

Geographic Routing

Alex Benn, Andrew Harp

Abstract—Geographic routing is a promising technique for wireless packet routing. In very large ad-hoc networks, or in severely resource-constrained situations such as sensor networks, it is impractical for each node to maintain global routing information. Geographic routing routes by geographical coordinates, vastly reducing the amount of state maintained at each node. We have developed and implemented a simulation testbed for modeling wireless ad-hoc networks. This simulation testbed features an impressive suite of graphical features and data logging mechanisms. After implementing this simulation environment, we have tested BVR under a wide range of mobility conditions in order to gain more insight into its performance characteristics under mobility.

I. INTRODUCTION

Routing is a vital component of any large interconnected network. Making routing decisions on traditional wireline networks is a well-studied problem, and a number of excellent routing protocols have been written and used effectively. However, the designers of these routing protocols have made a number of assumptions that make them inapplicable to wireless routing. First, wired networks group the Internet by breaking it up into subnets, where each subnet is a physical collection of nodes connected to a single router. These subnets are grouped according to contiguous ranges of IP addresses. Wireless ad-hoc networks, on the other hand, may be connected in an arbitrary fashion, where two nodes may be close to each other but have very different addresses. Since a

packet can no longer be sent to a router according to the range of IPs that router is responsible for, packets must be routed entirely according to the specific location of the destination node.

A number of routing protocols have been developed for ad-hoc wireless networks. DSDV [1], DSR [2], and AODV [3] are three examples of existing routing protocols designed for the specialized world of wireless. These routing protocols operate by maintaining a large table of destinations and a best route to each, storing either a picture of the whole network or of those nodes to which packets have been forwarded in the recent past. Since the number of flows and the number of destinations can grow linearly with the size of the network, these approaches require $O(n)$ state on each machine for n nodes in the whole network. Also, since nodes may be able to move, data in the tables can quickly become stale, either requiring costly table maintenance overhead, or potentially resulting in bad route selection.

Geographic routing fights these problems in a clever way: instead of maintaining information on every node about the best route to each destination node, geographic routing routes according to the geographic location of the intermediate and destination nodes [4], [5]. As long as the network has certain properties, a node can look at its list of neighbors and pass the packet to the neighbor closest to the destination. In this way, a node only needs to store geographic information about itself, its immediate neighbors, and its destinations; if the number of neighbors of each individual

node is constant, then increasing the size of the network has no effect on the amount of state each node must maintain, and has very little effect on the . As multihop networks grow to hundreds or even thousands of nodes, this factor will be critical to their usability and success.

II. BEACON VECTOR ROUTING

Geographic routing is an important area of current research. Researchers have developed a number of routing protocols. Some of these protocols require that nodes have some knowledge of their true geographic locations, acquired perhaps through GPS or manual input. However, we consider this requirement to be impractical for large-scale deployment and low per-node effort, both important for many of the environments where geographic routing is appropriate.

Beacon Vector Routing [6], or BVR, obviates the need for absolute knowledge about the geographic location of nodes. BVR makes its routing decisions based on the *beacon vector*, a set of coordinates associated with each node. The beacon vector stores the distance of a given node to each of the k nearest *beacons*. Beacons are self-selected nodes that serve as anchorpoints for the network by broadcasting their presence periodically. In this manner, a transmitter can address a specific node by attaching that destination's beacon vector to each transmitted packet.

When a node makes a routing decision, it knows only the beacon vectors of its neighbors. This restricts the amount of state each node knows about the network. The node finds the neighbor whose beacon vector is most similar to that of the destination, and attempts to route toward that neighbor. If none of the neighbors is an improvement on the best node the packet

has already passed through, then we perform a fallback algorithm where the packet is simply routed directly to the beacon that is closest to the destination. Once it reaches that beacon, a scoped flood takes place and the packet will reach the destination. On the way to that beacon, the packet may pass through a node that has a better neighbor, in which case normal beacon vector routing will resume.

III. EXPERIMENT

Mobility is a vital consideration for many large ad-hoc networks. However, the seminal paper on BVR [6] has little discussion of the effects of mobility on the performance of BVR. In situations where some or many of the nodes may move, techniques such as BVR may still be useful, but performance remains untested. We sought to fill this gap.

A number of our experiments examine the performance of BVR under various degrees of mobility. We also compare BVR to broadcast in order to determine routing stretch, and to give a baseline for lightweight constant-state routing mechanisms. Since mobility will introduce more cases that trigger BVR fallback, we wished to determine how much stretch increases as mobility increases the inaccuracy of the beacon coordinates of nodes.

The question of mobility also leads directly to a discussion of beacon selection. Beacons provide a framework for routing packets to all nearby nodes. For this reason, it is extremely important that beacons not move, as mobility in beacons can have two ill effects. First, a mobile beacon may invalidate existing beacon coordinates of all nearby nodes, and additionally will hamper the fallback mechanism where packets are forwarded to the nearest beacon node. Under BVR, beacons are selected randomly. We wished to determine whether

and to what degree routing performance would benefit from preferring stationary nodes when selecting beacons. In this vein, we have also elected to compare situations where beacons are stationary to those where beacons move as much as non-beacons.

It is worth noting that our experimental setup makes three assumptions. First, we assume that transmissions are occurring in the steady state: beacon information has already been disseminated across the network, and sources know the beacon vector of their destination nodes. Second, nodes have perfect simultaneous knowledge of the transmitting status of all nodes within range, so two nodes that can hear each other will never transmit at the same time. As a result, collisions can only occur because of the hidden-terminal problem. Third, although nodes are not assumed to know anything about their true location, we assume that there is some mechanism to determine whether a node is mobile or not for selecting beacons. There are a number of possible low-cost means for doing this, such as a statistical analysis of the turnover of neighbor nodes, or an accelerometer attached to the node; we did not wish to sidetrack our project with a careful consideration of solutions to this problem and the various shortcomings of each solution.

All experiments were performed on a 100 node network. We limit transmissions to one packet per second from one fixed node to another fixed node, while varying the amount of movement allowed for nodes, and whether beacons are allowed to move. While we only run each simulation for 100 seconds, we run the entire set of simulations 10 times with a different random seed each time so that we can extract a more consistent result from the averages. Seeding the random generation allowed us to re-use the same set of ten net-

works between BVR and broadcast, and across mobility situations.

IV. RESULTS

As we expected, increasing mobility in the network also increases the total number of transmissions for BVR, as it causes BVR to fallback to broadcast routing as topology information becomes increasingly outdated. However, even with high amounts of mobility, BVR continues to outperform basic broadcast.

Broadcast gave us the lowest hop count, which is not surprising, as it is basically a breadth-first-search. Of course, it also is the least efficient in terms of network usage.

We think that the dip in hop count for both sets of BVR experiments may be due to the increased disorder of the network making making our implementation of BVR fallback to broadcast at some point in its journey, after which the packet will reach the destination by the most direct path. However, if the network is too disordered, the probability becomes greater that it will simply start off in the wrong direction.

Oddly, in our experiments, non-moving beacons did nothing to reduce our transmission count, and in fact hurt it slightly. This could potentially be due to some combination of fallback and our wandering behavior, which changes the likelihood distribution of where a wandering node will be found. However, stationary beacons did provide the best overall reception percentage.

V. POTENTIAL IMPROVEMENTS

In performing these experiments, we came up with several ideas for how routing performance could be improved under high mobility situations. We believe that the non-moving beacons performance would be greatly aided

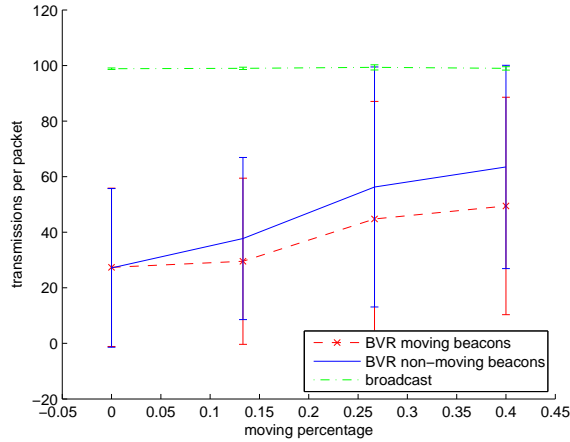


Fig. 1. Relationship of mobility to transmission count. This is the total number of transmissions in the entire network for a single packet to travel from the source to the destination.

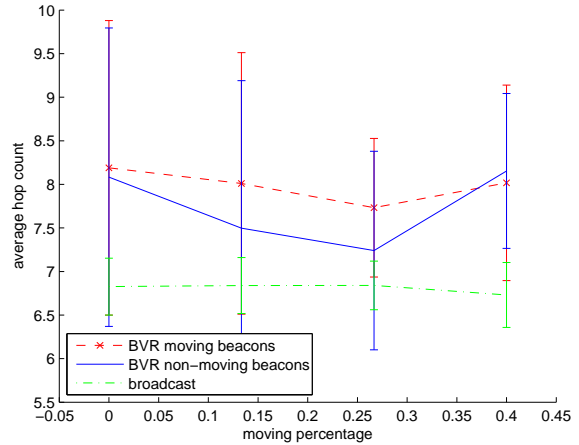


Fig. 3. Relationship of mobility to the average hop count. Notice that BVR dipped under intermediate mobility, then went back up as mobility increased.

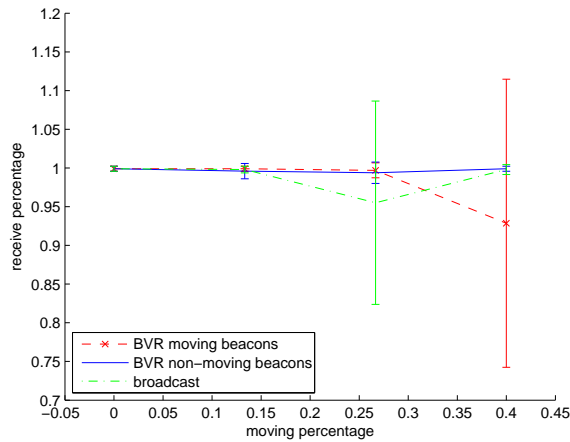


Fig. 2. Relationship of mobility to proportion of packets received

if, as nodes were moving around the simulation environment, they updated their beacon vectors according to the nodes they came into contact with. This could be done in an ad-hoc manner as nodes discover other nodes. A node would simply take the minimum on a per element basis of all Beacon vectors it currently knows

about, and add one to the counts for its personal value. Some additional bookkeeping, however, would be necessary for when nodes go out of range or go silent, so that it can fall back to the second best known value. Were we to implement these features it is hard to imagine stationary Beacons not outperforming moving beacons.

VI. CONCLUSION

Our finding that stationary Beacons have no benefit over randomly-chosen Beacons was surprising. This is an interesting find, though it is concerning that it is such at odds with intuition. However, we show that stationary Beacons significantly improve the success rate over moving beacons when mobility is high, an important benefit. We also show that hop count is improved for intermediate mobility situations. These features show that this topic certainly warrants further investigation. There are other modifications that could be made to BVR to provide mechanisms for packet delivery improvement under mobility that could be

used in applicable large-scale networks, such as citywide vehicular networks. One possible improvement we contemplated is to leave a crumb trail as nodes move relative to one another: if a node recently saw a destination node but recently lost contact with that node, it could use information it knows about its neighbors to forward that packet using an estimation of the new beacon vector of the destination. Unfortunately due to time constraints we did not have enough time to explore this and other ideas.

REFERENCES

- [1] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*. New York, NY, USA: ACM, 1994, pp. 234–244.
- [2] 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.
- [3] C. E. Perkins, "Ad-hoc on-demand distance vector routing," in *In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 90–100.
- [4] D. M. Nicol, M. E. Goldsby, and M. M. Johnson, "Simulation analysis of virtual geographic routing," in *WSC '04: Proceedings of the 36th conference on Winter simulation*. Winter Simulation Conference, 2004, pp. 857–865. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1161890>
- [5] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information." ACM Press, 2003, pp. 96–108.
- [6] R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica, "Beacon vector routing: Scalable point-to-point routing in wireless sensor networks," in *In NSDI*, 2005.

APPENDIX WIRELESS SIMULATOR

A large portion of our project involved the design and implementation of an event-driven

simulator in which to run our experiments. Written from scratch in Python on top of an OpenGL accelerated Java-based rendering library, the simulator was designed with versatility and ease of use in mind. The graphical environment shows the physical and topological layout of the nodes in the simulation, and can optionally show a great deal of information about each element in the virtual world. A user may pan and zoom in this virtual world, pause, and control the speed of simulation arbitrarily. It also features a number of impressive graphical effects.

We elected to use the event framework that we implemented to handle both the simulation itself and the visualization frontend. This means that viewport redraws are placed in the event queue alongside simulation events, and the perceived speed of the simulation is controlled by the simulated-time interval between redraw events. This choice was made to greatly simplify implementation, though it somewhat limited what we could do with the visualization tools. For example, since input is only handled each frame, the interface becomes sluggish if the speed is set to a large value; additionally, frame rate can vary widely depending on the activity of the nodes in the simulation.

Having written our own simulator from scratch gives us the flexibility we need to implement and modify our comparison routing protocols, and to have total control over network layout and mobility behavior. However, development of a brand-new piece of software did not come without its own costs. An overwhelming majority of the time we spent on the project was dedicated to tracking down bugs and simply developing the infrastructure necessary to test the simulation environment itself. We also ran into severe performance limitations that hampered our ability to perform

large-scale simulations.

Figures 4–6 show various graphical features of our simulation environment. Figure 4 shows a node, all the nodes that it can hear and can hear it, and a detailed listing of simulation information at the left. Figure 5 shows a high-level view of the topology of a moderate-sized network. Figure 6 shows the color-coding of the elements of the simulation. For example, broadcast packets are color-coded blue, while unicast packets are yellow, ack packets are green, etc. Additionally, nodes show their current status with color: purple nodes are receiving, red nodes are transmitting, and cyan nodes are hearing a collision due to two or more transmissions within range.

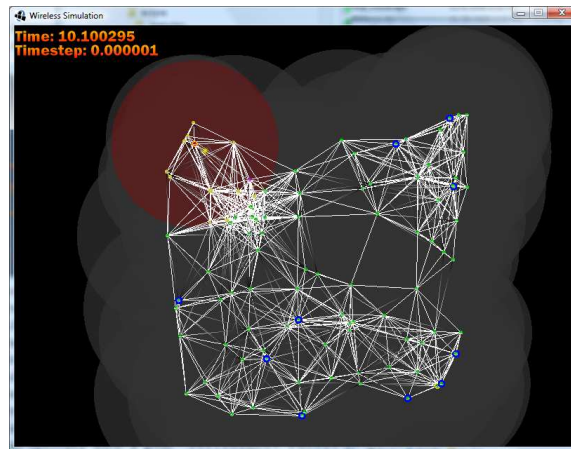


Fig. 5. Overall network topology. The red area represents range for a transmitting node.

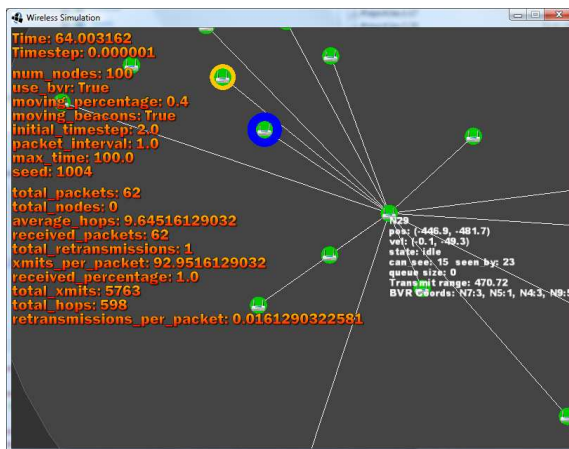


Fig. 4. Zoomed in view of a selected node, with experiment parameters and gathered statistics on the left.



Fig. 6. Broadcasting packets, cyan nodes represent collisions.

```
51.986235: REG N37.attempt_xmit()
           in 0.000009s @ t=51.986244
51.986235: REG <bvr_routing.BVRRouteStruc.announce_coords()
           in 5.561993s @ t=57.548229
51.986244: RUN N37.attempt_xmit()
           - routing pkt: type=network-bc
           - <broadcast_routing.BroadcastRouteStruct
             instance at 3913840>: dropping packet
           - Done routing
           - N37: pkt dropped by route()
51.996280: RUN N60.mutual_add_to_neighbors(N13)
```

Fig. 7. Example log output from event simulator.