GeoCast - Geographic Addressing and Routing *

Julio C. Navas and Tomasz Imielinski
{navas,imielins}@cs.rutgers.edu
Computer Science Department
Rutgers, The State University
Piscataway, NJ 08855

Abstract

In the near future GPS will be widely used, thus allowing a broad variety of location dependent services such as direction giving, navigation, etc. In this paper we propose and evaluate a routing and addressing method to integrate geographic coordinates into the Internet Protocol to enable the creation of location dependent services. The main challenge is to integrate the concept of physical location into the current design of the Internet which relies on logical addressing.

1 Introduction

In the near future Global Positioning System (GPS) cards will be deployed in each car and possibly in every user terminal. User's *location* will become information that is as common as the *date* is today; getting input from GPS, when outdoors, and other location providing devices, when indoors. Availability of location information will have a broad impact on the application level as well as on network level software.

Below, we list some of the possible new services and functionalities:

- Geographic messaging: sending a message selectively only to specific subareas defined by latitude and longitude. For example, sending an emergency message to everyone who is currently in a specific area, such as a building, train station, or a highway.
- Geographic Advertising and Geographic Services: Advertising and providing a given service only to clients who are within a certain geographic range from the server (which may be mobile itself); such as everyone within a mile from the server.
- "Who is around" service: finding out who is currently present in a specific geographic area defined by an arbitrary polygon.

To support such applications, location should become a first class citizen, starting from the IP level and proceeding all the way to the application layer. Routing protocols for geographic messages need to be developed. Furthermore, the geographic destination for a message should not be confined to a single point, but should be able to be specified as any arbitrary polygon.

The Geographic Messaging Project researched possible system designs for accomplishing geographic messaging. Three designs were looked at: a Domain Name System approach, a multicast approach, and a geographic routing approach. The details of these approaches are specified in RFC 2009 [5].

This is a DARPA sponsored Integrated Technology Demonstration (ITD) which is being led by Rutgers DataMan Lab within the GloMo (Global Mobile) program. Under the DARPA experiment, we have implemented a prototype of the geographic routing system described in [5] and are in the process of evaluating it. The immediate plans call for the deployment of an experimental national network capable of routing geographic messages with nodes at Rutgers DataMan Lab, CECOM in Fort Monmouth (U.S. Army), University of California at Santa Cruz, and Bolt, Beranek, and Neumann

*This research work was supported in part by DARPA under contract number DAAH04-95-1-0596, NSF grant numbers CCR 95-09620, IRIS 95-09816 and Sponsors of WINLAB.

(BBN). The geographic routing system implementation will soon be made available to the DARPA GLOMO participants.

The main objective of this paper is to report the efficacy of the geographic routing approach. Section 2 talks about some background and related work. Section 3 discusses the addressing model used for geographic routing. Section 4 gives an overview of the geographic routing system. Section 5 discusses some implementation issues of the router itself. Section 6 describes the experiments conducted and analyzes the results. Section 7 talks about one future research direction for geographic routing. Finally, Section 8 draws some conclusions.

2 Background and Related Work

Linking an IP Address with a geographical location has been of interest for quite some time already. The recent redesign of the Internet Protocol (IP) [2] and the advent of the Global Positioning System [9] gave a new stimulus for this work.

The first attempt to design a system that actually routes packets according to their geographic destination and the work that is closest to ours is Cartesian Routing by Gregory G. Finn [4].

In the proposed redesign of the IP protocol [2], IP Address Type Space was specifically allocated for geographic addresses [3] [7]. These proposed IPv6 geographical addresses are an abstract geographically-nested hierarchy with no relation or connection to any underlying coordinate system.

In [3] and [7] the sender of a "geographic message" would be unicasting messages only to such hosts which have geographic addresses. The methods in this paper attempt to provide the more general ability of sending a message to all recipients within a geographical area, regardless of whether the hosts have geographical addresses or not.

Xerox's PARC lab pioneered early work on location dependent services [8].

3 Addressing Model

Two-dimensional GPS positioning offers latitude and longitude information as a two dimensional vector, < latitude, longitude >, where longitude ranges from -180 (west) to 180 (east), and latitude ranges from -90 (south) to 90 (north).

Thus < 40.48640, -74.44513 > is an example of the GPS coordinates for the town of New Brunswick, New Jersey, U.S.A.

Assuming the use of single precision floating-point numbers, four bytes of addressing space are necessary to store latitude and four bytes are also sufficient to store longitude. Thus a total of eight bytes are necessary to address the whole surface of earth with precision down to 0.1 mile!

In the future, once we have more experience using GPS receivers and are able to determine the optimal number of significant digits necessary for geographic routing, we will most likely switch to fixed-point integers because of their greater efficiency.

3.1 Using a Geographic Destination Address

A geographic destination address would be represented by some closed polygon such as:

- point
- circle(center point, radius)
- polygon(point(1), point(2), ... , point(n-1), point(n), point(1))

where each vertex of the polygon is represented using geographic coordinates. This notation would be used to send a message to anyone within the specified geographical area defined by the closed polygon.

For example, if we were to send a message to city hall in Fresno, California, we could send it by specifying the geographic limits of the city hall as a series of connected lines that form a closed polygon surrounding it. Therefore the address of the city hall in Fresno could look like: polygon([36.80,-119.80], [36,85,-119.76], ...).

4 Routing Geographically

In our Internet Engineering Task Force (IETF) Request For Comments (RFC) [5], we detailed three methods for achieving geographically-routed messages: a geographically-aware router solution, a multicast solution, and a Domain Name Service (DNS) solution. At this point in time, we decided to implement and evaluate the geographically-aware router solution and not the multicast solution because of the multiplicity of proposed multicast protocols. It is not clear which multicast proposal will become dominant in the future. Furthermore, it is not clear whether the future dominant multicast protocol would be able to handle peculiarities of mobile computing such as mobile routers. As such, the

geographically-aware router solution best lent itself to future research endeavors in geographic routing because we would have full control of the routers themselves.

The geographically-aware router solution uses the polygonal geographic destination information in the geographic message header directly for routing. Geographic routing is implemented in the Application layer in a manner similar to the way multicast routing was first implemented. That is, a virtual network which uses geographic addresses for routing will be overlayed onto the current IP internetwork. We would accomplish this by implementing our own routers which are geographically-aware. These routers would use IP tunnels to transport data packets through areas which do not support geographic routing.

4.1 The Components of the Geographic Routing Method

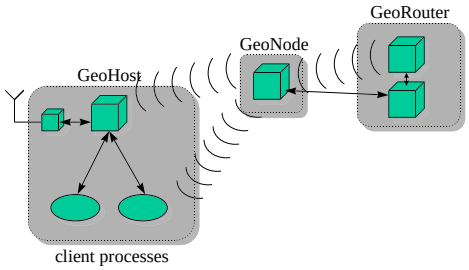

Figure 1: All of the Components of the Geographic Routing System

The system is composed of three main components: the GeoHosts, GeoNodes, and the GeoRouters. See figure 1.

4.1.1 GeoRouter

Geographic Routers (GeoRouter) are in charge of moving a geographic message from a sender to a receiver. The current prototype GeoRouter has the following capabilities:

1. Basic multi-hop geographic routing.

2. Automatic detection of multiple network interfaces, type of network interface (whether wired or wireless), other GeoRouters, and base station GeoNode programs.

3. Automatic configuration of routing tables based on detected information. Assumes a flat network.

4. Manually configurable to do hierarchical network routing.

5. Ability to tunnel through areas that do not (yet) provide geographic routing.

See section 4.2 for an overview of the geographic routing algorithm.

4.1.2 GeoNode

A GeoNode is an entry/exit point for the routing system. The main function of the GeoNode is to store incoming geographic messages for the duration of their lifetime and to periodically multicast them on all of the subnets or wireless cells to which it is attached. Each subnet and each wireless cell will have at most one GeoNode. The lifetime of a geographic message is specified by the sender of the message. Message lifetimes are necessary because the receivers of geographic messages may be mobile and may possibly arrive at the message destination just after the geographic message first arrives.

Since, most likely, there will be several geographic messages residing in a GeoNode at one time, the multicasting of the various messages will be scheduled. The scheduling algorithm will take into account the size of the message, the priority of the message, and the speed of the subnet's transport medium. Clients wishing to receive geographic messages would then tune in to the appropriate multicast group to receive them.

4.1.3 GeoHost

This daemon is located on all computer hosts which are capable of receiving and sending geographic messages. Its role is to notify all client processes about the availability of geographic messages, the host computer's current geographic location, and the address of the local GeoNode.

4.2 Routing Overview

Sending a geographic message involves three steps: sending the message, shuttling the message between routers, and receiving the message.

4.2.1 Sending Geographic Messages

In order to send a geographic message, the programmer would use the Geographic Library routine SendToGeo(). The function will, first, contact the local GeoHost Daemon and query it for the IP address of the local GeoNode. It will then send the message directly to the GeoNode, which, in turn, will simply forward the message to the local GeoRouter.

4.2.2 Router Actions

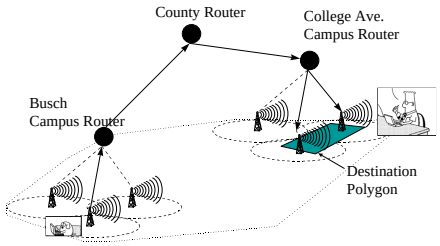

Figure 2: Geometric Routing Example

Once the datagram arrives at a GeoRouter, it must first determine if it services any part of the area of the destination polygon. To do this, the router determines if the destination polygon and the router's service area polygon intersect[1] [6] each other. If not, then the router simply sends the message to its parent router.

However, if the polygons intersect each another, then the router does service the area described by the intersection polygon. First, if the polygons only partially intersect, then the router only services part of the target area. Therefore, it sends a copy of the message to it's parent router for further routing beyond its own reach. The router now tests each child node's service area to see if it intersects the destination polygon. Those that do will be sent the geographic message.

In Figure 2, a user on Busch Campus wishes to send a message to the destination polygon on the College Ave. Campus. Upon sending the message, it is passed to the Busch Campus router. By using polygon intersection, the router determines that it does not service the target area, so it forwards the message to its parent, the county router. Using the same algorithm, the county router decides that its child node, the College Ave. router services the destination area and forwards the message to it. The College Ave. router, in turn, forwards the message to the two GeoNodes which control the target area. These two GeoNodes then deliver the message to all of the users in the target area.

The router keeps a cache of the latest geographic message packets and their destinations. When a router receives a geographic message packet, it will use the incoming packet's Message Id as a key into the cache. If this is not the first packet to arrive for this destination and if the timer on the cache entry has not yet expired, then the cache will return a list of all of the next hop addresses to which copies of the packet must be sent.

4.2.3 Receiving Geographic Messages

Once a geographic message has been sent to a GeoNode from a geographic router, the receive process can begin. The GeoNode will store the message locally, assign a multicast group to it, and append it to the list of available messages. Periodically, the GeoNode will multicast the list of available messages on a well-known group address. Also, the GeoNode will periodically multicast the message on its assigned multicast group. The GeoHost daemons will receive the list of available messages from the GeoNode and determine if the host computer is located inside any of the messages' destination polygons. When a client process executes a RecvFromGeo() call from the Geographic Library, the function will join the appropriate multicast address and receive the geographic message itself.

5 Router Implementation

5.1 Format of a Geographic Message Header

A geographic message header has the following format and order:

Message ID	(9 bytes)
Port Number	(2 bytes)
Lifetime	(4 bytes)
Flags	(1 bytes)
Area Type	(1 bytes)

followed by one of the following according to the Area Type:

[1] Detecting polygon intersection takes O(n) time.

- Type = 1 (Point)
  Latitude (8 bytes)
  Longitude (8 bytes)

- Type = 2 (Circle)
  Latitude (8 bytes)
  Longitude (8 bytes)
  Radius (8 bytes)

- Type = 3 (Polygon)
  Number of Points (2 bytes)
  Latitude[i] (8 bytes)
  Longitude[i] (8 bytes)

Followed by the message data body of arbitrary length and content. Note that the Message ID is of the form:

Host Computer IP Address	(4 bytes)
Process ID	(4 bytes)
Message Sequence Number	(1 byte)

For example, if the geographic destination is described by a point, the header would look like figure 3.

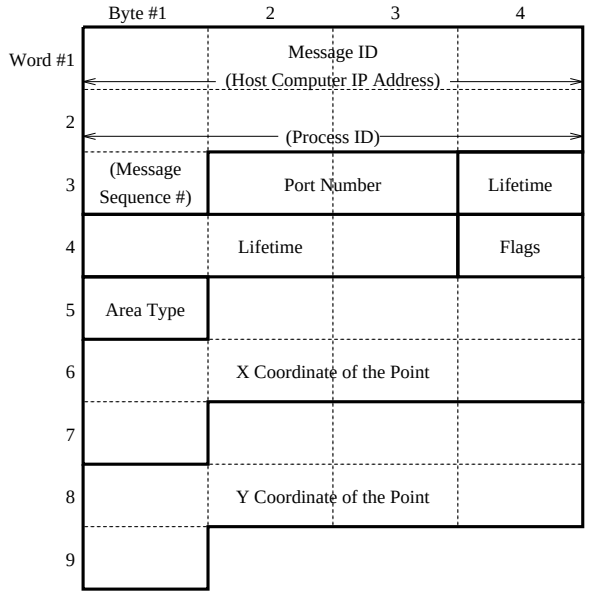

Figure 3: Geographic Routing Packet Header for a Point Destination

5.2 How the Routing Table is Implemented

The routing table, which is the heart of the geographic router, is comprised of three main areas: a cache, a polygon describing the router's service area, and lists of child and parent nodes.

5.2.1 Cache

In order to allow for rapid access to the individual cache items while, at the same time, not wasting memory space by preallocating a set hash table size, the router's cache is implemented as a dynamic hash table with chaining.

Individual cache items are really a three-tuple *<message id, list of next hops, time-stamp>*.

The message id, which is copied from the geographic message packet header, is used as the key into the hash table. The message id is being used because it can be stored in a relatively small space of fixed width. As such, comparisons between message ids can be done quickly and in a straight-forward manner. However, it would be better (though more complicated and computationally expensive) to use the actual destination polygon itself as the key into the cache because this would allow the cache to be used to full advantage. How to do this, though, is not obvious and future experiments will deal with this matter.

The list of next hops is a list of those child or parent nodes to which this router should forward a copy of the packet to.

Finally, the time-stamp indicates the age of the cache item. The time-stamp is updated every time that the cache item is used. The cache is periodically checked for stale cache items. Stale cache items are those items whose time-stamp is older than some threshold. For the purposes of these experiments, the maximum age of a cache item is thirty seconds and the cache is checked every fifteen seconds for stale items.

5.2.2 Calculating a Router's Service Area

A router's service area is, theoretically, the union of the service areas of its child nodes. However, such a union may produce a polygon which has internal "holes" and which may have sections which are concave. Many algorithms which perform polygon intersection, however, require simple polygons which are convex. Therefore, instead of calculating the union of the child node areas, the *convex hull* was calculated instead. In order to calculate the convex hull, all of the polygonal and circular descriptions of the child node areas were first converted into a list of points. For circles, geometry was used to find ten equidistant points on the circle's perimeter.

5.2.3 List of Child and Parent Nodes

The Routing Table has separate lists of child nodes and parent nodes. The main reason for this division is the different treatment accorded to each. Both lists are maintained as unsorted doubly-linked lists.

Each element on the list of child nodes contains all of the information about the child node and this router's connection to it. Such information includes the child node's service area polygon, a bounding rectangle around that polygon, the operational status of the node, and how to reach that node so that geographic message packets can be forwarded to it.

In comparison, the list of parent nodes is much simpler. Each element of the parent list only needs to know the status of the corresponding node and how to get information to it.

5.3 How Intersection is Performed

When we want to discover if a router services the destination area for a geographic message, we have to perform a polygon intersection test between the message's destination polygon and the router's service area polygon. If the test is positive, the router would then need to perform the same test with each child node's service area.

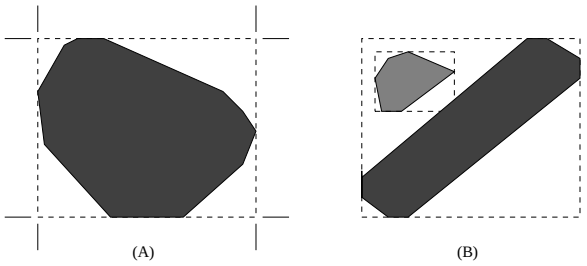

Figure 4: Example of a bounding rectangle around a polygon in (A). Comparing two bounding rectangles for intersection, however, does not guarantee that the underlying polygons intersect (B).

However, given a large number of child nodes, calculating polygon intersection for each child node would be computationally burdensome and may degrade the router's efficiency. Therefore, in order to reduce the computational overhead involved, the following step was taken: a bounding rectangle was created for each polygon. See Figure 4-A. The intuition behind this is that two rectangles can be tested for intersection in a constant number of steps while the same test for a polygon depends on its number of vertices.

The intersection test now works in the following manner. First, the bounding rectangles are compared to see if they intersect each other. If this test fails, then the underlying polygons also do not intersect each other. A successful test, however, does not guarantee that the underlying polygons intersect because of corner cases such as in Figure 4-B. If the test does succeed, however, then the underlying polygons are tested for intersection. The actual intersection test used depends on whether the polygons are actually points, circles, or generic polygons. See Table 1.

6 Experimental Results

The purpose of these experiments is to evaluate the efficacy of the geographic routing system. In order to do this, a prototype system was created using C++ which runs on either Linux 2.0 (Slackware 3.1) or Sun's Solaris 2.5. When running the experiments, the hardware and software setup shown in Figure 5 was used.

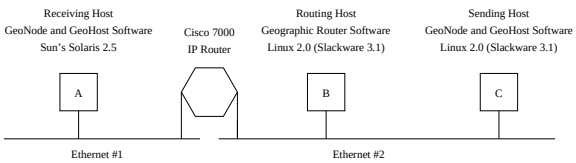

Figure 5: Hardware and Software Setup used in the Geographic Routing Experiments

Three separate host computers (A, B, and C) and two different subnets were used. Executing on host A were Ethernet #1's GeoNode and host A's GeoHost software. The GeoNode for Ethernet #1 believes that its service area is a circular region centered at the coordinates (-60.00 degrees longitude, 60.00 degrees latitude) with a radius of one degree. Also executing on A was the geographic message reception program. Correspondingly, the GeoNode for the subnet, Ethernet #2, and the GeoHost software for host C both executed on host C. The GeoNode for Ethernet #2 believes that its service area is a circular region centered at the coordinates (-70.00 degrees longitude, 60.00 degrees latitude) with a radius of one degree. The program that sent the geographic messages also resided on host C. The geographic router itself was run on host B. After starting up, the geographic router connected itself to the two GeoNodes by using IP tunnels and inserted information about them and their service areas into its routing table.

Shape Intersection Methods Used		
First Shape Type	Second Shape Type	Method Used
Polygon	Polygon	Polygon-Polygon Intersection
Polygon	Circle	First the circle is changed into a generic polygon. The default number of points is ten but the user can specify a different number in the configuration file. Then Polygon-Polygon Intersection is performed.
Polygon	Point	Point in Polygon Detection.
Circle	Circle	Circle-Circle Intersection.
Circle	Point	Point in Circle Detection.
Point	Point	Equality of the (x,y) values.

Table 1: Listing of the Various Methods used to Intersect the Differing Types of Shapes used in Geographic Routing.

All tests were performed late at night when no other users were present on the computers being used. Computers B and C were both Pentium Pro equipped PCs running Linux 2.0 (Slackware 3.1), while computer A was a Sun SparcStation 10 running Solaris 2.5.

In order to report the time to route a geographic message packet, the geographic router software was modified so that it would report the average amount of time to route one hundred packets. A hundred copies of the same packet would be sent through the geographic router. The router would record the time of the arrival of the first packet and the departure of the hundredth packet. The difference between the two measurements was then divided by a hundred and the result was recorded as the amount of time it took to geographically route a packet. Each packet was sent half a second after the previous packet. Unless otherwise stated, this method was used to gather every data point in the coming experiments.

6.1 Comparison of the Routing Time for Differing Destination Shape Types

The first experiment attempts to discover the difference in routing times for a point, a circle, or a polygon. A hundred separate messages of each type were sent to the area centered at (-60 longitude, 60 latitude) with the circle having a radius of .5 degrees and the polygon having three vertices equidistant from each other and each being .5 degrees from the center. The routing times for each shape type were averaged together. See Table 2.

Time to Route Differing Shapes	
Shape Type	Route Time (usec)
Polygon	4426
Circle	3527
Point	389

Table 2: Comparison of the Time in Microseconds to Geographically Route the Three Basic Polygon Shapes.

The table shows that routing to a circle destination is nine times more expensive than routing to a point. Furthermore, routing to a polygon is 25% more expensive than routing to a circle. In comparison, Bradner [1] finds that an IP router will route an IP packet in 1 microsecond (assuming that the address is already in the cache). It should be noted that an IP router uses firmware to route a packet which would account for some of the time difference.

6.2 Test of the Effect of the Cache on the Routing Time for Large Messages

After the first experiment, we have an idea of how long it takes to route a single geographic packet. However, how much would a cache affect those routing times - especially for large multi-packet messages? This experiment attempts to answer that question.

For this experiment, three messages (one of each shape type) were sent that consisted of a hundred packets each. The geographic destination of the messages was the same as in the previous experi-

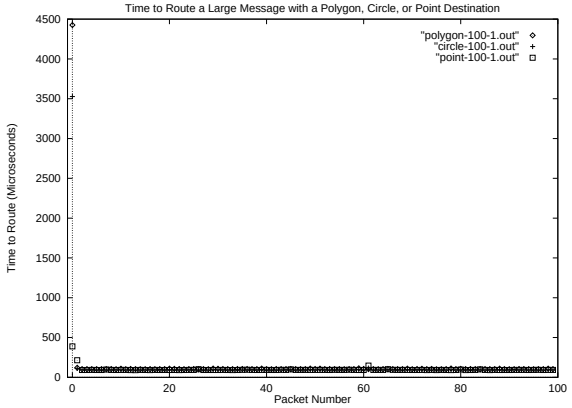

Figure 6: Comparison of the Amount of Time to Route Multiple Packets Going to the Same Polygonal, Circular, or Point Destination.

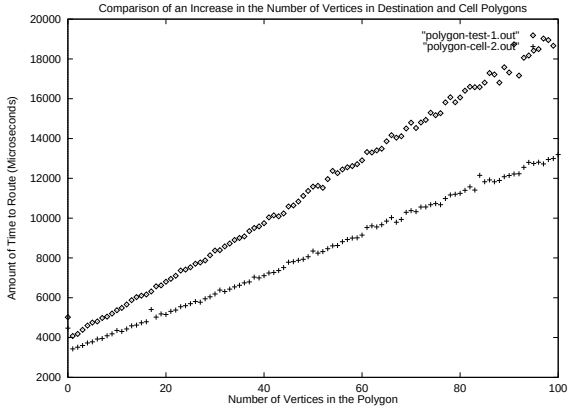

Figure 7: Comparison of the Amount of Time Needed to Route a Single Packet while Increasing the Number of Vertices in the Target or GeoNode Polygons.

ment. The route time results are shown in Figure 6. As before, the first packet of each message suffered a large routing time. However, after the first packet, all other packets were routed using the information in the cache. This caused the routing time for all subsequent packets to drop to around 93 microseconds. This is just 2.1% of the amount of time it takes to route a polygon, 2.6% of a circle's routing time, and 24% of a point's routing time.

This experiment shows that, given a prevalence of larger multi-packet messages, the geographic router would operate more efficiently. This would particularly be the case in a connection-oriented communication model rather than a connection-less model. During the initial connection phase, a small packet could be sent ahead to incur the heavy initial first-packet routing times. All subsequent packets would enjoy the benefits of the router cache. If the packet traffic for a particular connection is bursty, small "maintenance" packets could be sent periodically in order to ensure that the cache items do not time out.

6.3 Comparison of the Effect on Routing Time of Changing the Number of Vertices in the Destination and the Routing Table Node Polygons

During the first experiment, the message with a polygonal destination used a polygon with just three vertices. Now we will discover what is the affect of using polygons with increasing numbers of vertices. Also, when increasing the number of vertices, we attempted to determine if there is a difference in effect between changing the geographic message's destination polygons or changing the routing table entry polygons which represent the GeoNode service areas.

We first measured the effect of increasing the number of vertices of the message's destination polygon. For this experiment, the service areas of the GeoNodes were represented as polygons with three vertices. The geographic destination of the messages was the same as in the previous experiment. The size of the destination polygon started with three vertices and steadily increased by one until the last message which had 102 vertices. The results are shown in Figure 7 as the data points called "polygon-test-1.out."

The results show that as the number of vertices in the destination polygon increases, the time cost of routing that message also increases linearly. The rate of increase is 250 microseconds per vertex.

Next, we measured the effect of increasing the number of vertices of the GeoNode's service area polygon. For this experiment, the destination polygons of all of the geographic messages were represented by polygons with three vertices. The geographic destination of the messages was the same as in the previous experiment. The size of the GeoNode's service area polygon started with three vertices. After each geographic message was received and routed, the service area polygons of the GeoNodes were increased in size by one vertex. The results are shown in Figure 7 as the data points called "polygon-cell-2.out."

The results show that as the number of vertices in the service area polygons of the GeoNodes increases,

the time cost of routing a message also increases linearly. The rate of increase is 100 microseconds per vertex. Curiously, this rate of increase is only 40% of the above. Note also that all of the data points for "polygon-cell-2.out" are less than the data points for "polygon-test-1.out."

It seems counter-intuitive that the two lines should be so different. In order to determine the cause of this difference, the router was profiled to see where the bottleneck was occurring. We found that the main contributor to the difference between the two lines was the cost of translating the destination polygon from the form stored in the geographic message header to a form that could be used for the intersection routine. Finding a way to make this process more efficient would greatly reduce the difference between the two lines.

This experiment shows that it would be more efficient to use polygons with only a few vertices. Perhaps, before using the polygons, the number of vertices could be "reduced" by approximating the polygons with other polygons which have fewer vertices. For example, large time savings could be realized by approximating a 100-vertex polygon (with a routing time of 19,000 microseconds) with a 20-vertex polygon (with a routing time of 6,500 microseconds). However, care must be taken to ensure that any approximation contain the original polygon or packets will not be forwarded to a next-hop node which should have received it. Because of the loss of specificity, packets may be forwarded to nodes that shouldn't have received it in the first place. Future experiments will need to be undertaken to determine whether the increase in routing efficiency outweighs the possible impact of such approximations on the overall internetwork.

6.4 Test of the Effect of Increasing Routing Table Size on the Time to Route

The purpose of this experiment is to determine what effect does increasing routing table size have on the time to route a geographic packet. Furthermore, we want to compare the different effects that a large routing table has on packets bound for small geographic regions, which would intersect with a small number of routing table entries, and packets bound for large regions. which would intersect a greater number of routing table entries.

In order to make the comparison, two separate tests are performed. Initially there are only two GeoNode entries in the routing table with GeoNode

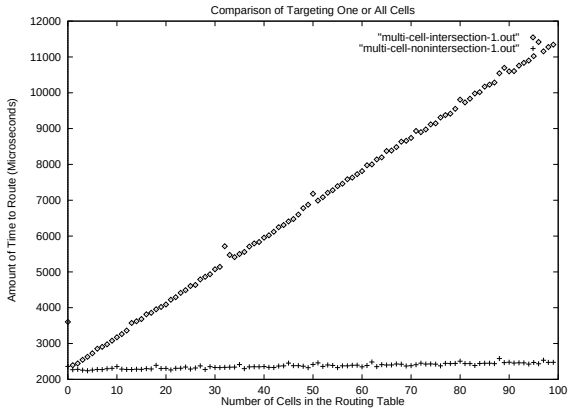

Figure 8: Comparison of the Amount of Time to Route a Single Packet to Either All Cells or Just One Cell in the Routing Table.

#1 being centered at <-60 longitude, 60 latitude> and GeoNode #2 being centered at <-70 longitude, 60 latitude>. The service area polygon for all GeoNodes is a circle with a radius of one degree. All messages will be sent from GeoNode #2's area toward GeoNode #1's area. Each message will have a circular destination polygon with a center at <-60 longitude, 60 latitude> and a radius of .5 degrees. After each message is routed, a new GeoNode entry is added to the the routing table.

Note that the manner in which the routing table is being set-up is artificial. For reasons which are explained below, each new routing table entry's service area polygon is constructed as a copy of either GeoNode #1's polygon or GeoNode #2's polygon. A real geographic routing table would have routing table entry polygons which are either disjoint or only slightly overlapping. The routing table is being set up in this manner for two reasons. First, this configuration allows us to manipulate only one variable at a time - that is, the number of routing table entries. Second, as far as routing times are concerned, the end result for the experiment is the same; the exact placement of the polygons is immaterial as long as they intersect (or do not intersect - depending on the test being done) the destination polygon.

For the first test, every time a new GeoNode entry is included in the routing table, its service area polygon is made to be exactly the same as GeoNode #1. In this manner, each new message will intersect with one more routing table entry than the previous message. The resulting routing times are shown in Figure 8 as the data points named "multi-cell-intersection-1.out." Predictably, as the number of routing table entries that the message polygon

intersects increases, the routing time also increases at a linear rate of 94 microseconds per additional routing table entry.

For the second test, every time a new GeoNode entry is included in the routing table, its service area polygon is made to be exactly the same as GeoNode #2. In this manner, even though there are more entries in the routing table, each new message will intersect with only one routing table entry. The resulting routing times are shown in Figure 8 as the data points named "multi-cell-nonintersection-1.out." Somewhat surprisingly, despite the fact that there are greater numbers of routing table entries, the routing time stays fairly stable. This is where the benefit of first using the bounding rectangles when testing for intersection comes into play. As a result, each new routing table entry requires only an additional two microseconds of routing time.

This experiment demonstrates that the cost of routing geographic messages with polygons that intersect a large number of routing table entries greatly outweighs the cost of routing messages with small polygons which intersect only a small number of routing table entries. A hierarchically configured network would alleviate this problem by ensuring that the individual routing tables will contain a minimal number of entries.

For a flat network model, however, there is another possible solution. Up until now we have assumed that every entry in the routing table represented another GeoNode or router. However, because of physical limitations, each router only has a small number of incoming/outgoing lines. As such, each router is only directly connected to a small subset of the other routers or GeoNodes in its routing table. Perhaps, then, the router should have only one routing table entry per incoming/outgoing line. The polygon for that line's entry would be the union of all of the other routers or GeoNodes that would be reached by using that line. The router/GeoNode on the other end of that line would be considered the next-hop to the whole region defined by the union.

7 Future Work - Geo-Multicasting

While geocasting is an important service, it is more likely that we will multicast rather than broadcast into the geographical areas. For example, we will be interested in reaching all *motorists* on a specific highway, or all *police cars*, rather than reaching *everybody*. This will be accomplished by geographically directed multicast. The geocasting method described before can be modified to accommodate Leo-multicasting. The hierarchy of GeoRouters can be used also to maintain information about multicast group memberships. Other solutions are also possible, based on a concept of *area codes* analogous to the ones used in telephony today. Geo-Multicasting will be described in more details in a forthcoming paper.

8 Conclusions

The geographic messaging project introduces location as a "first class citizen" both in message addressing and routing. In this paper we have attempted to address the efficacy of routing geographically. To this end, under a DARPA-sponsored Integrated Technology Demonstration (ITD) which is being led by Rutgers DataMan Lab within the GLOMO (Global Mobile) program, we constructed the prototype geographic routing system described here and evaluated it through various experiments. The results from these experiments will be used as a guide to improve the efficiency of geographic routing. Future experiments will evaluate geographic routing over a wide-area internetwork. Immediate plans call for the deployment of an experimental network capable of routing geographic messages with nodes at Rutgers DataMan Lab, CECOM in Fort Monmouth (U.S. Army), University of California at Santa Cruz, and Bolt, Beranek, and Neumann (BBN).

While the time to geographically route a packet can potentially be large, the experiments show that there are several techniques for reducing the overall routing cost. First of all, the router's cache significantly reduces the routing time for all but the first packet. This first packet suffers a large routing time. However, after the first packet, all other packets are routed using the information in the cache. This causes the routing time for all subsequent packets to drop to around 93 microseconds. This is just 2.1% of the amount of time it takes to route a polygon, 2.6% of a circle's routing time, and 24% of a point's routing time. Therefore, given a prevalence of larger multi-packet messages, the geographic router would operate more efficiently.

Secondly, using bounding rectangles around all polygons allows us to easily ignore routing table polygons which do not intersect with a message's destination polygon. The cost of routing messages with polygons that intersect a large number of routing table entries greatly outweighs the cost of routing messages with small polygons which intersect

only a small number of routing table entries. As the number of routing table entries that the message polygon intersects increases, the routing time also increases at a linear rate of 94 microseconds per additional routing table entry. On the other hand, for messages with polygons that intersect only a few routing table entries, each non-intersecting routing table entry requires only an additional two microseconds of routing time.

It should be noted that, since the current implementation is completely in the application layer, pushing the geographic routing functionality to the kernel (thereby eliminating the user/kernel boundary crossings) or to firmware would also greatly reduce the routing times. For instance, the amount of time necessary for a Cisco 7000 IP router to reference its cache using firmware is approximately one microsecond [1]. Comparatively, the geographic router needs 93 microseconds to reference its cache even though the cache is implemented as a hash table.

One curious experimental result is the difference in routing cost of an extra polygon vertex in a destination polygon versus a routing table entry polygon. The cost of the former is 250 microseconds per vertex. However, the cost of the latter is just 100 microseconds per vertex which is only 40% of the former. The main contributor to the difference between the two was the cost of translating the destination polygon from the form stored in the geographic message header to a form that could be used for the intersection calculations.

As the users of tomorrow become ever more mobile, the importance of making geographic location an integral part of message routing and resource discovery will increase. As such, the routers and switches of the internetwork will need to increase their intelligence and understanding of the geographic world around them.

References

[1] Scott O. Bradner, *Lab Test / Ethernet Bridges And Routers*, ftp:// hsdndev.harvard.edu/ pub/ ndtl, Harvard University Network Device Test Lab, March 1995.

[2] Deering, S., and R. Hinden, *Internet Protocol, Version 6, Specification*, RFC 1883, Xerox PARC, Ipsilon Networks, December 1995.

[3] Deering, S., and Hinden, R., Editors, *IP Version 6 Addressing Architecture*, RFC 1884, Ipsilon Networks, Xerox PARC, December 1995.

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

[5] T. Imielinski and J. Navas, *GPS-Based Addressing and Routing*, RFC 2009, Computer Science, Rutgers University, March 1996.

[6] J. O'Rourke, C.B. Chien, T. Olson, and D. Naddor, *A new linear algorithm for intersecting convex polygons*, Computer Graphics and Image Processing 19, 384-391, 1982.

[7] Rekhter, Y., and T. Li, *An Architecture for IPv6 Unicast Address Allocation*, RFC 1887, Cisco Systems, December 1995.

[8] Mike Spreitzer and Marvin Theimer, *Providing location information in a ubiquitous computing environment*, 14th ACM Symposium on Operating System Principles, Vol 27, No 5, Dec 1993; also in "Mobile Computing", Ed by T.Imielinski and H.Korth, Kluwer 1996. pp 397 -423

[9] —, *A Technical Report to the Secretary of Transportation on a National Approach to Augmented GPS Services*, http:// www.navcen.uscg.mil/ gps/ reports/ reports.htm