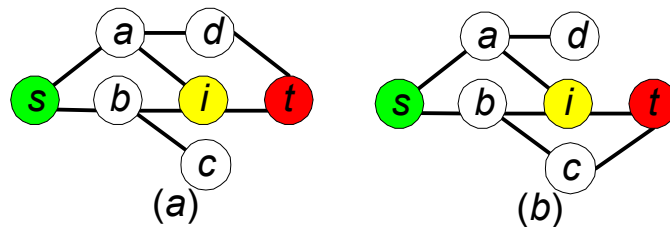


Figure 2: Networks with similar distance vector information in s .

Remark 3: It is easy to see that Theorem 1 holds even for *k-local distance-vector forwarding* algorithm with arbitrary $k \geq 0$. The proof is by taking the network in Figure 2-(a) and (b) and adding k nodes between nodes a and i , and b and i .

Clearly, avoiding a specific node is a particular case of routing via shortest-path-avoiding or area-avoiding paths. In the following section we present a routing methodology that addresses these routing objectives, based on vector-distance information. Though no deterministic algorithm can guarantee solving these problems for *arbitrary networks*, it can still be practically effective due to the high connectivity of common networks.

III. A METHODOLOGY FOR DISTANCE VECTOR NAVIGATION AND ITS USE IN VIRTUAL-POSITIONING-SOURCE-ROUTING

A. Distance Vector Positioning and Beacon Assisted Navigation

In shortest path routing, each node chooses a neighbor that leads towards the destination via the shortest path. It is done by choosing the next-hop through which the shortest path to the destination is minimized. This complies well with the distance vector information available in the nodes. In order to route via other paths, a different forwarding approach must be used.

We introduce a general routing paradigm that utilizes the distance-vector information (or part of it) for general routing purposes. The paradigm views the distance-vector of a node as an indicator of its relative position in the network and makes use of this indicator (as well as the indicators of the node neighbors) to forward the packets according to the routing objectives.

Specifically, we use the term *beacon* to denote a specific node in the network used as reference point for positioning. Let $\{b_1, \dots, b_B\} \subseteq V$ be a set of beacons, where B is the number of selected beacons. For *node* v_i we define its *beacon position*, β^i , as a vector of its shortest distances to the selected beacons, $\beta^i = (d(v_i, b_1), \dots, d(v_i, b_B))$. The beacon position can be thought of as a “coordinate system” to indicate the position of a node on the network. This is in analogy to latitude-longitudes coordinates used in geographical navigation or a set of distances from satellites used Global Positioning Systems (GPS). The beacon-positions of a node and its neighbors can be used in making forwarding decisions. For example, the well known shortest-path routing is a specific case of this paradigm in which the destination node serves as the sole beacon and the forwarding algorithm forwards to the lowest position neighbor. This coordinate system is in fact a space that is formed by the distance vector and the projection on beacon coordinates.

Many routing methodologies can be based on this routing paradigm. For example, a simple forwarding rule based on this paradigm can be: “Forward a packet to a neighbor that is closer to node t and as far as

possible from node v_j ". Such routing methodology looks reasonable for the purpose area-avoidance routing in some cases, while might be useless for shortest-path avoidance (if the nodes along the shortest path are unknown). Our focus in this work will be on a methodology called *Virtual-Position-Source-Routing* (VPSR) while other methodologies that utilize beacon position are left for further study.

B. Virtual Positions – the Basis of the Algorithm

Given a beacon set of size B , the term *virtual position* denotes an arbitrary vector of dimension B , representing a set of distances to the beacons. The term *virtual* is used since for an arbitrary such vector (x_1, \dots, x_B) there may not exist in the network a node whose beacon position equals (x_1, \dots, x_B) .

For each node we define the metric of *distance from the virtual position*. If the beacon position of node v_i is given by vector β^i and the virtual position is given by a vector vp , then the distance of v_i from the virtual position is given by $\|\beta^i - vp\|$ where several distance metrics can be used, such as presented in [15]. For Example, using two beacons s and t and the L^2 -norm as the distance metric, the distance of node v_i from the virtual position $vp=(vp_s, vp_t)$ is: $\|\beta^i - vp\|_2 = \sqrt{(d(i,s) - vp_s)^2 + (d(i,t) - vp_t)^2}$ and if the L^1 -norm is used it is $\|\beta^i - vp\|_1 = |d(i,s) - vp_s| + |d(i,t) - vp_t|$. In this work we will focus on distances in L^2 -norm.

The *Virtual-Position-Source-Routing* (VPSR) forwarding algorithm, proposed here and described in detail in the next sub-section, aims at trying to route the packet through a finite sequence of virtual positions.

C. The Principles of Virtual-Position-Source-Routing

The idea behind the VPSR is that the source “draws” an imaginary (*virtual*) path consisting of a sequence of virtual positions (vp^1, \dots, vp^k) . Each packet carries all the required routing information, such as the forwarding algorithm type, the Ids of the beacons and the sequence of *vps* in the header. The objective of the forwarding algorithm is to forward packets on a path that is as close as possible to the virtual path.

A node v_r that receives a packet aims at forwarding it to its neighbor v_n that is the closest (among the neighbors) to the current target $vp^i, 1 \leq i \leq k$. In case there are several equally good neighbors one of them is selected using a tie breaking rule, such as lowest node address. If there is no neighbor closer than v_r to vp^i , then v_r is a local minimum. In that case, v_r changes the current target to vp^{i+1} . In this case, if $i=k$ (last *vp*) and the packet did not reach the destination, the shortest path to the destination is taken. In some scenarios this last stage can be avoided, as will be later discussed. The detailed description of the VPSR routing algorithm can be found in Appendix A.

Technical Report

The algorithm resembles some geographic routing methodologies (e.g. the methodology of GPS) where distances from two or more reference-nodes are used to derive the relative location of nodes in the network. A pair of distances describes two possible physical positions that are at the intersection points of two circles (see Figure 3). Using a sequence of pairs of such positions one could describe a path (two parallel paths which are equally good for our purposes) the algorithm strives to follow.

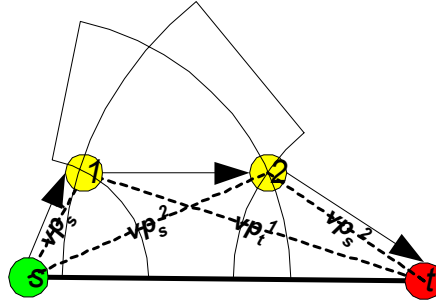


Figure 3: A route described by distances from beacons s and t using the sequence of positions: $vp^1 = (vp_s^1, vp_t^1)$, $vp^2 = (vp_s^2, vp_t^2)$.

For illustration, consider a scenario in which VPSR uses nodes s and t as beacons. One can plot a projection of the network on a plane in which the nodes are located based on their distances to the beacons. We will use this plane to illustrate the result of VPSR. Figure 5 depicts the projection of the network depicted in Figure 4. In Figure 5 the x -axis and the y -axis denote the distances of a node to s and t respectively. In that example, VPSR is implemented using a single virtual position (4,4) in the sequence. The planned path formed by this virtual position is drawn in a dashed line (from s to (4,4) and from that point to t via the shortest path). The actual path traversed by the packet is $s \rightarrow 5 \rightarrow 4 \rightarrow 15 \rightarrow 13 \rightarrow 2 \rightarrow 3 \rightarrow t$ (blue dashed line).

Note that if two beacons are used, the plane is bounded by a three-sided rectangle resulting from the triangular inequality. That is, for any node: $d(s,i) + d(i,t) \geq d(s,t)$, $d(s,i) \leq d(i,t) + d(s,t)$ and $d(i,t) \leq d(s,i) + d(s,t)$.

One should note that the actual route differs from the planned imaginary path described by the set of virtual positions. The reason is that nodes have limited number of neighbors and in some cases the packet cannot be routed on the imaginary path. The intuition is that the network projected on the plane in Figure 5 is not dense, which leaves areas uncovered. Clearly, as density increases on the projected plane (which is a function of the network structure) the likelihood that the actual route will resemble the planned route increases.

Technical Report

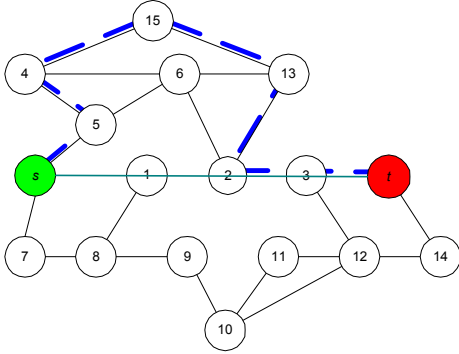


Figure 4: Network with uniform link lengths

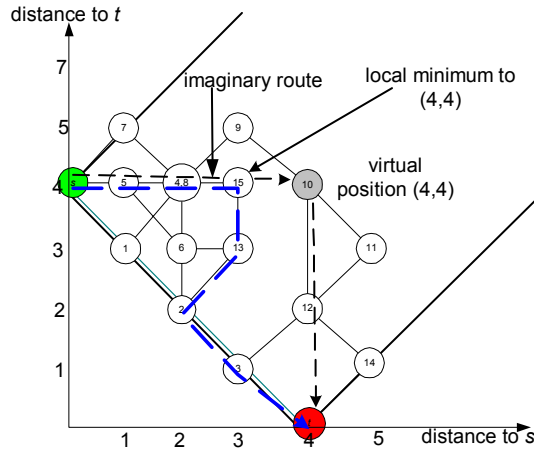


Figure 5: Projection of the network

VPSR can select the number of beacons used. In this work we focus on VPSR using two beacons. The properties of VPSR using more than two beacons are left for further study.

Remark 4: The VPSR uses a technique that reminds the weak-source-routing. Nonetheless, since the routing information in distance vector networks is not sufficient for choosing *actual* intermediate nodes for node-avoidance routing, VPSR deals with this by using *virtual* intermediate locations, through which a packet should traverse.

Using the VPSR routing methodology we will focus on two algorithms: the first addresses shortest-path avoiding routing and the second addresses the area-avoidance objective.

For shortest-path avoidance, the source uses itself and t as beacons and calculates a set of virtual positions describing an arc (or any other geometric form) detouring the shortest path. The beacons and the set of virtual locations are set in the packet header. See implementation details of the VPSR algorithm for shortest path avoidance (denoted by *ShortestPathAvoidanceVPSR*) in Appendix A.

The algorithm for area avoidance directs the packets through the shortest path until a node whose distance to the avoidance area center node is smaller than a certain (configurable) distance. When reaching such node, the VPSR method, using the center node x and t as beacons, is activated. The sequence of vps is determined by the source. We used a method of choosing vps that form a triangular route detouring the avoidance area. Clearly, many other heuristics for choosing the vps can be used. See implementation details of the algorithm (denoted by *AreaAvoidanceVPSR*) in Appendix A .

In the sequel we analyze the basic properties of VPSR and its performance. We first analyze it in uniform link cost networks and then turn to general link cost networks.

D. VPSR in Uniform Link Cost Networks

In this section we assume that the cost of each link is 1.

Lemma 1: Consider a packet that is forwarded from node v_i , whose beacon position is $\beta^i=(d(i,m),d(i,n))$ (where v_m and v_n are arbitrary beacons), to virtual position (vp_m, vp_n) . Then the packet will traverse each node on the path only once.

Proof: Since in every step (moving from a node to its neighbor) of the algorithm it is required that the distance to the vp reduces, the same node cannot be traversed twice. ■

While a single step of VPSR (from a node to vp) is loop-free, the use of multiple vps does not guarantee loop-free navigation. Nonetheless, we will show in Theorem 4 an upper bound for the path length, when an arbitrary sequence of vps is used. We start by establishing a bound on the length of the path traversed between a node and a vp . One should note that using the L^1 -norm, the path length between node v_i and a vp , L , is $L \leq \| \beta^i - vp \|_1$ since on each step the difference must be reduced by at least one. Though it may look like the upper bound using L^2 -norm might be high since there are many possible routing options, in fact it is low, as shown in the following theorem:

Theorem 2: Consider a packet that is forwarded from node v_i , whose beacon position is $\beta^i=(d(i,m),d(i,n))$ (where v_m and v_n are arbitrary beacons), to virtual position $vp=(vp_m, vp_n)$. Then the length of the path, L , traversed by the packet until reaching a local minimum is bounded by: $L \leq 2 \max(u) + \min(u) - 1$, where $u = (|d(i,m) - vp_m|, |d(i,n) - vp_n|)$.

Proof: The proof is by examining all possible ways to move from one node to another on the way from node v_i to a vp and showing the upper bound on the number of steps traversed, which equals the number of traversed nodes. Note that in case the algorithm stops in local minimum before reaching a node whose beacon position in is not similar to the vp 's position, this path is shorter than the potential path that reaches a node whose beacon position is vp .

Assume WLG that $\beta^i \geq vp$ ($d(i,m) \geq vp_m$ and $d(i,n) \geq vp_n$); all other cases can be treated in a symmetric way. Let $\delta_m^i = d(i,m) - vp_m$ $\delta_n^i = d(i,n) - vp_n$. Also assume WLG that $\delta_m^i \geq \delta_n^i$ (the inverse case is symmetric).

In order for VPSR to advance from v_i to a next hop v_j , the distance to the vp must reduce. Hence, one of the following conditions must hold in every step along the path (A plane with a vp , node v_i and possible next hops is illustrated in Figure 6):

- a) $\delta_m^i = \delta_m^j$ and $\delta_n^i > \delta_n^j$ (node a , this case does not exist if $\delta_n^i = 0$)

Technical Report

b) $\delta_m^i > \delta_m^j$ and $\delta_n^i > \delta_n^j$ (node *b*)

c) $\delta_m^i > \delta_m^j$ and $\delta_n^i = \delta_n^j$ (node *c*)

d) $\delta_m^i > \delta_m^j$ and $\delta_n^i < \delta_n^j$ (node *d*, this case does not exist if $\delta_m^i = \delta_n^i$).

Observation 1: If in all nodes along the path the forwarding is done using only one of the first three conditions, the path length is bounded by $\delta_m^i + \delta_n^i$.

This observation derives from the fact that in every step at least δ_m^i or δ_n^i decreases by one unit, their values never increase and their final values are 0.

Observation 2: The boundary formed by the horizontal line ($\delta_n^i = 0$, line *w* in Figure 6) and the diagonal line ($\delta_m^i = \delta_n^i$, line *z* in Figure 6) are never crossed (For the sake of analysis, we refer to each step in the upper quadrant as if it happened in the lower quadrant).

Observation 3: Forwarding due to the last condition may repeat at most in $\delta_m^i - 1$ nodes.

The reasons are: 1) In each such step δ_m^i decreases by one unit, 2) Observation 2, and 3) The “lowest point” that can be reached by such step is ($\delta_m^i = 1, \delta_n^i = 1$) (the point $\delta_m^i = 0, \delta_n^i = 0$ cannot be reached by such step due to Observation 2).

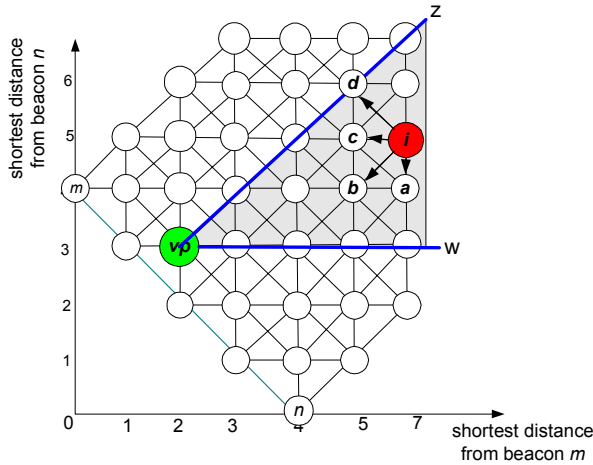


Figure 6: Possible next hops on from node *i* to a *vp*.

Putting Observation 1 and Observation 3 together leads to an upper bound on the number of steps $L \leq 2\delta_m^i + \delta_n^i - 1$. Thus, for the general scenario $L \leq 2 \max(u) + \min(u) - 1$. ■

Observation 4: A lower bound on the path length is $\max(u)$.

The upper bound in Theorem 2 is tight, as demonstrated next: In Figure 7 we illustrate a network in which the path length from node *q* (at beacon position $\beta=(3,3)$) to the *vp* (at position $vp=(5,5)$), using *s* and

Technical Report

t as beacons is: $L=2(5-3)+(5-3)-1=5$. As can be seen, the shortest path can be of length 2 ($a \rightarrow i \rightarrow vp$) but since no such path exists, the actual length traversed by the packet is 5, $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow vp$. This example can be extended for any length.

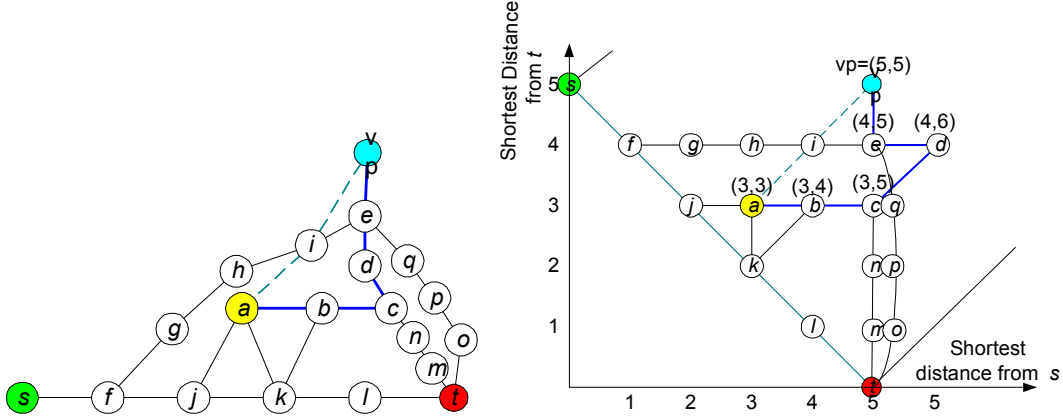


Figure 7: Example for upper bound on path length

Note that the distance between beacons affects the path length. If the beacons are too close to each other, the potential of moving away from the shortest path between the node and a vp reduces. Of course, this will reduce the chance of routing via the desired paths. For example, if a single beacon is used (or two beacons very close to each other) the bound is the length of the shortest path between the node and the vp .

Theorem 3: Consider a uniform link cost network, where v_a (an arbitrary node) and t serve as beacons. If the last virtual position in the sequence of vp s is $vp=(d(a,t),0)$, the packet will reach t via the shortest path.

Proof: It is enough to show that for any node there is a closer neighbor to the vp , which is also the beacon position of t . Showing this property it is clear that the packet will carry on advancing until reaching t .

Starting from an arbitrary node v_i , we will show that there must be at least one neighbor v_j having $\| \beta^j - vp \| < \| \beta^i - vp \|$.

In case $|d(i,a) - d(a,t)| < d(i,t)$, there is a neighbor v_j on the shortest path from v_i to t in which $d(j,t) = d(i,t) - 1$. Since for v_j $\| \beta^j - vp \| < \| \beta^i - vp \|$, the packet will be routed to this node.

If $|d(i,a) - d(a,t)| > d(i,t)$ there are two possible cases:

1. If $d(i,a) > d(a,t)$ there is a neighbor v_j on the shortest path from v_i to a in which $d(j,a) = d(i,a) - 1$, Since for this neighbor $\| \beta^j - vp \| < \| \beta^i - vp \|$, the packet will be routed to v_j . Note that the packet is then routed through the shortest path towards the destination.

Technical Report

2. The case where $d(i,a) < d(a,t)$ is impossible since for every node we have $d(i,a) \leq d(i,t) + d(a,t)$ (triangle inequality).

If $|d(i,a) - d(a,t)| = d(i,t)$, clearly, v_i is either on the shortest path from node a to t or t is on the shortest path from v_i to node a . In both cases advancing to node v_j on the shortest path from v_i to t reduces the distance to the vp .

We have showed that for any node there is an adjacent node that is closer to the vp ($d(q,t),0$) and thus the packet will be continuously forwarded until reaching t . ■

Another observation is that if the algorithm does not use t as one of the beacons, the algorithm cannot guarantee reaching the destination without resorting to the shortest path routing. This observation derives from the fact that there can be several nodes having position similar to t in which the algorithm will reach a local minimum and stop. If not reaching to t , the algorithm must resort to shortest path routing to reach t .

For the sake of presentation we first show the upper bound for the path length traversed by VPSR using L^1 -norm for forwarding decisions. Then, based on this result, we turn to the case of L^2 -norm. One should note the difference between norms used for forwarding decisions in the VPSR and norms used to measure the path length in the following results.

Consider the VPSR in uniform link cost network, using two arbitrary beacons v_m and v_n and set of vps vp^0, vp^1, \dots, vp^K ($vp^0 = s$). We are interested in an upper bound for the path length (number of nodes traversed) that starts from s ($s=v_0$), advances to node v_i that is closer (in the L^1 -norm) to vp^l than vp^0 . Then from v_i , the path continues to a node v_2 that is closer to vp^2 than vp^1 , and so on. That is, in every step the path goes from v_i to v_{i+1} which is closer to vp^{i+1} than v_i .

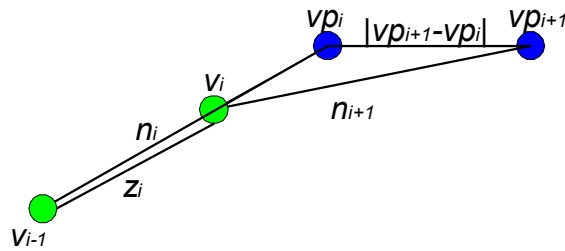


Figure 8: Nodes and vps in definitions

Claim 1: Let z denote the number of hops traversed from node v_i aiming at vp , and stopping in node v_j . Using the VPSR with L^1 -norm, the following holds: $\|\beta^j - vp\|_1 + z \leq \|\beta^i - vp\|_1$.

The proof is by realizing that every time z increases, the distance to the vp decreases by at least one unit.

Technical Report

Lemma 2: Let L_k be the number of nodes traversed by VPSR up to step k . L_k is bounded by:

$$L_k \leq \sum_{i=1..k} \|vp^i - vp^{i-1}\|_1.$$

Proof: Let $n_i = \| \beta^{i-1} - vp^i \|_1$. In the L^1 -norm this is given by $n_i = |d(i-1, m) - vp_m^i| + |d(i-1, n) - vp_n^i|$. Let z_i denote the number of hops traversed in step i . Clearly (from claim 1) $z_i \leq n_i$. Thus, according to the definitions, $L_k \leq n_k + \sum_{i=1..k-1} z_i$.

We will show by induction on k that: $n_k \leq \sum_{i=1..k} \|vp^i - vp^{i-1}\|_1 - \sum_{i=1..k-1} z_i$.

Starting from v_0 , then $n_1 \leq \|vp^1 - vp^0\|_1$. According to the definitions $L_1 = z_1$ and clearly $L_1 \leq n_1$.

Inductive Assumption: We assume $n_k \leq \sum_{i=1..k} \|vp^i - vp^{i-1}\|_1 - \sum_{i=1..k-1} z_i$ and need to prove for $k+1$.

According to the triangular-inequality, the distance between v_k and vp^{k+1} obeys $n_{k+1} \leq \|vp^{k+1} - vp^k\|_1 + \|\beta^k - vp^k\|_1$ (see illustration in Figure 8).

Also, according to Claim 1 $\|\beta^k - vp^k\|_1 \leq n_k - z_k$. Thus, $n_{k+1} \leq \|vp^{k+1} - vp^k\|_1 + n_k - z_k$. Using the inductive assumption we have: $n_{k+1} \leq \sum_{i=1..k+1} \|vp^i - vp^{i-1}\|_1 - \sum_{i=1..k} z_i$. Using the definition for L_i , we finally have:

$$L_k \leq \sum_{i=1..k} \|vp^i - vp^{i-1}\|_1 \blacksquare$$

Theorem 4: Using K vps an upper bound for the path length: $L \leq \sum_{i=1..K} \|vp^i - vp^{i-1}\|_1 + D$, where D is the network diameter (maximal distance between two nodes in the network).

Proof: According to Lemma 2, path length until reaching v_K is bounded by $L_k \leq \sum_{i=1..k} \|vp^i - vp^{i-1}\|_1$. Node v_K can be any node in the network. Since VPSR may resort to the shortest path routing between and t , the upper bound for this is D . \blacksquare

Nonetheless, in practical cases a significantly smaller bound can be achieved:

Theorem 5: If v_n and t are used as beacons and

$$vp^K = (d(n, t), 0), \text{ the path length is bounded by: } L \leq \sum_{i=1..K} \|vp^i - vp^{i-1}\|_1.$$

The proof is immediate by from Theorems 3 and 4. This bound is tight due to paths that traverse all vps.

Using the result for the L^1 -norm we show an upper bound for the VPSR when L^2 -norm is used:

Theorem 6: An upper bound for the path length using the L^2 -norm: $L \leq 2 \sum_{i=1..K} \|vp^i - vp^{i-1}\|_1 + D$.

Technical Report

Proof: One should note that the proof of Theorem 5 can be carried out for any distance measure for which Claim 1 and the triangular-inequality hold. In particular consider navigation of VPSR using the L^2 -norm. Consider a distance measure (in \mathbf{R}^2) given by $\|a - b\| = |2a_x - 2b_x| + |2a_y - 2b_y|$. One can verify that the triangular inequality holds for this distance measure. Further, Claim 1 can be shown to hold for this measure as well (since, according to Theorem 2, the number of steps to advance from point (x,y) to point $(x + \Delta x, y + \Delta y)$ is bounded by $\Delta x + \Delta y + \max(\Delta x, \Delta y)$). (Thus, one can show that $L_k \leq 2 \sum_{i=1..k} (\|vp_m^i - vp_m^{i-1}\|_1 + \|vp_n^i - vp_n^{i-1}\|_1)$, which is twice the bond of L^1 -norm, and the upper bound for path length in general is then: $L \leq 2 \sum_{i=1..K} \|vp^i - vp^{i-1}\|_1 + D$. ■

Note that the bounds are for VPSR using arbitrary sequence of vps . Choosing a logical sequence (e.g. sequence describing an arc) will lead to relatively short paths. Having discussed the length limits we now show that in a dense enough network, the use of VPSR guarantees that a packet will follow the path dictated by the source.

Observation 5: Consider a network where each node has eight neighbors as depicted in Figure 9-(a) and where two beacons are used. Assume that all the virtual positions provided obey $vp=(vp_i, vp_j): |vp_i - vp_j| < d(i,j)$. Then the packet is guaranteed to follow the path dictated by the sender.

The requirement $|vp_i - vp_j| < d(i,j)$ is trivial and only requires that each of the vps is a reasonable address; the sender of course can select such points easily. This observation derives from the fact that as long as $|vp_i - vp_j| < d(i,j)$, each node has neighbors leading it to all possible directions with respect to vp position. This is analogous to routing on continues plane. That is, node v_i relatively to node v_j has all possible neighbors with respect to the distances from it, namely closer by one, farther by one or having equally distance from both beacons. When looking at the network on a plane where the y -axis is the distance from one beacon and the x -axis is the distance from other beacon (see Figure 9-(b)) it is clear that from every position one can move to any other position through the expected shortest path. Also, it is clear that there are nodes with all possible real beacons positions. In Figure 9-(b) we illustrate a path from s to t , where s and t are the beacons, passing through vps : $(0,4) \rightarrow (2,4) \rightarrow (4,2) \rightarrow (4,0)$.

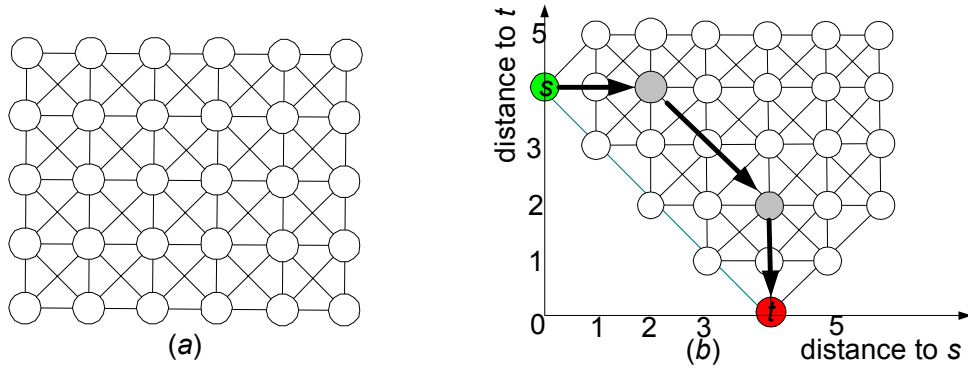


Figure 9: Optimal network for VPSR

E. VPSR in non-uniform link cost networks

Next we analyze the VPSR in non-uniform link length (cost) networks. In the algorithm described in Section III-C the VPSR did not take into account link lengths (costs). This might lead to very long paths in non-uniform link cost networks. Since the objective of VPSR is to route packets on limited length paths, sometimes even at the cost of passing through avoidance nodes, we enhance the algorithm by the following method: In the process of finding the next hop we use a parameter α that reflects the ratio between the gain in distance (to a virtual position) achieved by moving to a neighboring node and the cost of traveling to the neighboring node. That is, a packet will be forwarded from node v_i to a neighbor node v_j only in case $\alpha \cdot c_e < \Delta d$, where $\Delta d = \|\beta^i - vp\|_2 - \|\beta^j - vp\|_2$, vp is the target virtual position and c_e is the cost of the link between v_i and v_j . Parameter α reflects the tradeoff between the routing flexibility and the path length. Note that if $\alpha \rightarrow 0$ the algorithm may yield very long routes as will be shown in Observation 8. The implementation details of the algorithm are given in Appendix A.

In non-uniform link cost networks VPSR guarantees the following properties:

Theorem 7: Consider a packet that is forwarded from node v_i , using two arbitrary beacons, to virtual position vp . Then the length of the path, L , traversed by the packet until reaching a local minimum is bounded by: $L \leq \alpha^{-1} \|\beta^i - vp\|_2$.

Proof: In every step the distance between v_i and the vp must decrease. The maximal ratio between the length of the edge and the decrease in the path length is limited by parameter α , which leads to:

$$L \leq \alpha^{-1} \|\beta^i - vp\|_2 \blacksquare$$

Observation 6: $|V-1|$ is the upper bound on number of nodes a packet may traverse between a node and a virtual position.

Technical Report

Theorem 8: The upper bound for path length produced by an arbitrary sequence of K vps is:

$$L \leq \alpha^{-1} \sum_{i=1..K} \|vp^i - vp^{i-1}\|_1 + D.$$

Proof: Using a method similar to the proof of Theorem 5, with l' -norm where distances are measured as $\|a - b\| = |\alpha^{-1}a_x - \alpha^{-1}b_x| + |\alpha^{-1}a_y - \alpha^{-1}b_y|$ we can put the upper bound for the path length by the above expression for L . ■

Observation 7: If $a \leq \sqrt{2} - \sqrt{2 - 2D^{-1} + D^{-2}}$ (where D stands for network diameter), then it does not limit the routing. In such a case if the last virtual position is $(d(s,t),0)$, the algorithm will always reach the destination without resorting to the shortest path routing. This is similar to the case of uniform link networks (see Theorem 3). However, in case this does not hold, the algorithm might be required to resort to shortest path routing for reaching the destination.

Observation 8: If $\alpha \rightarrow 0$ is very small the algorithm may yield very long routes. One should note that the forwarding mechanism might lead to long paths towards a virtual position despite the existence of shorter paths. An example to these properties is depicted in Figure 10. The beacons in this example are s and t and $d(s,t)=100$. We start from node v whose position is $(40,120)$ and aim at reaching the target virtual position $vp=(80,80)$. From v to a , $\Delta d = \sqrt{40^2 + 40^2} - \sqrt{40^2 + 39^2}$ and c_e is 100. Note that a has the shortest distance to the virtual position in comparison to the other neighbors of v , namely s and t . If $\alpha < \frac{\sqrt{40^2 + 40^2} - \sqrt{40^2 + 39^2}}{79}$,

to the virtual position will be on the long path: $v \rightarrow a \rightarrow b \rightarrow c \rightarrow \dots \rightarrow vp$, otherwise the algorithm will stop at this point. Note that there is a shorter path from v to VP , namely $v \rightarrow s \rightarrow vp$, but going from v to s is impossible by the algorithm since v is closer by our metrics to the vp than to s .

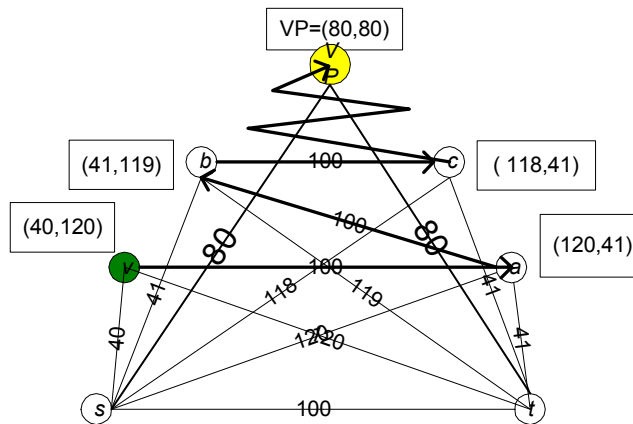


Figure 10: Long paths in Virtual-Position Source-Routing without limiting parameter α

F. Choosing Intermediate Virtual Positions for VPSR

Up to this point we have introduced the basic properties of the VPSR routing algorithm. This algorithm can be used for various objective functions as fundamental machinery for non-shortest path routing. For example, the VPSR can be used for dispersing packets over several multiple disjoint paths for security enhancement [25]. Having the general algorithm, there are several algorithmic issues to be addressed: 1) Which nodes should be used as beacons? 2) How many *vps* to used, and 3) how to choose the *vps* to achieve the routing objective?

In this work we focus on the shortest-path avoidance and area-avoidance objectives. For the shortest-path avoidance the obvious candidates for beacons are the source and the destination since the sender does not have any other useful information. For the area-avoidance the obvious candidates are the center node and the destination.

Generally speaking, using many *vps* increases the chances for a packet to traverse a path that is similar to the virtual path dictated by the source. Using many *vps* can provide protection against traveling to undesired directions. For example, consider the network in Figure 11 using *s* and *t* as beacons and a single $vp=(2,2)$. In this case, the next hop (after *s*) will be the node at position (1,2) (where the navigation to (1,3) stops due to local minimum), which leads to the shortest path. However, if we use three *vps*: $vp^1=(1,3)$, $vp^2=(2,2)$ and $vp^3=(3,1)$ the resulting path will avoid the shortest path and go according the desired path. However, since the routing information is located in packet header, using many *vps* increases overhead. Thus, the tradeoff is clear.

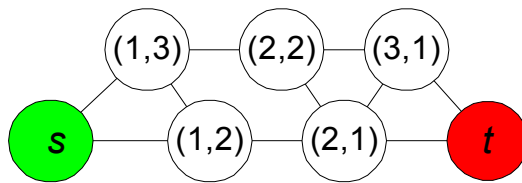


Figure 11: Choosing *vps*

There are various heuristics that can be used for choosing the *vps*. The approach we used is to choose *vps* that describe a geometrical path that detours the avoidance nodes. For shortest path avoidance we used a heuristic in which the *vps* describe a triangle between the source and the destination. The height of the triangle is determined according to the maximal permitted length. For area avoidance, we use a sequence of *vps* that form a triangle that detours the avoidance area. After choosing the number of *vps*, *n*, we divide the triangle into *n* equal portions and according to geometric calculation we determined the sequence of *vps*.

Technical Report

An interesting insight about the VPSR is that it might not perform as well as expected under certain network topologies, even if these networks are not poorly connected. An example to such topology is a grid network. The reason is that, due to symmetry, many nodes can have similar beacon positions in the “system of coordinates” dictated by the beacons. This reduces the number of potential next-hops and thus reduces the routing flexibility. For example, in Figure 12 we show a grid network where s and t are used as beacons. In this figure, all the nodes having the same index within a shadowed area have identical beacon positions. As can be seen, many nodes have only two adjacent nodes with different positions (e.g. most of the nodes marked by 3 have only two adjacent different beacons positions, namely, nodes marked by 2 and 4). A remedy for this can be the use of a third beacon, to break the symmetry inflicted by the two beacons, and placement of the beacons away from each other.

As we will show in the simulation analysis, the VPSR results, in most cases, in node avoidance paths with relatively small increase of the path length. Nonetheless, when there are many equally good shortest paths (e.g. a grid network), other routing algorithms based on the virtual positioning routing paradigm, can be considered. For example, for area-avoidance forwarding one can select from the neighbors who are closest to the destination, the one which is the most distant from the center (avoidance) node.

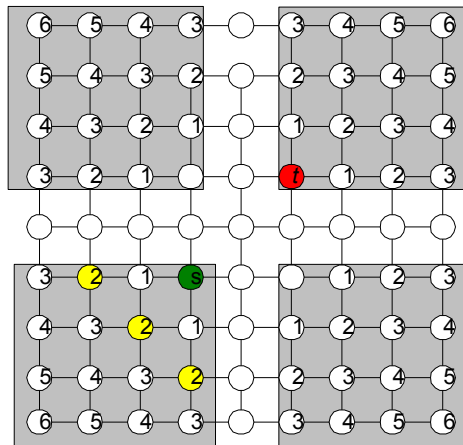


Figure 12: VPSR is a grid network

IV. EVALUATION WITH SIMULATION

In this section, we evaluate the quality of VPSR in general networks via simulation. We divide our discussion into disjoint-path avoidance and area-avoidance.

A. Simulation Methodology

We experimented the results of VPSR over three network topologies: *i)* The Barabasi-Albert model that produces the power law distribution for network connectivity. We used the BRITE [5] 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 100x100 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 edge was the Euclidian distance between the nodes. We validated that all tested networks were connected.

For each simulation we generated 50 random networks under the same parameters. In each network we randomly chose source and destination pairs. For each pair we executed the routing algorithm as will be explained next.

B. Shortest-Path Avoidance Simulation Results

To evaluate the results we use the shortest path *node jointness ratio* metric, defined as follows: Let p_1 denote the shortest path and p_2 the path resulting from VPSR. For path p , $V(p)$ is the set of nodes in path p . The node jointness ratio metric, ρ , is measured by $\rho = V(p_1) \cap V(p_2) / V(p_1)$.

For every network we randomly chose 50 source-destination pairs and tested the node jointness ratio for shortest path avoidance. To evaluate the quality of VPSR we used two other algorithms as reference. The first algorithm calculated, using a *centralized* algorithm, the shortest path having minimal node jointness ratio. In this algorithm we directed the edges and from every node constructed two nodes: v_i and v_j . All incoming edges were connected to v_i and all outgoing edges were connected to v_j . The nodes were connected by an edge whose cost is MAX_COST. On this network we executed the Dijkstra algorithm to find the optimal path. The second algorithm implemented the *Random-Source-Routing* (we will refer to as RSR), for details see [8]. In this algorithm the source randomly chooses an intermediate node v_i . By specifying the intermediate node in the header, the weak source routing mechanism routes the packet on the shortest path from s to v_i and then on the shortest path from v_i to t . We chose an intermediate node having the distance that is half (or as close as possible to half) of the shortest path length.

Figures 13-16 depict the simulation results. As can be observed in Figures 14 and 16, the jointness ratio increases with the path length. This is a result of having higher routing flexibility and increased chances of

Technical Report

running away from the shortest path. From thorough study of the simulations results, we saw that VPSR results in high likelihood of routing via paths partially or fully disjoint from the shortest path, thus achieving its objective. As expected, this likelihood increases with the growth in network connectivity. We observed that in most cases, if jointness occurs, it occurs near the source or the destination while in the middle of the path it occurs infrequently. This makes the VPSR attractive for many purposes such as security, where avoiding nodes on the middle of the shortest path is important [25]. One should note that the network generated by the Baranassi and Waxman model are designed to have more than one route for each pair of nodes and therefore the optimal jointness ratio is very close to zero.

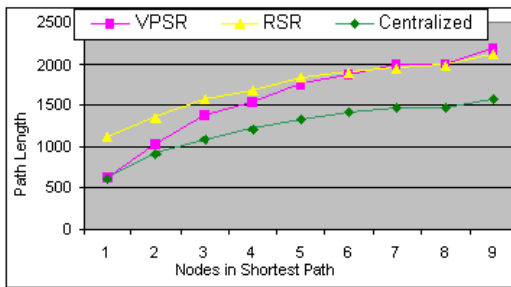


Figure 13: Path length for Waxman model $\alpha = 0.5$

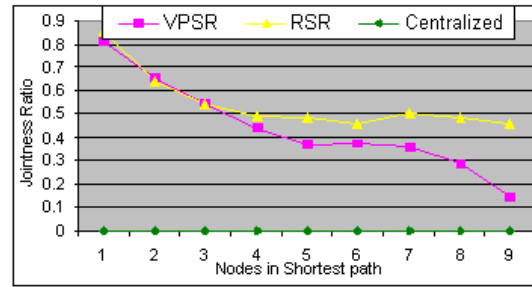


Figure 14: Node jointness ratio for Waxman model $\alpha = 0.5$

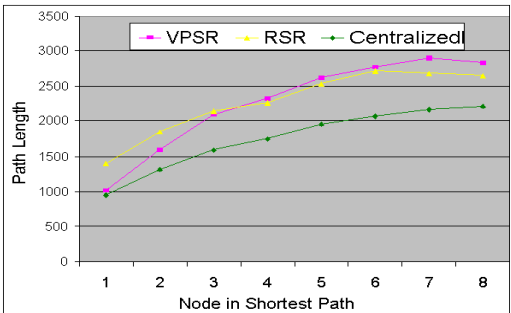


Figure 15: Path length for Baranassi Model $\alpha = 0.5$

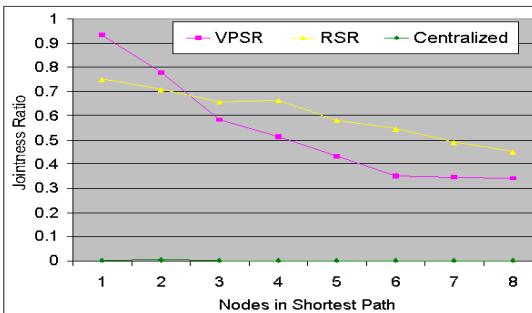


Figure 16: Node jointness ratio for Baranassi model $\alpha = 0.5$

From Figures 14 and 16 we can also observe that the VPSR results, in most cases, in lower jointness ratio in comparison to the RSR algorithm. The difference increases in the number of nodes along the shortest path. This is a result of the fact that the random selection of intermediate nodes in RSR, may lead to intersections in the middle of the path, while in VPSR the likelihood for such intersections is very small. The path lengths for both algorithms were similar in most cases and not by much longer than the optimal path, as can be seen from Figures 13 and 15.

Similar conclusions were found for the ad-hoc topologies.

C. Area-Avoidance Simulation Results

An avoidance area in our model, is described by a center node, v_i , and all nodes having shortest distance length from v_i that is less than R . We use a *node jointness ratio* metric that is similar to the shortest-path node jointness ratio where we replace the nodes on the shortest path by the nodes in the avoidance area.

The VPSR for area avoidance chooses the shortest path in case it does not contain nodes from the avoidance area. Hence, to get meaningful results, we planted the center nodes to be along the shortest path. In Figures 17 and 18 we present the results of the VPSR under the ad-hoc network model, where the parameter for connectivity radius is $D=10$ and $D=20$ respectively, and $R=5$.

The results show that, similarly to the shortest path avoiding, the likelihood of routing via area-avoiding routes increases dramatically in the network connectivity growth, as can be observed in Figures 17-18. The average path length increase (relatively to the shortest path) was about 15%. Thus, the algorithm is successful in providing an area-avoiding route at the cost of relatively small path length increase. The results for the other network models exhibited similar results.

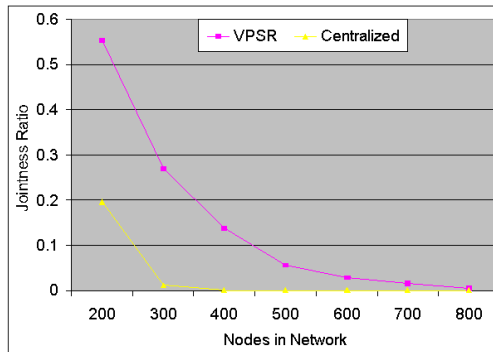


Figure 17: Area-Avoiding in ad-hoc topologies with $D=10$ and $R=5$ ($\alpha = 0.5$)

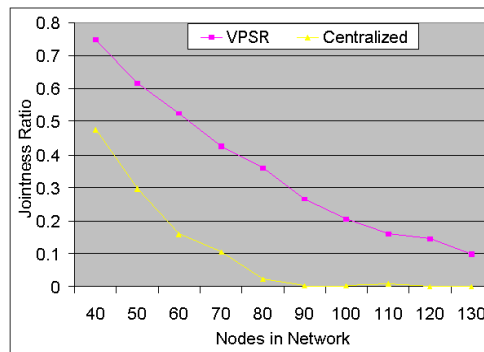


Figure 18: Area-Avoiding in ad-hoc topologies with $D=20$ and $R=5$ ($\alpha = 0.5$)

V. CONCLUSIONS AND SUMMARY

We introduced the concept of node-avoiding routing whose goal is to address security vulnerabilities inherited in the traditional shortest path routing. We addressed the objectives of shortest-path avoidance and area-avoidance. We addressed them in the framework of distance-vector networks which were formed originally to yield shortest-path routing *only*.

We introduced a routing paradigm based on a concept of incorporating the full (or part of) the distance vector information in the forwarding procedure. Based on this paradigm we proposed and studied the Virtual-Position-Source-Routing (VPSR) forwarding methodology. VPSR is based on the concept of

Technical Report

routing through virtual positions in the network, where the positions are determined by distances from several nodes. We studied the properties of this methodology and evaluated its effectiveness for a wide set of networks. The results showed that it is quite effective in achieving these objectives.

The paradigm of distance vector routing may have potential for a variety of other algorithms which we leave for further study.

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] BRITE: Boston university Representative Internet Topology generator, www.cs.bu.edu/brite/
- [6] M. Chay, S. Moon, Chong-Dae Park, and Aman Shaikh "Placing Relay Nodes for Intra-Domain Path Diversity".
- [7] J. Chen, P. Druschel, D. Subramanian "An Efficient Multipath Forwarding Method", INFOCOM 1998: 1418-1425.
- [8] R. Cohen, G. Nakibli, "On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing", IEEE Infocom 2004.
- [9] Y. Guo, F. Kuipers, P. Van Mieghem "Link Disjoint Paths for reliable QoS", International Journal of Communication Systems 779-798 NOV 2003.
- [10] J. P. Hespanha and S. Bohacek, "Preliminary results in routing games," in Proc. Of the 2001 American Control Conference, June, 2001.
- [11] C. Hoops "Analysis of an equal-cost, multi-path algorithm", RFC 2992, Nov. 2000.
- [12] B. Karp, H.T Kung, "Greedy Perimeter Stateless Routing for Wireless Networks", in Proceedings MobiCom 2000, Boston, MA, August, 2000, pp. 243-254.
- [13] P. P. C. Lee, Vishal Misra and D. Rubenstein, "Distributed Algorithms for Secure Multipath Routing", IEEE Infocom 2005.
- [14] 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.
- [15] H. Lim, J. C. Hau, C. Hoi, "Constructing Internet Coordinate System Based On Delay Measurement", Internet Measurement Conference 2003.
- [16] 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.
- [17] T. Nguyen, A. Zakhor, "Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks", Infocom 2003.

Technical Report

- [18] A. Orda, A. Spronston, "Efficient Algorithms for Computing Disjoint QoS Paths", Infocom 2004 Hong-Kong.
 - [19] 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.
 - [20] 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.
 - [21] D. Sidhuand, R. Nairand, S. Abdallah "Finding Disjoint Paths in Networks", In Proc. of ACM SIGCOMM' 91, 1991.
 - [22] Villamizar, "OSPF optimized multi-path (OSPF-OMP)", draft-ietf-ospf-omp-03.
 - [23] 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.
 - [24] 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.
- H. Zlatokrilov, H. Levy, "Privacy and Reliability by Dispersive Routing", IEEE INFOCOM 2006, Barcelona.

Navigation in Distance Vector Spaces and Its Use for Node Avoidance Routing
Technical Report

VI. APPENDIX A: *VIRTUAL-POSITION SOURCE-ROUTING* ALGORITHM

The sender “draws a virtual path” by a sequence of *virtual positions* (each *virtual position* contains two distances) from beacons m and the destination n ($\{(vp_s^1, vp_t^1) \dots (vp_s^k, vp_t^k)\} \in P$, where vp_s^i and vp_t^i are distances of pair i from the source and the destination respectively). Each packet will contain the sequence of vp Nodes forward the packet to a neighbor that minimizes the *distance from virtual position*, defined here in l^2 -norm distance function. If the packet is in *local minimum* (no adjacent nodes closer to the *virtual position*), the next *virtual position* ($i+1$) becomes the virtual target (the *virtual position* at the top of the list can be removed). If the list is empty the shortest path to the destination is taken.

The Virtual-Position Source-Routing (VPSR) forwarding algorithm both for the uniform and non-uniform link cost networks is as follows (for the uniform link cost networks $\alpha \rightarrow 0$):

```

VPSR ( $m, n, pairSet$ )
{
    nextHop = NIL
    tmpNode = NIL
    minDistance = MAX_INT
     $vp_m = pairSet.head().vp_m^i$ 
     $vp_n = pairSet.head().vp_n^i$ 

    if ( $pairSet.size = 0$ )
        return nextSP( $t$ )           //the next hop using the shortest path algorithm

     $LD = \sqrt{|SP(s) - vp_m|^2 + |SP(t) - vp_n|^2}$  //Local distance to the virtual position
                                           //SP()-return the next hop via shortest path

    forall adjacent nodes  $v_j$ 
    {
         $VjD = \sqrt{|d(j, m) - vp_m|^2 + |d(j, n) - vp_n|^2}$ 

        if ( $VjD < LD$  and
             $VjD < minDistance$  and
             $|LD - VjD| > \alpha c_{i,j}$ ) //Extension factor – does not take affect in uniform link cost networks
        {
            minDistance =  $VjD$ 
            nextHop =  $v_j$ 
        }
    }
    //in case of equally good neighbors, choose the one closer to n, in case there are
    //equal choose the one with the minimal address that is not on the shortest path from m to n
    else if (nextHop != NIL and
            minDistance ==  $VjD$  and
            ( $d(j, t) < d(i, t)$  or  $d(j, t) = d(i, t)$  and  $address(v_j) < address(nextHop)$ ))
    {

```

Navigation in Distance Vector Spaces and Its Use for Node Avoidance Routing

Technical Report

```
        nextHop = vj
    }
}

if ( nextHop == NIL )
{
    pairSet.pop()
    return VPSR ( s, t, pairSet)
}
return nextHop
}
```

Note that **all** network nodes MUST implement this forwarding algorithm.

ShortestPathAvoidanceVPSR (s, t, pairSet)

(s=source node, t = destination, pairSet – set of virtual locations set by the source)

```
{
    return VPSR ( e, t, pairSet)
}
```

AreaAvoidanceVPSR (e, t, d, pairSet, inVPSR)

(e=center node, t = destination, R=avoidance area distance from e, inVPSR-indicator in header)

```
{
    X = a*d (a is a constant indicating when to initiate the VPSR. In simulations we used a=2)
    if ( ! inVPSR )
        v = shortestPath(t)
        if ( d(v,e) < X )
            return v
        else
            set in packet : inVPSR = true
    return VPSR ( e, t, pairSet)
}
```