Geographic Routing in Large-Scale Highly-Dynamic Mobile Ad hoc Networks

Ben Newton, Jay Aikat, Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
Email: {bn, aikat, jeffay}@cs.unc.edu

***Abstract*—In the near future extremely large scale mobile ad hoc networks of thousands or tens of thousands of mobile nodes will be physically feasible and desirable for a host of applications. However, routing within these networks is challenging, especially at high data rates and when node movement is highly-dynamic. In this work we present Topology Aware Geographic Routing (TAG), a position-based routing protocol that strategically uses local topology information (when available) to make better local forwarding decisions, decreasing the number of hops required to deliver a packet when compared with other geographic routing protocols. In addition TAG is able to reliably deliver packets even in topologies that violate the often used but unrealistic *unit disk graph* and *quasi-static* assumptions. We present empirical results from a variety of simulations, illustrating how TAG outperforms GOAFR+, GFG, and OLSR in both theoretical environments and in a simulated, real-world, continental-scale airborne network.**

I. Introduction

Mobile wireless ad hoc networks consisting of thousands of mobile nodes are desirable for many applications. For example, the United States military envisions a highly-adaptive "ultra-large" ad hoc network consisting of tens of thousands of nodes [1], [2]. Similarly, Google's project Loon, proposes launching tens of thousands of networked balloons into the upper atmosphere in an effort to provide network access to users on the ground even in remote locations [3], [4]. Another example application in which we are specifically interested is connecting thousands of in-flight commercial aircraft in the continental United States into a high-capacity airborne network. However, scaling ad hoc networks to this size is difficult at best. In particular, efficient routing in such large dynamic networks is an open problem.

Geographic routing (also called Geometric or position-based routing) considers the physical positions of the nodes when making forwarding decisions. This unique routing strategy has been proposed as one solution to the challenge of routing in huge mobile ad hoc networks [5]. Much of the work in this domain, however, has been largely theoretical, ignoring many real-world concerns, such as a realistic and constantly changing topology. In this work we introduce Topology Aware Geographic Routing (TAG) a new geographic routing protocol that is able to essentially guarantee delivery while supporting constantly changing topologies that are not restricted to be unit disk graphs. Further, we show how the new protocol can achieve even better performance by taking advantage of local topology information.

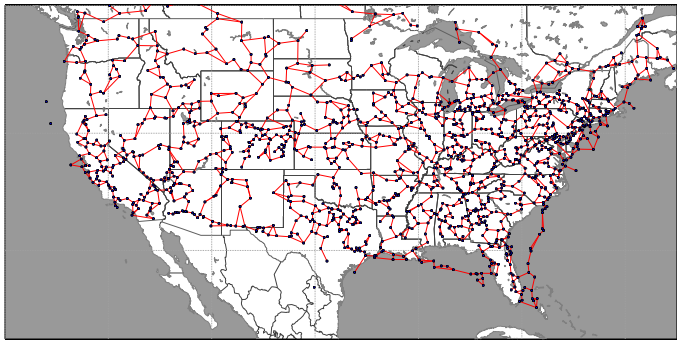

Fig. 1. Continental-scale airborne network topology connecting the positions of a set of actual aircraft as seen at 2:00 p.m. on July 9, 2015

The development of this new geographic routing protocol is motivated by our desire to develop a set of protocols sufficient to enable high-bandwidth communication among a large set of high-flying aircraft. Such a network could be utilized for a host of applications, especially if it were connected to the Internet. Figure 1 shows a snapshot of the actual positions (blue dots) of commercial aircraft above 10,000 feet altitude at 2:00 p.m. on July 9, 2015, as reported by the Federal Aviation Administration's (FAA's) Aircraft Situation Display to Industry (ASDI) data [6]. The figure also shows a conceivable topology (red edges) for interconnecting these aircraft assuming each plane is equipped with three directional data links (as described below). The topology of such a network is constantly changing as nodes (aircraft) join or leave the network, and as nodes move in and out of one another's communication range. Our goal is a routing protocol that can successfully deliver a high percentage of packets in such a constantly changing network, while keeping the overhead minimal.

II. Background

A. Unit Disk Graphs

Before packets can be routed, network nodes must first establish communication links with other nodes. The network topology indicates the arrangement of these links between nodes in the network. For traditional wired networks the network topology is determined by the physical cables connecting network nodes, and it is generally fairly static. In contrast, most wireless networks use omni-directional antennas to transmit their messages in all directions to all nodes within

some maximum range. The topology of a wireless network is often modeled as a *unit disk graph* (UDG), where an edge exists between any two nodes whose separation distance is less than 1 unit (the maximum range). In other words, a node is always connected to all other nodes within some maximum transmission range (which has been normalized to 1 unit), and is never connected to any node outside that range [7]. The unit disk graph is an accurate model of a 2-dimensional wireless network if: (1) all nodes have the same transmission power, and the same receive gain, yielding the same maximum transmission radius, (2) every node's transmission pattern is a perfect circle (each node transmits and receives equally in every direction), and (3) there are no radio opaque obstacles and no multipathing to interfere with this perfect transmission. While convenient for theoretical analysis, unit disk graphs rarely match real-world wireless network topologies [7], [8], [9].

B. Directional Data Links and Topologies

Another option for wireless communication is the use of directional data links. Rather than communicating by sending energy in all directions, these nodes utilize "smart" antennas [10], to instead focus their transmission energy towards the intended receivers, enabling lower energy use, longer ranges, and less interference. Directional Radio Frequency (RF) links have been used by the military for years [11], enabling long-range covert communications. Free-Space Optics (FSO) is another cutting-edge directional communication method that is becoming more attractive as demand for higher data rates increases and frequency spectrum allocations become more difficult to obtain. These steerable directional FSO links communicate using pulses of light from a laser, essentially enabling wireless fiber-optics. DARPA has demonstrated data rates up to 10 Gbps and distances up to 200km using a hybrid RF/FSO link[12]. Both Google and Facebook have proposed utilizing FSO links in their airborne balloon [13] and airborne drone [14] networks. In our research we consider commercial aircraft outfitted with three FSO transceivers.

Unlike an omni-directional data link, which can essentially form connections with every other node within range, a Free-Space Optics directional link can form a connection with only one other node at any given point in time. Thus, the degree (number of incident edges) at each vertex in a directional network topology graph is limited to the number of FSO links on the associated node. More importantly, the topology for a network using directional links must be explicitly managed. Some protocol must coordinate which of the possibly many potential neighbors each link should point at and connect with. Networks of directional links avoid contention and interference at the cost of having a more complicated topology which must be managed.

C. Airborne Network Topology Management

As mentioned earlier, one motivating application for our proposed routing protocol is a continental-scale airborne network of commercial cargo and passenger aircraft. To connect these airplanes to one another at high data rates over long ranges requires the use of directional links (FSO or RF). The use of directional links, in turn, requires a topology management protocol to explicitly manage which links point at what other nodes as the airplanes move and the topology evolves.

In our prior work [15] we introduce a distributed topology management algorithm, called DCTRT, for forming a degree-constrained topology graph among the nodes. Nodes using this algorithm and associated protocol are able to actively manage a large-scale network topology in a de-centralized manner, exchanging nothing more than position information with nearby nodes. Given only this position information, each node periodically computes a local topology, and establishes the links in the topology incident at that node. Because there is no explicit connection agreement between nodes, there can be cases where links are pointed at nodes that don't reciprocate. However, generally, given a dense enough network and the right algorithm, a sufficient topology is produced.

For our airborne network we note that the FAA has mandated that by the year 2020 every aircraft operating in controlled airspace (classes A, B or C of the U.S. National Airspace System) must constantly broadcast their position information using Automatic Dependent Surveillance - Broadcast (ADS-B) [16]. This information is received by FAA systems and other aircraft, effectively allowing aircraft to exchange their position information with all other in-range aircraft once a second. By utilizing this ADS-B information, our topology management scheme is able to operate with absolutely no overhead on the directional links.

D. MANET Routing Protocols

Mobile ad hoc networks typically employ nodes that act as both routers and end-users. Nodes that are not within mutual transmission range of one another must communicate by relaying their messages through a series of intermediate nodes, each acting as a router. A routing protocol is needed to determine at each node in the series, where the message should be forwarded next. Traditional routing protocols are not well suited for routing in networks where the topology changes frequently [17]. Mobile Ad hoc network (MANET) routing protocols are designed to support networks of mobile nodes, but even these protocols often fail to efficiently route packets in highly-dynamic, large-scale networks [18]. Consider, for example, a network of ten thousand mobile nodes, and two nodes on opposite edges of this network attempting to communicate. A reactive MANET routing protocol (one which seeks to establish routes "on demand"), will need to buffer packets until it can find a path through the network to the distant node. Once a path is found packets may begin flowing. However, unless the nodes are essentially stationary, there is a high probability the path will quickly become invalid. All it takes is one node along the long path to leave the network or to move out of range of an adjacent node supporting the path, to make the path obsolete. Packets must then be delayed

or dropped while the path is repaired, and extra management packets will likely need to be sent, incurring overhead.

Using, instead, a proactive routing protocol (which seeks to always maintain routes between every pair of nodes) only exacerbates the situation, generating what has been described as "torrents" of link status change messages in an effort to recover from any failure on any path [19]. The overhead associated with these messages can quickly saturate the available network bandwidth for a large networks with a frequently changing topology. Despite the high cost, only a fraction of the packets may actually arrive at their intended destinations because the routing information so quickly becomes stale. In contrast, geographic routing protocols don't actively maintain routes, and never need to determine or repair paths through the network.

E. Geographic Routing

Geographic routing protocols forward packets based on the geographic position of the destination, not by identity or address. The nodes supporting a geographic routing protocol need only store minimal information about other nearby nodes, and never global topology or path information. There is no need to set up a path because hop-by-hop decisions are made about where the packet should be forwarded next. This allows the protocols to quickly adapt to changes in the network while avoiding stale information.

These protocols, however, assume that a node is aware of its own physical location and the positions of nearby nodes. Further, these protocols require that a sending node knows the position of a packet's intended destination. This information is often obtained using a separate location service protocol [20]. As described previously, in our example airborne network application, the positions of nearby nodes are fortuitously already being exchanged via ADS-B [16]. We call nodes that are within transmission range that node's neighbors, and nodes that are within position exchange (ADS-B) range its community members. Neighbors to which a connection exists are directly connected neighbors.

There are two main approaches to geographic routing: *greedy routing* and *face routing*. In greedy routing (aka greedy forwarding or earlier as Cartesian routing) each node a packet visits attempts to greedily forward it so that it makes maximum progress towards its destination [21]. The packet is forwarded at each hop to whichever directly connected neighbor is nearest the packet's destination. Greedy routing is generally efficient, but it alone cannot guarantee delivery in arbitrary networks. If the packet reaches a node whose neighbors are all further away from the destination than it is, the packet has reached a dead end, or local minimum, and cannot make greedy progress. If local minima exist in a topology greedy routing must employ some other method to assist in backtracking and routing around the void (or hole) in the topology.

Face Routing (aka Compass Routing II) is the other main geographic routing approach [22]. It requires that the topology graph be planar. A planar graph is one where no two edges cross, or in other words, one where all edge intersections occur at vertices. A planar graph, therefore, consists of a set of regions, called faces, bounded by edges. There may be many interior faces, which make up a graph, but always also one extra exterior face which encompasses all other space. Face routing seeks to traverse faces of the planar graph, while at each step advancing towards the destination.

To better understand face routing, it may help to visualize the planar graph as a maze. The edges are corridors of the maze, and the vertices are small rooms with a doorway leading to a corridor for each of the potentially many adjacent edges. The following right-hand rule applies equally well to solving mazes and exploring the boundaries of faces in a planar graph. "Upon entering the maze, always follow the wall to your right. When confronted by an intersection, always turn right, keeping the wall always to your right" [23]. Just as we can get out of any maze by keeping our right hand on the right entry wall, we can always travel around the entire boundary of any face by, at each room (vertex), taking the doorway to the corridor (incident edge) immediately to the right. To determine which edge (corridor) should be used to initially start exploring a face boundary, one may imagine traveling from the center of the face which is to be traversed towards the start vertex, and following the same right hand rule upon reaching the vertex (enter the first doorway to your right). Note that one could, with equivalent results, instead use the left-hand rule (always turning left) to travel the opposite direction around any face.

Various face routing methods utilize the right-hand and/or left-hand rules to traverse the faces of a planar graph and arrive at the destination. The general concept is to have a packet travel around one face until an adjacent face is found which is closer to the destination. The algorithm then changes faces, and begins exploring the boundary of the new face. If the topology graph is static, planar, and based on a unit disk graph, and node locations are accurate, face routing guarantees that one of two things always eventually happen: (1) the destination is encountered while exploring a face boundary, in which case the packet can be delivered, or (2) an entire face boundary is explored and no adjacent face is found that is closer to the destination. In the second case, the destination is always unreachable given the assumptions above. However, these assumptions are not always easy to guarantee [24].

Greedy and face routing are often combined producing so-called hybrid greedy-face routing protocols, or greedy-face-greedy protocols. These protocols start routing greedily until a local minimum is reached, at which point face routing is used to route around the face between the local minimum and the destination. Greedy routing then takes back over, until another dead end is reached. Many geographic routing protocols employ this greedy-face-greedy cycle but in a variety of different ways [19], [25], [26].

III. Related Works

Here we briefly discuss some of the most important and relevant geographic routing protocols. (For more details see [27].) GFG [26] was the first routing protocol to combine greedy and face routing. Upon reaching a local minimum GFG employs

face routing until reaching a node that is located closer to the destination than the node where face routing commenced, at which point it returns to greedy routing. GPSR [19] which is one of the most widely cited geographic routing protocols is largely a duplication of GFG with minor variations.

The GOAFR (pronounced "gopher") family of protocols build on a slightly modified version of face routing called Adaptive Face Routing (AFR) [25], [28]. One challenge with face routing is deciding in which direction a face boundary should be explored. It is possible that exploring in a clockwise manner could result a huge number of edges being explored before switching back to greedy routing, whereas exploring the same face in a counter-clockwise manner would return to greedy routing after just a couple of hops. Adaptive Face Routing (AFR) employs a bounding circle centered at the destination or an ellipse with foci at the source and destination, that can be adaptively increased in size as a packet attempts to determin in which direction a face boundary should be traced. Greedy Other Adaptive Face Routing Plus (GOAFR+), which most closely matches our proposed protocol, combines greedy routing and a version of AFR to arrive at a geographic routing protocol which is provably worst-case optimal and average-case efficient.

A method similar to ours is employed by [29], where the GPVFR algorithm exchanges path vector information between nodes. The resulting method outperforms both GOAFR+ and GPSR. GPVFR's use of topology information is similar to TAG's, but TAG does not require extra overhead to exchange information.

Geographic routing protocols (including GOAFR+ and GPVFR) generally assume a quasi-static topology, one that does not change for the duration of a particular routing activity. TAG overcomes this assumption, supporting topology changes even while packets are in flight.

IV. Topology Aware Geographic Routing (TAG)

We now describe Topology Aware Geographic Routing (TAG), our new geographic routing protocol. First we describe the protocol's header and its contents, and then we detail a base version of TAG. The base version is essentially a Greedy-Face-Greedy algorithm with the addition of a bounding circle much like that used in GOAFR+ [25]. Finally, in Sections V, VI, and VII we describe various extensions to the base protocol.

A. Shim Header

The only source of overhead for our routing protocol is the addition of an extra header to each packet's contents. This is a "shim header" like that used in Dynamic Source Routing (DSR) [30], or even MPLS [31] in wired networks. In contrast to protocols such as DSR, however, the size of our header does not grow with the length of the path or the size of the network. Table I lists and describes the contents of the header, while Table II lists the input parameters and default values.

Each node is assumed to have a single unique identifier (node ID). This value could be the IP-address associated with one of its interfaces, or any other node unique value. The header first includes a mode flag, M. This value enables each packet to independently store which of four modes it is operating in (greedy mode or one of three face routing modes). Next, the header includes a source node ID, s, a destination node ID, d, and the ID of the node where the packet last began tracing the boundary of a face (face start node ID), f. In addition, e stores the node ID of the second node visited (or to be visited) while tracing the boundary of a face. In conjunction with f, this value essentially encodes the first edge traversed on the current face. D and F store the destination location and the face start location respectively. The Boolean cw value facilitates the choice of tracing a face boundary in either a clockwise (right-hand rule) or a counter-clockwise (left-hand rule) direction. The radius of the packet's current bounding circle (which will be described shortly) is stored in r. Finally, t stores the time at which this packet began tracing the boundary of a face. If the nodes were assumed to be static, or even quasi-static, s, d, f, and t would not be necessary, and e could be replaced with a position value. The inclusion of these extra header fields is one of the costs of fully supporting node movement. Assuming 16-bit values suffice for location resolution, and the radius and time values also require only 16 bits each, 256 nodes could be supported with a 17-byte header, and 65,536 nodes with a 21-byte header.

TABLE I
Fields in the additional packet header

M	Mode: GREEDY, FACE-FWD, FACE-REV, or FACE-RETURN
s	Source Node ID
d	Destination Node ID
D	Destination Location (x, y)
f	Face Start Node ID
F	Face Start Location (x, y)
e	first edge traversed (ID of 2nd node visited)
cw	Direction (clockwise or counter-clockwise)
r	Radius of the bounding circle
t	time started face

TABLE II
TAG Parameters

ρ_0	Initial Bounding Circle Radius (used only in base version)	1.4
ρ	radius increase factor	$\sqrt{2}$
c	position exchange range	1.44 or 288 km
l	maximum link range	1.0 or 200 km
n	links per node (degree)	3
u	topology update rate	once per second

B. Algorithm Overview

Our base algorithm works by first greedily advancing until either reaching the destination or a local minimum (GREEDY). If the destination is reached, the packet is delivered, and our job is done. Otherwise, upon reaching a local minimum the

packet changes to face routing mode (FACE-FWD), and begins tracing, in a clockwise direction, the boundary of the first face encountered by a line extending from the local minimum node to the destination.

The packet begins routing around this face inside of an annulus (a rind-shaped object) centered at the destination. The space inside the inner circle of the annulus is closer to the destination than the node (f) where this packet started face routing, and the radius of the inner circle is equal to the distance from the face start position (F) to the destination (D). The outer circle is a bounding circle whose radius (r) was set when switching to face routing mode. If the packet crosses the inner circle it has made progress, and is now closer to the destination than when it began face routing. It may now safely attempt to switch back to GREEDY mode. If, on the other hand, the packet would cross the outer circle on its next hop, cw is flipped and the packet begins backtracking (in the counter-clockwise direction). The packet eventually arrives at the node where it began face routing (f), and continues tracing the face in the opposite direction (FACE-REV). As before, if the packet crosses the inner circle it transitions to GREEDY mode, but if on its next hop it would cross the outer circle, cw is again flipped (again exploring clockwise), and the packet transitions to the FACE-RETURN mode. This will cause the packet to backtrack all the way to the node where it started face routing (f). Finally, the radius of the outer circle (r) is increased, and the packet begins the entire face routing procedure again, starting in the FACE-FWD mode. While face routing, if the packet ever encounters the destination, it is delivered. If however, while in FACE-FWD mode, the packet detects that it has traced the entire boundary of a face, there is no path to the destination, and the packet is intentionally dropped. Dropping the packet at this point is actually a critical benefit of Face Routing, for if the packet were allowed to continue trying to reach the unreachable destination, it would waste valuable resources while it looped indefinitely.

C. Algorithm Details

Before each packet begins its journey, the position of its intended destination must be determined. Depending on the type of network and destination node, the destination position may need to be obtained from a separate location service such as [20].

For each new packet to be routed, the destination position and ID (D and d) are added to the shim header, along with the source node (current node) ID. Every packet starts in GREEDY mode (M= GREEDY). The shim header is attached to the packet and then the TAGFORWARD function, shown below, is called.

TAGRFORWARD is the main routing function, and is called at each hop along the path. If the packet has reached its destination the packet is delivered. Otherwise various methods are called depending on which mode the packet is in, with each method returning the ID of the node to which the packet should be sent next. For packets in GREEDY mode, the GREEDYORREVERTTOFACE function is called to either send the packet to the neighbor that is nearest the destination, or revert to face routing if the current node is a local minimum. If the packet is in FACE-FWD or FACE-REV the FACECOMMON method is simply called (detailed later). Lastly, for packets in the FACE-RETURN mode (in which the packet is returning to the face start node), the next node to be visited while backtracking to the face start node is determined by the RIGHTHANDFORWARD function. If, however, the packet has reached the face start node, the boundary circle is expanded and face routing is restarted by (EXPANDANDRESTARTFACE). Lastly, if the packet is in FACE-FWD or FACE-REV the FACECOMMON method is called.

```
1:  function TAGFORWARD(h, incomingIf)
2:      if h.d == self.id then
3:          deliver packet
4:      else
5:          switch h.M
6:              case GREEDY
7:                  next = GREEDYORREVERTTOFACE(h)
8:              case FACE-FWD
9:                  next = FACECOMMON(h, incomingIf)
10:             case FACE-REV
11:                 next = FACECOMMON(h, incomingIf)
12:             case FACE-RETURN
13:                 if h.f == self.id then
14:                     next = EXPANDANDRESTARTFACE(h, IncomingIf)
15:                 else
16:                     next = RIGHTHANDFORWARD(h, false)
17:                 end if
18:         forward to next and call TAGFORWARD
19:     end if
20: end function
```

The FACECOMMON method first determines the proposed next node to be visited according to the right-hand rule. If the packet is (1) in FACE-FWD mode, (2) at the face start node f, and (3) expected to visit node e next, the packet has made a full loop around the face, and it is mercifully dropped because there is no route to the destination. If instead, the packet reaches a node that is nearer the destination than the face start node, it calls GREEDYORREVERTTOFACE. This method starts greedily routing unless this node is found to be another local minimum, in which case face routing is started on the face between this node and the destination. Ignoring for now, the highlighted code, if the proposed next node lies outside the bounding circle then if the packet was in FACE-FWD mode, we REVERSETHEDIRECITON the packet was traveling in (changing directions, for example, from clockwise to counter-clockwise) and change to FACE-REV mode. If instead the packet was already in FACE-REV mode we also REVERSETHEDIRECTION in which the packet is traveling and change to FACE-RETURN mode. In some cases the packet is already at the face start node so the FACE-RETURN mode is skipped and the bounding circle is immediately expanded and face routing restarted with EXPANDANDRESTARTFACE.

V. TOPOLOGIES THAT ARE NOT UNIT DISK GRAPHS

Our face routing algorithm assumes that at some point while traveling around any face one of three events will occur: (1) the packet will arrive at the destination, (2) the packet will

```
1:  function FaceCommon(h, incomingIf)
2:      next = RightHandForward(h, false)
3:      if h.M == FACE-FWD and self.id == h.f and next == h.e then
4:          return failure                    ▷ No route to destination, drop
5:      end if
6:      if Dist(self.position,h.D) <Dist(h.F,h.D) then
7:          return GreedyOrRevertToFace(h)    ▷ Fall back to greedy
8:      end if
9:      if Intersects(F, D, self.position, next.position) then
10:         return StartFace(h)
11:     end if
12:     if Dist(next.position, h.D) >h.r then
13:         next = ReverseTheDirection(h, incomingIf)
14:         if h.M == FACE-FWD then
15:             h.M = FACE-REV
16:             return next
17:         end if
18:         if h.f == self.id then
19:             return ExpandAndRestartFace(h)
20:         end if
21:         h.M = FACE-RETURN
22:         return next
23:     end if
24:     return next
25: end function
```

arrive at a node that is closer to the destination than the node at which face routing began, or (3) only if there is no route to the destination, the packet will arrive back at the node at which it started, having traversed the entire face. The implied assumption in these cases is that either there is a node closer to the destination on the face that stands between the start node and the destination, or the destination is unreachable. This assumption is true in a unit disk graph, but what of the scenario shown in Figure 2 where **S** is the source of a packet; **D** the destination, and the circle encompasses the space which is closer to **D** than **A**? Note that **B** and **C** are both outside the large circle, and are both further from **D** than **A**. This also means that A is a local minimum. If the unit disk graph assumption were employed here, **A** and **D** would have to be connected, since they are closer to one another than **C** and **D** (which are presumably closer to one another than 1 unit). Since this is not a unit disk graph, tracing the boundary of the face from **A** to **B** to **C** bring us back to **A** without finding any node which is closer to **D**. This violates the assumption of our proposed routing scheme, since there is a route to the destination through **B** or **C**.

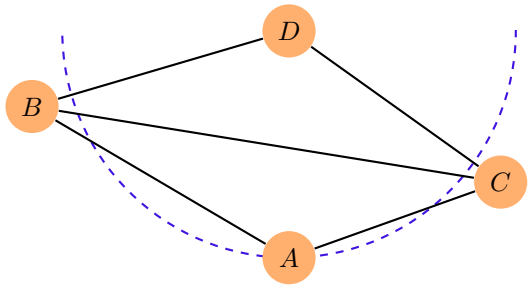

Fig. 2. Example of a topology that violates the Unit Disk Graph assumption

To overcome this issue we simply add three lines to the FaceCommon function (10-12 highlighted). As the packet moves around the face, we intersect the line segment representing the next edge to be traversed with the line segment extending from the face start position to the destination position in a manner similar to basic face routing [22]. If an intersection is detected, we simply restart face routing at the current node. The packet will then seek to navigate the face between the current node and the destination. Depending on the geometry of the graph, the next face traversed may not be the next face encountered by the line which intersected the edge, however, progress towards the destination has been made. This very simple change to our algorithm allows us to conquer the potential issues with non-unit disk graphs.

VI. Topology Awareness

We now propose an enhancement to our base algorithm for cases where a node has or can obtain local information regarding the topology of the network. The base algorithm outlined above has a few key decision points, and we propose using any topology information available to help make the best possible decisions. Note that in general performance can be improved without global topology information, and even when local topology information differs from the actual global topology. Nodes need not learn the entire topology to benefit from the topology awareness enhancements!

When a packet initiates face routing, it can proceed in a clockwise direction (right-hand), or decide instead to travel in a counter-clockwise direction. If topology information is available, it can be used to make a better choice about which direction to head. Similarly, if some topology information is available, the initial radius of the bounding circle can be set intelligently. Topology information can also assist in the selection of the next hop for greedy routing, enabling a more global greedy choice. The immediate next greedy hop, however, must always make progress towards the destination.

To enable this enhancement local Topology information can be exchanged periodically via some other protocol, incurring an overhead penalty. However, in the case where a topology management protocol is employed, the local topology information may already be known by each node. In protocols such as LTRT [32] nodes use the positions of their neighbors to compute their own view of the local topology and make local transmission power decisions, ultimately controlling the network topology. Similarly, networks supported by directional links can use similar distributed topology management protocols to determine where to point their directional links. The nodes in these networks already have the local topology information necessary for to benefit from this enhancement without any extra overhead.

Since this information is essentially only used to provide "hints" to the routing protocol, it doesn't even have to be correct. If a packet starts face routing in the non-optimal direction, it will eventually turn around and go the other way. If a chosen bounding circle size is too small, it will eventually be increased. This is important, because in the case where local topology management protocols are used, the local topology information does not always match the global topology.

The StartFace function shows how the topology aware aspects can be inserted into our face routing function. This function is called any time a packet is to begin routing around a face. Its purpose is to set up the header values correctly for face routing. The highlighted lines are the only lines added or changed to support topology awareness. We call two nodes connected if the local topology graph contains a path between them, and we call two nodes directly connected, if they are connected with a path of length 1 hop. The FindConnectedNodeNearestPosition function uses the available local topology graph to find and return the ID of the node nearest the destination to which there is a path from this node.

An additional function, RightWayToReach takes the nearest node ID and determines which direction (counter-clockwise or clockwise) face routing should proceed to efficiently reach the nearest node. This is far better than our base protocol, where faces boundaries were always initially traced in a clockwise direction. Lastly, the SmartRadius function determines the appropriate radius value for the bounding circle. Previously, as in GOAFR+ this value was set to the product of the distance from F to D and an initial radius factor. The SmartRadius function determines which node on the path to the "nearest" node is furthest from the D and uses that distance as the radius value, ensuring that expected path can be traveled without increasing the radius. In cases where the "nearest" node is the current node, the radius is set such that the bounding circle encompasses the entire area within which the local topology is known.

```
1:  function StartFace(h)
2:      h.M = FACE-FWD
3:      h.f = self.id
4:      h.F = self.position
5:      nearest = FindConnectedNodeNearestPosition(h.D)
6:      h.cw = RightWayToReach(nearest)   ▷ before always clockwise
7:      h.r = SmartRadius(nearest, h.D)   ▷ before, Dist(F,D) * ρ0
8:      next = RightHandForward(h, true)
9:      h.e = next
10:     h.t = time.Now()
11:     return next
12: end function
```

VII. Dealing with Topology Changes

Standard face routing can fail miserably when the movement of nodes causes changes in the topology. For example, Figure 3 shows a topology of 5 nodes. Assume that initially the edge between nodes A and C does not exist. A packet beginning face routing at node S and traveling clockwise would visit node A, then B. Now assume that while the packet is at node B, the network topology changes and the edge between A and C is added. Upon reaching node C the packet would use the right-hand rule to determine where to go next, and would find that the new edge should be traversed, sending the packet to A, not to node E. A would again forward the packet to B, and B to C, and the packet would be stuck in a potentially infinite routing loop.

Most other geographic routing protocols assume a static or quasi-static network topology, suggesting that the routing

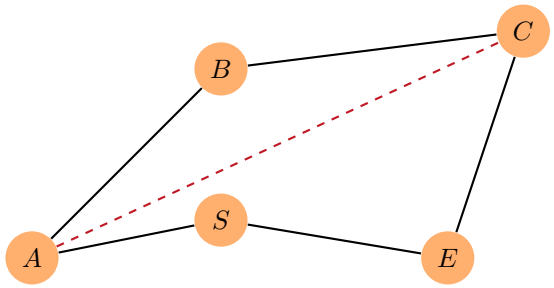

Fig. 3. Example of a routing loop formed when the topology graph changes while face routing

is so fast compared to the node movement that no topology changes happen while a packet is in flight. This is obviously not realistic, as in a busy network there will always be packets flowing, some of which will be affected by a topology change. To our knowledge we are the first to propose a greedy-face-greedy style geographic routing protocol which can operate correctly on topology graphs which are constantly changing.

A. Approach

Our approach is to essentially take a snapshot of a face (in a distributed manner) each time a packet begins tracing a face boundary. The snapshot, not the potentially changed current topology, is then used to determine how to forward the packet around the face. Back to our example in Figure 3, assume the packet is again at node **S** and is begin routed around the face in a clockwise direction. This time, however, the packet follows the face as it existed at the time face routing began. The packet would visit node **A**, then **B**, then **C**. Node **C** would realize that at the time the face traversal began, its connection to node A did not exist, and so would correctly forward the packet to **E**, avoiding the potential routing loop.

What if, however, instead of the link between **A** and **C** being added to the graph, it was instead taken away? Assume the packet starts at node S, where it takes a virtual "snapshot" of the face that includes the connection from **A** to **C**, and while traveling to node **A**, the link from **A** to **C** goes away. We need a way to get the packet to node C and allow it to continue to traverse the face, but the direct path no longer exists. The packet could, instead, take the indirect path to **C**, traveling through node **B**. In Section VII-C we will describe how this can be done.

B. Storing the Snapshot

To support topology changes while a packet is in-flight we propose sending a packet along the face as it existed at a previous point in time. To achieve this, we add a time field, t to the packet header, that corresponds to the time at which the current face boundary began being traced. In addition, every node stores in a small buffer the positions and IDs of nodes they have recently (in the last few seconds) been directly connected to. The amount of storage required is essentially controlled by the maximum degree of each node. When a node needs to determine where to forward a packet as it travels

around a face, it need only use the time stored in the packet to look up the IDs and positions of nodes it was connected to at the given time. It then makes the routing decision it would have been made at the time the face routing began, and the virtual "snapshot" was taken.

C. Via-Points

Our proposed method works until the rare situation is encountered where a node is no longer directly connected to the next node on the face snapshot. To overcome this, we note that our routing protocol allows a packet to be routed to any node whose position and ID are known. Since the position and ID of the next node along the boundary of the snapshot face are known, we forget about our ultimate destination, briefly, and concentrate on routing the packet to the next stop along the face, which we call the "via-point". Our shim header becomes a stack of shim headers, and we push on a new header whose destination is the via-point. The packet is now forwarded like a new packet, toward the via-point. Once the via-point is reached, the header is popped off, and the packet is ready to continue routing around the original face, having "virtually" traced the face edge that no longer exists.

VIII. Simulation Results and Evaluation

To evaluate TAG we present the results of several simulations, which demonstrate the advantages of TAG compared with other geographic and MANET routing protocols. We first report the results of simulating our protocol in a theoretical square field (as in [25]), randomly placing static nodes on a plane in a square measuring 20 units on each side. Next we briefly describe a study comparing TAG and OLSR. Finally, we report results from a more realistic simulation, where mobile nodes trace the paths traveled by over 600 actual aircraft moving at hundreds of miles per hour.

A. Simulation Environment

We use the ns-3 [33] network simulator (version 3.21) for our evaluation experiments. We utilize the default ns-3 implementation of OLSR, and our own implementations of GFG and GOAFR+ (these latter protocols are not included with ns-3). Since our goal at this stage is to measure the performance of our routing protocol without interference from the environment or other layers of the stack, we assume perfect wireless links and a collisionless MAC layer. By perfect links we assume that no packets are dropped at the link layer, and that the directional links are able to re-point and reconnect instantly. Instant re-connection could essentially be accomplished by doubling the number of links and allowing one to remain connected while it's counterpart established the next connection. A collisionless MAC layer is also not so unrealistic for directional links where interference is less common given the directional nature of the connections. Further, we assume (as in [25]) that all position information required by the geometric routing protocol is available without extra communication overhead. This includes a node's knowledge of its position and the positions of nearby nodes (nodes within the position exchange range c of 288 km or 1.44 units), as well as each source node's knowledge of the destination positions of its flows. Remember for our airborne network application the positions of nearby nodes are already regularly updated via an independent system. Finally, we assume all nodes have a synchronized time reference, such as GPS.

B. Varied Density with a Static Topology

We first compare the performance of TAG, TAG with topology awareness disabled, GOAFR+(ρ_0=1.4, σ=$\frac{1}{100}$), and GFG on a static topology using the hop stretch metric [29]. Hop stretch is the ratio of the number of hops required by a routing protocol to reach the destination d to the number of hops on the shortest path (in terms of hops) between s and d. Lower hop stretch values imply better performance for the geographic routing protocol.

For this experiment the number of nodes is increased from a density value of 1 to a density value of 20 nodes per unit disk. A density value of 5 implies that on average each node will have 4 other nodes within communication range (within its unit disk). For reference, the density values 1 and 20 correspond to 128 and 2547 total nodes, respectively. For each run, a source node, s, and destination node, d, were randomly selected from the simulated nodes. A single UDP packet was then sent from the selected source node towards the destination node. The number of hops taken by the simulated routing protocol was recorded, as were the number of hops in the shortest path from the source to the destination. Figure 4 shows a plot of the resulting hop stretch values for the various protocols.

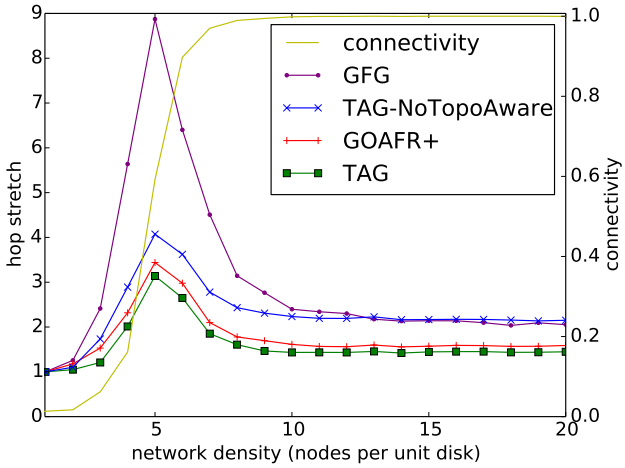

Fig. 4. Comparison of hop stretch with increasing node density

Each point on the figure represents the mean value for 2000 trials (each with a random set of node positions and source, destination pair). Also plotted on the right y-axis is the connectivity of the network at the given densities. Note that below a critical density of about 5 nodes per unit disk the source and destination have a path between them (are connected) less than half of the time. Nodes that are able to connect to one another at these densities generally

are within a few hops of one another. Hop stretch values for disconnected source/destination pairs are not included, yielding low hop stretch values for the lower densities. Once a majority of source/destination pairs are connected the hop stretch values become very large. This is because, although the nodes are connected, any path connecting them must wind its way through the "nearly disconnected" network. Finally the hop stretch values basically level off. TAG makes a slight improvement on the performance state-of-the-art GOAFR+ protocol, and performs significantly better than the basic Greedy-Face-Greedy protocol (GFG). The mean hop stretch for TAG is 1.73, compared with 1.97 for GOAFR+ and 3.57 for GFG. This improvement is credited to the effective use of the local topology information at each node.

C. Comparison with OLSR

We now describe a simple experiment comparing our routing protocol with Optimized Link State Routing (OLSR) [34]. We simulate 24 mobile nodes tracing the actual paths taken by aircraft on July 9, 2015. The topology management protocol described in Section II-C is used to form a topology assuming three directional links per aircraft and a maximum air-to-air link range of 200 km. The node movement and topology management is such that there always exists a topological path between every pair of nodes in the network. We simulate a low-rate (10 Kbps) UDP flow between every pair of mobile nodes in the network for 1 hour. The default Hello interval of 2.0 seconds is used for OLSR. TAG successfully delivers every packet achieving a PDR of 1.0, while OLSR is only able to successfully deliver 94.2% of the packets (a PDR of 0.942). In addition, OLSR sends over 281,000 overhead packets (a total of 53 Megabytes). Tag has no overhead besides the small additional header attached to each packet. Similar OLSR experiments with about 600 nodes have yielded PDRs as low as 0.44.

D. Realistic Airborne Network with Mobility

We now present results from another experiment utilizing the real mobility data of a subset of all commercial air-traffic in the United States (several hundred aircraft), and compare the performance of the chosen geographic routing protocols in this environment. The density of the network increases as the simulation progresses from 3.11 to 10.47 nodes per unit disk. In striving for perfect PDR, we do not limit the number of hops each packet may take (deactivating the IP TTL countdown). Figure 5 shows the cumulative distribution function (CDF) of the mean numbers of hops for each of 1000 individually simulated flows in the network. The curve for the optimal number of hops (given a global view of the network) is also plotted for comparison. TAG performs better than GOAFR+, especially in the tail of the distribution, with the worst performing flow requiring only 111 hops on average for TAG, but over 2,000 hops for GOAFR+ and nearly 3,000 hops for GFG. The high numbers of hops are a result of either cases where a packet gets stuck in a loop until the topology changes allowing progress, or cases where the basic geographic protocols make bad choices or must backtrack several times while expanding the bounding circle. Also notable, is the fact that many flows using GOAFR+ were unable to guarantee delivery, with some achieving PDRs lower than 0.96. The large numbers of hops in the worst case, highlight how helpful a small amount of local topology information can be, and the importance of supporting full mobility.

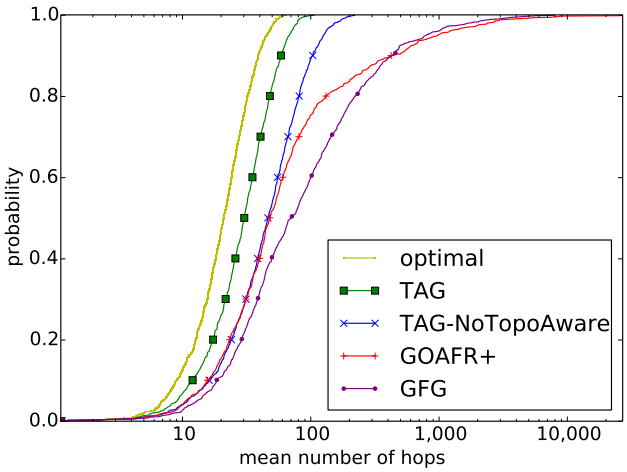

Fig. 5. CDF of the mean numbers of hops in a realistic scenario

IX. Future Work

These results are promising, but more work must be done to fully investigate TAG's performance, especially when higher-layer protocols are used. Preliminary results indicate that TCP functions well with TAG, and we hope to present more detailed results soon. We suspect that TAG's performance may still be improved upon by using probabilities to determine which direction a packet should be forwarded when even the available local topology information doesn't induce a preference. We plan to make this and other optimizations to the protocol going forward.

Finally, a more thorough temporal analysis of the protocol would be helpful. It could answer questions such as: How often are position updates are necessary? and How delayed can destination positions be?

X. Conclusion

We have introduced TAG a new geographic routing protocol that advances the state of the art by (1) supporting topologies that are not unit disk graphs, (2) using local unreliable topology information to make better geographic forwarding decisions, and (3) supporting full mobility of nodes without risk of looping. We detailed the routing protocol, and then presented simulated comparisons of TAG in a theoretical network, and a real-world large-scale mobile airborne network. TAG outperforms GOAFR+, GFG, and OLSR in our simulations. We envision a day when mobile ad hoc networks will

connect aircraft, spacecraft, watercraft, and vehicles, into high-capacity large-scale networks. Efficient routing protocols for these unique futuristic networks are needed, and we believe that TAG is a step towards that future.

REFERENCES

[1] J. Redi and R. Ramanathan, "The DARPA WNaN Network Architecture," in *2011 - MILCOM 2011 Military Communications Conference*, Nov 2011, pp. 2258–2263.

[2] DARPA, "Broad Agency Announcement BAA07-07 WNaN Adaptive Network Development (WAND)," Feb. 2007. [Online]. Available: http://www.federalgrants.com/WNaN-Adaptive-Network-Development-WAND-8854.html

[3] T. Simonite, "10 breakthrough technologies: Project loon," *MIT Technology Review - Best in Tech: 2015*, pp. 20–25, 2016.

[4] E. Teller and W. Patrick, "Balloon clumping to provide bandwidth requested in advance," Nov. 21 2013, wO Patent App. PCT/US2013/035,959. [Online]. Available: http://www.google.com/patents/WO2013173002A1?cl=en

[5] H. Frey, "Scalable geographic routing algorithms for wireless ad hoc networks," *IEEE Network*, vol. 18, no. 4, pp. 18–22, July 2004.

[6] "Aircraft situation display to industry: Functional description and interface control document," Volpe Center, Tech. Rep. Version 4.0, August 2000, report no. ASDI-FD-001.

[7] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Ad-hoc networks beyond unit disk graphs," in *Proceedings of the 2003 Joint Workshop on Foundations of Mobile Computing*, ser. DIALM-POMC '03. New York, NY, USA: ACM, 2003, pp. 69–78. [Online]. Available: http://doi.acm.org/10.1145/941079.941089

[8] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "On the pitfalls of geographic face routing," in *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, ser. DIALM-POMC '05. New York, NY, USA: ACM, 2005, pp. 34–43. [Online]. Available: http://doi.acm.org/10.1145/1080810.1080818

[9] ——, "Geographic routing made practical," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 217–230. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251219

[10] H. Boche, A. Bourdoux, J. R. Fonollosa, T. Kaiser, A. Molisch, and W. Utschick, "Smart antennas: state of the art," *IEEE Vehicular Technology Magazine*, vol. 1, no. 1, pp. 8–17, March 2006.

[11] Q. Balzano, J. Rzasa, S. Milner, and C. Davis, "High capacity tactical networks with reconfigerable, steerable, narrow-beam agile point-to-point rf links," in *Military Communications Conference, 2007. MILCOM 2007. IEEE*, Oct 2007, pp. 1–7.

[12] L. Stotts, N. Plasson, T. Martin, D. Young, and J. Juarez, "Progress towards reliable free-space optical networks," in *Military Communications Conference, 2011 - MILCOM 2011*, Nov 2011, pp. 1720–1726.

[13] R. Devaul, E. Teller, C. Biffle, and J. Weaver, "Balloon network with free-space optical communication between super-node balloons and rf communication between super-node and sub-node balloons," Jul. 18 2013, wO Patent App. PCT/US2013/020,705. [Online]. Available: https://www.google.com/patents/WO2013106348A1?cl=en

[14] M. Wohlsen, "Facebook drones to battle google balloons in the war of airborne internet," *Wired*, March 2014. [Online]. Available: http://www.wired.com/2014/03/facebooks-drones-launch-race-airborne-internet/

[15] B. Newton, J. Aikat, and K. Jeffay, "Analysis of topology algorithms for commercial airborne networks," in *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, Oct 2014, pp. 368–373.

[16] *Automatic Dependent Surveillance-Broadcast (ADS-B) Out Performance Requirements To Support Air Traffic Control (ATC) Service; Final Rule*, Department of Transportation - Federal Aviation Administration Std. 14 CFR Part 91, May 2010. [Online]. Available: https://www.gpo.gov/fdsys/pkg/FR-2010-05-28/pdf/2010-12645.pdf

[17] B. Epstein and V. Mehta, "Free space optical communications routing performance in highly dynamic airspace environments," in *Proceedings of Aerospace Conference, IEEE.*, vol. 2, 2004, pp. 1398–1406 Vol.2.

[18] B. Newton, J. Aikat, and K. Jeffay, "Simulating large-scale airborne networks with ns-3," in *Proceedings of the 2015 Workshop on Ns-3*, ser. WNS3 '15. New York, NY, USA: ACM, 2015, pp. 32–39. [Online]. Available: http://doi.acm.org/10.1145/2756509.2756514

[19] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 243–254. [Online]. Available: http://doi.acm.org/10.1145/345910.345953

[20] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 120–130. [Online]. Available: http://doi.acm.org/10.1145/345910.345931

[21] G. G. Finn, "Routing and Addressing Problems in Large Metropolitan-Scale Internetworks," University of Southern California ISI Research Report ISI/RR-87-180, March 1987.

[22] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *in proc. 11th Canadian Conference on Computational Geometry*, 1999, pp. 51–54.

[23] D. Q. O'Brien, "Maze demystified," *New York Times*, July 26, 1989. [Online]. Available: http://www.nytimes.com/1989/07/28/opinion/l-maze-demystified-303989.html

[24] H. Frey and I. Stojmenovic, "On delivery guarantees and worst-case forwarding bounds of elementary face routing components in ad hoc and sensor networks," *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1224–1238, Sept 2010.

[25] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '03. New York, NY, USA: ACM, 2003, pp. 267–278. [Online]. Available: http://doi.acm.org/10.1145/778415.778447

[26] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, ser. DIALM '99. New York, NY, USA: ACM, 1999, pp. 48–55. [Online]. Available: http://doi.acm.org/10.1145/313239.313282

[27] F. Cadger, K. Curran, J. Santos, and S. Moffett, "A survey of geographical routing in wireless ad-hoc networks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 621–653, Second 2013.

[28] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, ser. PODC '03. New York, NY, USA: ACM, 2003, pp. 63–72. [Online]. Available: http://doi.acm.org/10.1145/872035.872044

[29] B. Leong, S. Mitra, and B. Liskov, "Path vector face routing: geographic routing with local face information," in *13TH IEEE International Conference on Network Protocols (ICNP'05)*, Nov 2005, p. 12.

[30] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, pp. 153–181.

[31] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet Requests for Comments, RFC Editor, RFC 3031, January 2001. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3031.txt

[32] K. Miyao, H. Nakayama, N. Ansari, and N. Kato, "Ltrt: An efficient and reliable topology control algorithm for ad-hoc networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 12, pp. 6050–6058, December 2009.

[33] G. F. Riley and T. R. Henderson, "The ns-3 network simulator." in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Gnes, and J. Gross, Eds. Springer, 2010, pp. 15–34. [Online]. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-12331-3_2

[34] "Optimized link state routing protocol (olsr)," United States, 2003.