Proceedings of the 2004 Winter Simulation Conference
R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

SIMULATION ANALYSIS OF VIRTUAL GEOGRAPHIC ROUTING

David M. Nicol

Department of Electrical and Computer Engineering
University of Illinois, Urbana-Champaign
1308 West Main Street
Urbana, IL 61801, U.S.A.

Michael E. Goldsby
Michael M. Johnson

Sandia National Laboratories
P. O. Box 969
Livermore, CA 94551-9201, U.S.A.

ABSTRACT

Homeland defense applications will use large-scale ad-hoc networks of small devices. Routing is a crucial problem, for naive means do not scale well. Geographic Routing (GR) (Karp 2000; Giordano, Stojmenovic, and Blazevic 2003) offers hope for scalability, under the assumption that every device knows its geographic coordinates, e.g., through GPS. This solution is unsuitable though when there is no easy means of establishing a device's physical location. indoors. To address this limitation we propose *Virtual Geographic Routing* where we construct a virtual coordinate space and use GR within it. This paper describes VGR, compares the characteristics of paths VGR identifies with those that GR identifies, then presents theoretical and empirical evidence for its scalability.

1 INTRODUCTION

A wireless communication network can be established without external supporting infrastructure, when devices serve as routers as well. The ability to quickly assemble such networks, in a loosely organized fashion, gives rise to the designation "ad-hoc" networks. We are particularly interested in applications of ad-hoc networks where the devices are small, limited in capability, are immobile, require little power, but the number of devices is large. Any given device is able to communicate directly with only a small subset of the entire network. Examples of such architectures include sensor networks, e.g., those that might be deployed in an airport or train station to detect the presence of chemical or biological agents. A message sent from device d_1, destined for device d_2, will typically pass through a sequence of devices. d_1 chooses a device from among those devices it can directly reach, and sends the message to it. The recipient recognizes that it is not the message's target, and likewise forwards the message to another device. The process continues until the message is delivered. We are interested in the problem of determining how routes through the network are determined.

One routing solution is to execute a distributed all-pairs shortest path algorithm on the network (Lynch 1996). As a result of this computation, each device d_1 in the network would have a table that specifies, for *every* other device d_2 in the network, the neighbor to which a message addressed to d_2 should be forwarded. There are a number of problems with this approach. The storage required for a forwarding table is proportional to the number of devices; not only will this demand too much memory for large networks, it reserves valuable space in anticipation of many routes that may never be used. Some routing protocols (e.g., AODV (Perkins 2003) and DSR (Johnson, Maltz, and Hu 2003)) take a demand-driven approach, and compute the route to a specific target only when a message is created for that target. These protocols are designed with the vagaries of mobility in mind, where one must assume that the network is in constant flux. To find a route extant *now*, the source sends out "flooding" queries to all its immediate neighbors. Any recipient that is not the target repeats the query to its own immediate neighbors, provided that the query's time-to-live hop count remains positive after being decremented. When the target device is found, it reports success back to the neighbor that queried it, which reports success back to the neighbor that queried it, and so on, tracing back a route to the target, which the message then follows. If a flooding search with a time-to-live hop-count of n fails to turn up a route, one may initiate another flooding round with a larger count (e.g., $2n$) to look deeper into the topology; this is called an *expanding ring search*. The bulk of communication involved here is not necessary if the network topology is static; topological stability allows for pre-computation whose results can be used to optimize routing for all subsequent messages.

Geographic Routing (GR) (Karp 2000; Giordano, Stojmenovic, and Blazevic 2003) takes advantage of the stability of a given device's geographic location in the network. Assuming that geographic coordinates are bound to a device name and that the binding can be recovered through interaction with a location service (e.g., Li et al. 2000; Li et al. 2001), GR routing has a device route a message to a

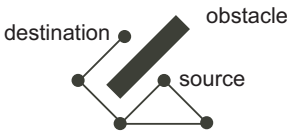

Figure 1: Void in Geographic Routing : A Message Source is Geographically Closer to the Destination than any of its Communication Neighbors

neighbor which is closer to the destination than itself, and which, among all such neighbors, is closest to the destination. Depending on the topology it may not be possible to route a message using this rule. Figure 1 illustrates a *void*, where the message source is geographically closer to the destination than any of its communication neighbors, because of a radio obstacle. Different techniques may be used in such cases, depending on the topology. A flooding search will always find a path, if there is one, and a found route can be cached for reuse.

The main limitation of GR we address is establishment of the geographic coordinates. Either the network devices discover their own locations using GPS, or they are loaded with their locations by some other means, sure to be time consuming if not automated. We set out asking if one could use device connectivity to create a virtual coordinate system to replace geographic coordinates. The main hoped-for advantage would be to achieve the performance of GR (in terms of lengths of routes discovered), scalability (in terms of slowly growing memory demands as network size increases), but without the complication of determining actual geographic locations for every device. This paper demonstrates a solution to these problems.

2 VIRTUAL GEOGRAPHIC ROUTING

Virtual geographic routing replaces the physical location of a device with a virtual location developed solely using information about the connectivity of the devices. We explore a metric based on a device's distance—in numbers of hops—from a set of distinguished devices called *anchors*. The intuition behind VGR is illustrated in Figure 2. Three distinguished points exist in the domain. A fourth point is characterized by having Euclidean distance a from one distinguished point, b from another, and c from the third—in the coordinate space induced by anchors it has coordinates (a, b, c). Circles around the anchors illustrate the set of points at a given distance from the anchor; (a, b, c) is the unique point that is distance a from the first anchor, distance b from the second anchor, and distance c from the third. A device that is close to this point in the plane necessarily is close to it in the anchor coordinate space.

For ad-hoc routing we can approximate distance using connectivity information to compute minimum distance

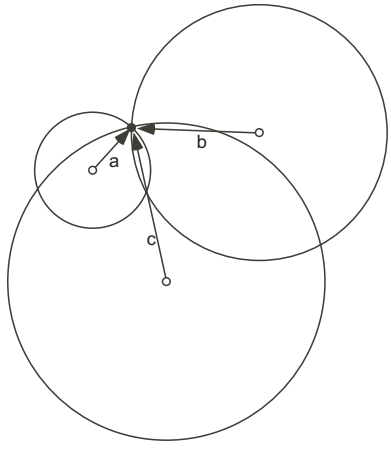

Figure 2: Labeling a Point in Terms of Distances from Three Anchor Points

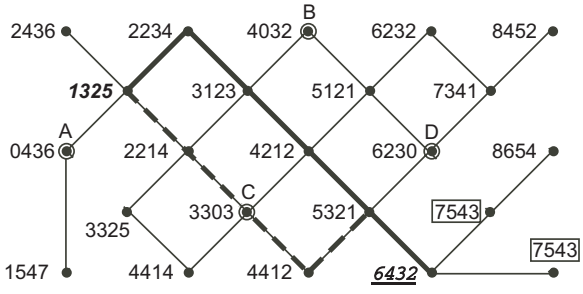

Figure 3: Virtual Geographic Routing : Devices Labeled with Distances to Anchors ABCD, Dotted Line Reveals GR Path from x to y, Solid Line Reveals VGR path, Label 7543 is Aliased

(measured in hops) from devices to anchors. Figure 3 illustrates the concept. It shows a graph whose edges illustrate the direct point-to-point communication sustainable by the radio environment. There are four anchors, illustrated with a circle around the graph node, and labeled A,B,C, and D. Each device is labeled with a four digit code whose first digit is the shortest number of hops to A, the second digit is the shortest number of hops to B, then to C and D respectively. The regularity of this graph induces two devices to receive the same label (7543); we call this *aliasing*. Like voids, aliases are a special case we must accommodate. The graph also highlights both the GR path from 1325 to 6432 (heavy dotted line) and the VGR path (heavy line), when the Euclidean distance metric is used on the virtual coordinate space, i.e. the distance between two points $\vec{P}$ and $\vec{Q}$ in virtual coordinate space is $||\vec{P} - \vec{Q}||^{1/2}$. It interesting to observe that while the paths are different, they both enjoy the same number of hops, and each step in the VGR path reduces the geographic distance to the destination.

A device's virtual coordinates give its location in a K-dimensional virtual space, where K is the number of anchors. The number of hops to each anchor can be developed by a simple distributed algorithm, similar to the distributed Bellman-Ford algorithm (Lynch 1996) but using messages that include only the hops counts to the anchor rather than to all devices.

Virtual geographic routing is stalled by local minima of the distance metric, just as real geographic routing is. As we've seen in the example, aliasing is also a problem. Since aliased devices can be several hops apart, aliasing causes a problem for the routing algorithm.

Location Service. We expect that VGR coordinates will be hidden from applications. For some applications, such as geographically based data storage and retrieval (Ratnasamy et al. 2002; Shenker et al. 2003), it is not necessary to associate locations with particular devices. Often, however, the devices are distinguished by unique device identifiers, and a message must be routed to the device with a particular identifier. In that case, for either real or virtual geographic routing, it is necessary to have a means of discovering a device's location given its identifier.

A service that supplies the location of a device given its device identifier is called a location service. A location service that is particularly suited to systems using geographic routing was presented in Li et al. (2000). The service is completely distributed, letting every device play the role of location server for a number of other devices. The service is itself based on geographic principles and works without any device knowing the identity of any of its location servers. It is presented in connection with real geographic routing and based on a recursive subdivision of the unit square, but the analysis and the algorithms carry over to K-dimensional virtual coordinate space. It requires an average amount of storage at each device that is proportional to the total number of devices with a very small constant of proportionality. The service has a communication pattern such that the expected path length approaches a constant as the number of devices increases (Li 2001), so it meets our scalability criterion.

Voids. A *void* occurs when a message cannot be routed to a neighbor that is closer to the destination. After considerable experimentation, we settled on an expanding ring search for *any* device closer to the destination. Here escape from a void is be found by flooding a search to find a device that is closer to the destination in the virtual geographic metric. An expanding ring search broadcasts a search request with a maximum search radius attached. The request is broadcast by all devices (to their limited set of communication partners) that receive it if they cannot satisfy the request and the search radius is not exhausted. (Such a radius is often called "time to live" or TTL.) Positive results are sent back to the device that originated the request and can be consolidated along the way to avoid congestion of the links near the requesting device. If no positive results are found at the current radius, the search is repeated with an increased radius. In a static, connected network, such a search will always eventually be successful.

The path traveled by the request is accumulated along the way and included in the request. The response is returned along the reverse path, and at each of the devices on the path, the next hop from the originating device toward the satisfying device is cached in a table of "escape hops". Such escape hops are associated with the satisfying device in the table.

When a void is encountered in the course of routing a message, the escape table is searched to see if it contains any entry associated with a device that is closer to the destination. If it does, the next hop in that entry is used, and the closer device becomes an intermediate routing destination. Before it is forwarded to the next hop, the message is marked to show that it is no longer in geographic mode, and the identity of the intermediate destination and the virtual geographic coordinates of the device at which the message left geographic mode are included in it.

A device receiving such a message first checks to see if it is itself closer to the destination than the device at which the message left geographic mode. If it is, it restores the message to geographic mode and continues the routing. If it is not, it looks in its escape table to see if it has an entry associated with the intermediate destination given in the message. If it does, it forwards the message to the next hop in that entry. Note that in a static network, it will always find such an entry. If it does not, it acts as if were the device that first encountered the void.

Aliases. An alias is recognized when the message reaches a device that has the same virtual geographic coordinates as the destination but is not the destination (i.e., it has a different unique identifier). An expanding ring search is used here, too. The search succeeds when the search request message reaches the device with the destination's device identifier. In this case, the response to the search returns the entire path from the originating device to the destination, and the originating device caches the path upon receiving the response, associating it with the destination's identifier. As an optimization, the destination caches the reverse path, associating it with the originating device.

When a device recognizes an alias in the course of routing a message, the alias cache is consulted. If the path to the destination is already present in the cache, no search need be done. Before the message is forwarded to the first hop in the path, the message is marked to show that it is in alias mode and the remainder of the path is placed in the message. At each step on the path, the next hop is removed from the message and the message is forwarded to it. That is, source routing is used between the two aliased devices.

Anchors. Placement of anchor nodes may be a priori, random or algorithmic. Placement is important because an uneven distribution of anchors among the other devices

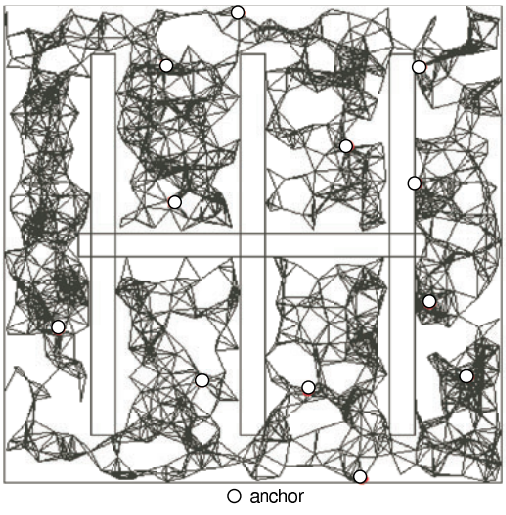

Figure 4: Sample Topology with Radio Obstacles, 1024 Devices, Average Degree 10, 12 Anchors

tends to produce a greater number of local minima and aliases. In the examples in this paper, the anchors were placed a priori (except for those in Figure 4, which were placed algorithmically.) For more on anchor placement, see Goldsby et al. (2003).

3 EXPERIMENTS

Next we use simulation to consider how VGR behaves relative to GR. We are interested in the length of routings these protocols find, the amount of memory they require, and any evidence we may find that VGR is scalable. We provide many graphs and tables in a tech report (Goldsby et al. 2003). We select some illustrative examples taken from topologies with approximately 1000 devices, where devices are placed randomly, subject to domain obstacles; for example, see Figure 4. We then look at resource requirements of networks of size up to 10000 devices, and consider how those requirements scale as the network size grows.

Figure 5 illustrates device-to-device path lengths, taken from 10 instances of networks, approximately 1000 devices each, as a function of the average number of neighbors. The upper graph plots this data for networks without barriers, the lower graph summarizes graphs with barriers. We plot maxima and averages for the cases of actual shortest path (SP), VGR, and GR.

In the case of no barriers, there are significant differences among SP, VGR, and GR with respect to the maximum path length. SP is far and away the shortest, with VGR yielding a 25% increase over GR. However, the differences on *average* path lengths are minute—almost indistinguishable. In the case of barriers, the relative difference between maximum VGR and GR path lengths is not so large, due to the increase in path length both methods endure owing to the barriers. The maximum length shortest path is again significantly smaller than either GR or VGR. However, once again we see that differences in average path length are comparatively much smaller.

It is also informative to consider how GR and VGR behave with respect to handling anomalies, voids (for both) and aliases (only for VGR). Table 1 summarizes some of these differences, for 10 randomly sampled networks of 1000 devices, with 10 neighbors on average, with and without barriers. As we consider how messages traverse the network, we can classify hops taken as being “geographic"—meaning in accordance to geographic routing rules—or “special", meaning that a special condition (like a void, or alias) is being handled. The first row of the table below gives the fraction of geographic hops for GR and VGR, where we see that in the case of no barriers, for both GR and VGR the majority of hops are geographic. The second row measures the fraction of times a message crosses a hop as part of a ring search (to fix an anomaly), and we see that in the case of no barriers, the fraction is minuscule for both GR and VGR. The third row describes the maximum and average depth into the network that an expanding ring search had to penetrate to resolve an anomaly, with GR and VGR behaving similarly. The last row measures the maximum and average number of hops between aliased devices in VGR.

Some differences between VGR and GR emerge when we consider networks with barriers. GR is much more susceptible to voids than VGR, as evidenced by the fact that only 66% of hops taken are geographic under GR, as opposed to 90% for VGR. Fully 77% of the hops traversed in exploring all paths worked out to be search hops, as compared with less than 5% for VGR. Likewise, both the maximum and average depth into the network GR has to penetrate to deal with voids are significantly larger than that for VGR. For VGR it is interesting to note that the maximum and average distances between aliased devices is substantially larger than when barriers are not present. These two tables show that for random networks, in the absence of barriers the overheads of anomalies are small, and equivalent for GR and VGR. However, when barriers are introduced, the overheads become more significant, in fact, for GR they become overwhelming. In such cases VGR is better able to deal with the voids that barriers induce.

We come to similar conclusions when we examine how many node coordinates (either GR or VGR) are stored in the escape tables used to resolve voids. Figure 6 gives data on networks of size 512, 1024, and 2048 devices; all networks are constructed to have an average neighbor count of 9. The upper graph plots average and maximum counts for GR and VGR on networks without barriers, the lower graph plots

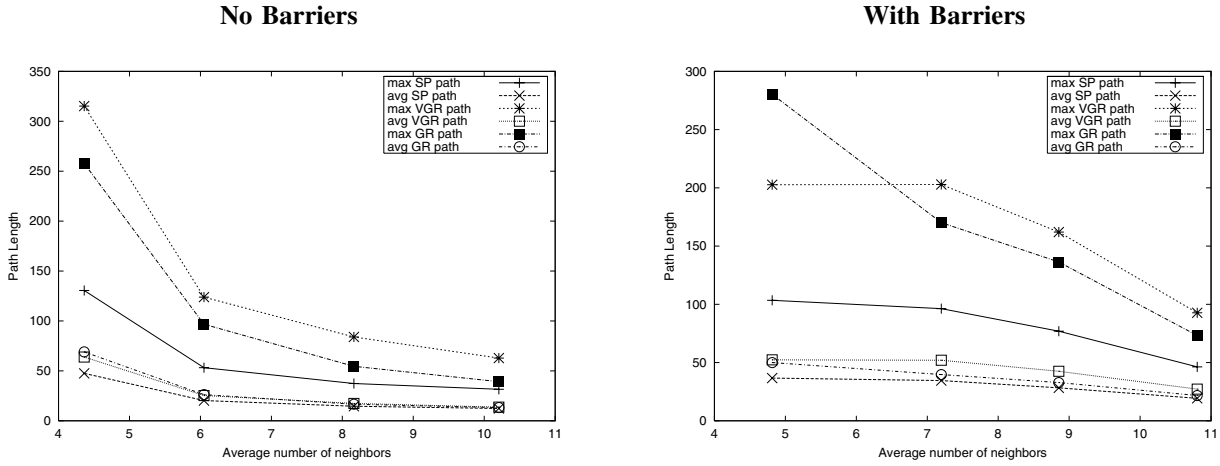

Figure 5: Path Length for Shortest Path (SP), VGR, and GR on Random 1000 Device Betworks, Without and With Barriers

Table 1: Path Anomaly Overheads for Networks With, and Without Barriers

	No Barriers			With Barriers		
	GR	VGR	GR/VGR	GR	VGR	GR/VGR
fraction geo. hops	0.95	0.93	1.02	0.66	0.90	0.73
ratio search hops / total hops	0.0034	0.0030	1.13	0.778	0.046	17
max/average flood radius	8.4/2.66	8.5/2.28	0.99/1.16	40.5/4.7	14.3/2.76	2.83/1.70
max/average hops to alias	—	2.3/0.99	—	—	16.6/3.56	—

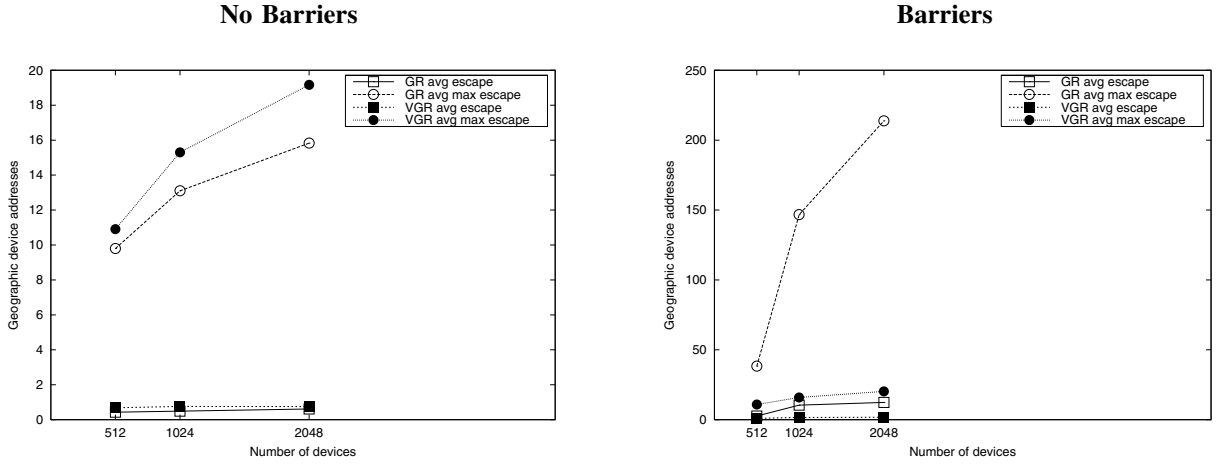

Figure 6: Escape Table Size (in Geographic Identities) for GR and VGR, on Randomly Sampled Networks With and Without barriers

the same for networks with barriers. The averages shown are taken from randomly sampled graphs (10 for network sizes 512 and 1024, 6 for network size 2048). The most significant difference between routing methods occurs for the maximum escape table size on networks with barriers. Here GR method shows again how it has more voids to deal with, and emphasizes the memory cost of doing so (at least if solutions to voids are cached as they are discovered, and not just recomputed when needed).

It should be noted that GR and VGR have potentially significant per-address memory demands. An escape table entry for GR holds the target node's unique identifier (1 word), and some geographic location information, say x and y (2 words). A VGR entry has that same unique identifier, and another word holding the hop count for every anchor. The simulations we report here all use 12 anchors; the memory cost of holding anchor hop counts depends on how we pack hops into words. A straightforward implementation of 1 word per hop count gives rise to 13 words per VGR identity. However, the most important point to be taken from the data shown is that the overhead from escape tables is not large, and therefore suitable for small, cheap, low-capacity sensors.

3.1 Simulation and Scalability

As we have seen, the memory requirements of VGR relate to storing coordinates of neighbors, storing escape tables for voids, and storing escape tables for aliases. Voids and aliases are handled on demand, which implies that the memory resource demands of VGR depend very much on the traffic patterns. If we are to take the approach we used in analyzing 1000 device devices—gather statistics from all paths—we will quickly run into computational issues, for the number of paths grows in the square of the number of devices.

In order to gather some confidence that VGR is scalable (in the sense that its resource demands do not grow quickly with the network size) we will have to consider something other than growth rates on models small enough to exhaustively study. Likewise, if one is faced with the prospect of deploying a network with 10's or 100's of thousands of devices, we need a means of estimating device memory sizes sufficient to hold escape and alias tables, once computed. (Of course, memory requirements can be traded off for computation/communication requirements by the simple expedient of having a device overwrite cached escape tables, and recompute them if and when needed.)

We report now on analysis that strongly suggests that VGR will scale as needed, and gives a practical means of creating upper bounds on memory demands on networks that are too large to explore all paths.

The basis for our analysis is a stochastic ordering relation, called *stochastic variability*, defined in Ross (1996) as follows. Let X be a non-negative random variable with cumulative distribution function F_X (and $\bar{F}_X(s) = 1.0 - F_X(s)$), likewise let Y be a non-negative random variable with distribution function F_Y. We say that Y is stochastically more variable than X, written $X \leq_v Y$, if for all $t \geq 0$,

$$\int_t^\infty \bar{F}_X(s)\, ds \leq \int_t^\infty \bar{F}_Y(s)\, ds.$$

Whenever $X \leq_v Y$, for any increasing convex function f, we have $E[f(X)] \leq E[f(Y)]$. Likewise, if we have a set of $2n$ random variables satisfying $X_i \leq_v Y_i$, $i = 1, \ldots, n$, with all the $\{X_i\}$ being independent and all the $\{Y_i\}$ being independent, and function $f : \mathcal{R}^n \to \mathcal{R}$ being increasing and convex, then $E[f(X_1, \ldots, X_n)] \leq E[f(Y_1, \ldots, Y_n)]$. The function $\max(X_1, \ldots, X_n)$ is increasing and convex.

Our overall approach is to look at the distributions of memory requirements, and maximum path length *from a randomly sampled device*. If we find evidence that these distributions are stochastically less than other distributions, then we can use the expected maximum of n of these bounding random variables as a bound on the expected maximum of n memory storage, or path length, random variables.

Pursuing this approach we examined a large number of histograms of memory requirements (from many randomly sampled networks), and noticed that some of these had heavier tails than one typically sees in, say, a normal distribution. We wrote code that, for a given network, constructed the empirical cumulative distribution function of the per-device requirements. This is the distribution associated with uniform random samples of devices. We then tested whether the exponential distribution with the same mean as the empirical distribution was stochastically more variable than the empirical distribution. In every network we tested, across a large range of sizes and samples, we found the exponential to be stochastically more variable than the empirical distribution.

We did a similar experiment on a path length statistic. Selecting a device at random, we compute the maximum length of a path from that device to any other device. On suitably sized networks we can compute the empirical distribution of this random variable by finding the longest such path, for every device. Histograms of this distribution suggested we consider normal distributions.

The theory of extrema is well developed (e.g., see Leadbetter, Lindgren, and Rootzen 1983; Galambos 1987), and results for exponential and normal distributions are known. In the case of exponentials, the expected maximum of n independent exponentials with mean μ grows in proportion to $\mu \log n$; in the case of the normal distribution the expected maximum grows as $\mu(2 \log n)^{1/2}$. These results give us the confidence we seek in VGR's inherent scalability. Logarith-

mic growth in per-device maximum memory requirements is scalable, growth in maximum path length bounded by the square root of the logarithm of network size is likewise scalable. Furthermore, the bounds provided by the exponential and normal distributions have, experimentally, proven not to be particularly tight.

We illustrate use and utility of these bounds in the graphs shown in Figure 7. For each network size we plot three data points : (i) the sample average of the graph's statistic (space requirements, or longest path from a device) based on random sampling, with the sample set being 5% of the network size, (ii) the maximum value of that statistic, observed by looking at *every* device, and (iii) a bound on the expected maximum, based on the assumption that the statistic's sample mean is the mean of the statistic's actual distribution. The idea is that for large networks we can use random sampling of devices to construct a sample mean, then use the analytic bound as a guide towards resource requirements. We include observed maxima in these graphs to illustrate the scale of the bounds as compared with the real maxima.

The upper graph in Figure 7 looks at space requirements, and shows that the average storage needed per-device is insensitive to the network size. This is not surprising, as we have seen already that networks without barriers do not need escape tables very much, so that the average device memory need is dominated by the cost of storing neighbor's VGR coordinates. On the other hand, increasing the network size definitely increases the global observed maximum. The "dip" in the observed max at 6000 devices is an artifact of random sampling, as only one network is analyzed for each device size, and there is variation in these statistics as one randomly samples networks. It is evident that the bound built using an exponential assumption is conservative, but it is also true that the gap between observed maximum and bound decreases over the interval of observation, raising the question of whether there may be a cross-over at a much larger network size.

The right-hand graph in Figure 7 provides similar data for the "maximum-path-length-from-a-device" metric. In this case we see slight growth in the average, which is to be expected because as the network grows given a fixed average connectivity, the network diameter grows as well. We also see that the observed maximum longest-path is quite close in value to the average, reflecting low variation in that measure. The bound on the maximum is obtained by assuming that the sample mean is the distribution's mean, and applying the formula for mean maximum of independent normals. We again see that this bound is very loose.

4 CONCLUSIONS

A number of emerging applications—including those in Homeland Defense—will rely upon networks of small communicating devices which self-organize into a network. The issue of routing messages within such a network is of active interest. We are particularly interested in techniques which will work on very large networks. One technique which scales, Geographic Routing (GR), routes with a sense of geographic direction, always pushing a message closer to its ultimate goal. Geographic coordinates give one a global frame of reference against which routing decisions can be made with relatively little memory or computational overhead. However, geographic routing requires support through GPS or some other method, costly in equipment, power, and (if device locations are loaded into them) time. This paper considers *virtual* geographic routing (VGR), where the network constructs a virtual coordinate system as a function of connectivity, then routes "geographically" with respect to that coordinate system. We describe the key ideas, use simulation to examine the nature of routes discovered by VGR as compared with GR on networks of size 1000 devices, and then consider the scalability of VGR's behavior and resource demands as network size grows. We find that VGR handles networks with severe communication obstacles better than GR, and otherwise has routes that are not significantly different in nature than GR's. We use the notation of stochastic variability to describe a means of assessing VGR resource demands for large networks, and to argue for its scalability as network sizes grow. VGR thus gives the advantages of GR, without each device needing to know its physical geographic location.

ACKNOWLEDGMENTS

This research was supported in part by DARPA Contract N66001-96-C-8530, NSF Grant CCR-0209144, and Department of Energy contract DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. In addition this project was supported under Award No. 2000-DT-CX-K001 from the Office for Domestic Preparedness, U.S. Department of Homeland Security. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security.

REFERENCES

Galambos, J. The Asymptotic Theory of Extreme Order Statistics. Robert E. Krieger Publishing, Malabar, FL, 1987.

Goldsby M., R.P. Tsang, M.M. Johnson, H.Y. Chen, N.R. Bierbaum, D. Kilman, H.R. Ammerlahn, and D.M. Nicol. Robust Message Routing for Mobile (Wireless) Ad Hoc Networks. Sandia National Laboratories Report SAND2003-8762, 2003.

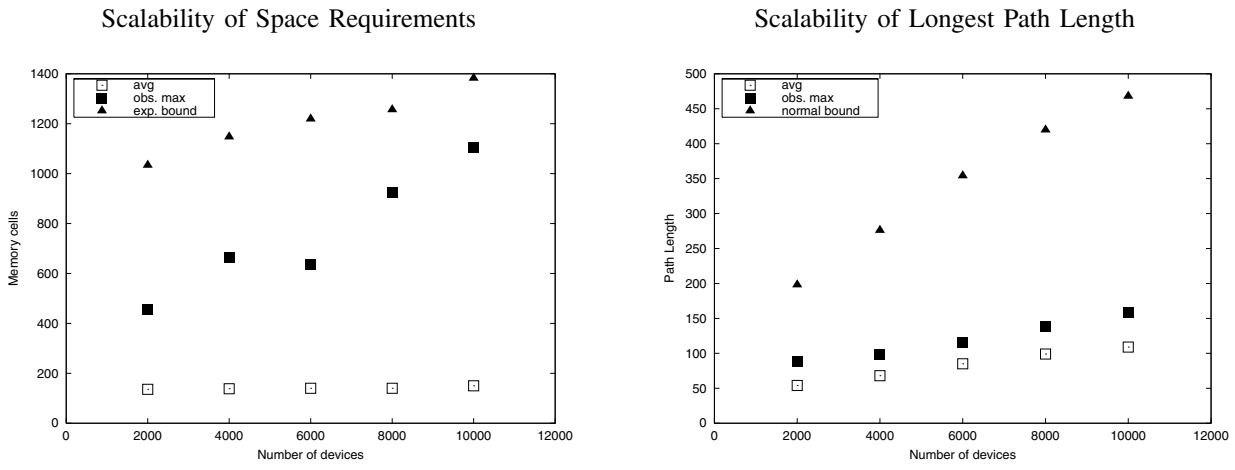

Figure 7: Memory Use and Path Lengths on Randomly Generated Networks with an Average of 9 Neighbors, and No Barriers

Giordano, S., I. Stojmenovic, and L. Blazevic. Position Based Routing Algorithms for Ad Hoc Networks: a Taxonomy. to be published in Ad Hoc Wireless Networking, X. Cheng, X. Huang and D.Z. Du, ed. Kluwer, 2003.

Johnson D., D. Maltz and Y.-C. Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR), 2003. Available online via `http://www.ieft.org/internet-drafts/draft-ietf-manet-dsr-09.txt` [accessed July 1, 2004].

Karp, R. Geographic Routing for Wireless Networks. Ph.D. dissertation, Harvard University, Cambridge, Massachusetts, 2000.

Leadbetter M.R., G. Lindgren, and H. Rootzén. Extremes and Related Properties of Random Sequences and Processes. Springer-Verlag, New York, 1983.

Li, J., J. Janotti, D. S. J. De Couto, D. Karger and R. Morris. A Scalable Location Service for Geographic Ad-hoc Routing. in Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, R. Pickholtz, S. K. Das, R. Caceres and J. J. Garcia-Luna-Aceves, ed., pages 120-130. ACM Press, New York, New York, 2000.

Li J., C. Blake, D. De Couto, H. Lee, and R. Morris, Capacity of Ad Hoc Wireless Networks, in Proceedings of the 7th ACM International Conference on Mobile Computing and Networking (MobiCom R01), pages 61-69, 2001.

Lynch, N. Distributed Algorithms. Morgan Kaufman, San Francisco, California, 1996.

Perkins, C., E. Belding-Royer and S. Das. Ad Hoc On-demand Distance Vector (AODV) Routing. 2003. RFC 3561, Internet Engineering Task Force, July, 2003. Available online via `http://www.ieft.org/rfc/rfc3561.txt` [accessed July 1, 2004].

Ratnasamy, S., B. Karp, Y. Li, Y. Fang, D. Estrin, R. Govindan and S. Shenker. GHT: A Geographic Hash Table for Data-centric Storage, in Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications, pages 78-87. ACM Press, New, York, New York, 2002.

Ross, S. Stochastic Processes, 2nd edition. John Wiley and Sons, 1996.

Shenker, S., S. Ratnasamy, B. Karp, R. Govindan and D. Estrin. Data-centric Storage in Sensor Nets. in ACM SIGCOMM Computer Communication Review, 33(1), pages 137-142, ACM Press, New York, New York, 2003.

AUTHOR BIOGRAPHIES

DAVID M. NICOL is Professor of Electrical and Computer Engineering at the University of Illinois, Urbana-Champaign, and member of the Coordinated Sciences Laboratory. He is co-author of the textbook *Discrete-Event Systems Simulation*, and served as Editor-in-Chief at ACM TOMACS from 1997-2003. He is the General Chair of the 2004 Conference on Principles of Advanced and Distributed Simulation, and the General Chair of the 2006 Winter Simulation Conference. From 1996-2003 he was Professor of Computer Science at Dartmouth College, where he served as department chair, and at the Institute for Security Technology Studies served as Associate Director for Research and Development, and finally as Acting Director. From 1987-1996 he was on the faculty of the Computer Science department at the College of William and Mary; 1985-1987 he was a staff scientist at the Institute for Computer Applications in Science and Engineering. He has a B.A. in mathematics from Carleton College (1979), an M.S. (1983) and Ph.D. (1985) in computer science from the University of Virginia. His research interests are in high performance

computing, performance analysis, simulation and modeling, and network security. He is a Fellow of the IEEE.

MICHAEL E. GOLDSBY augments the technical staff at Sandia National Laboratories, California, and is a freelance software author. His interests include parallel and distributed algorithms, programming methodology and virtual environments. He received his B.Sc. in mathematics from the University of Arizona and is a member of the ACM.

MICHAEL M. JOHNSON is a Distinguished Member of the technical staff at Sandia National Laboratories, California, where he works on homeland security applications to help protect the nation against the use of weapons of mass destruction. He received his M.Sc.degree in Computer Engineering from the University of California, San Diego, California in 1991. His research interests include high performance computing, modeling and simulation, and embedded systems design.