# Evaluating the Control Overhead of Routing Protocols in Dynamic Ad Hoc Networks

Victoria Manfredi
Dept. of Computer Science
University of Massachusetts
Amherst, MA, USA
Email: vmanfred@cs.umass.edu

Robert Hancock
Roke Manor Research Ltd
Romsey, Hampshire, U.K.
Email:robert.hancock@roke.co.uk

Jim Kurose
Dept. of Computer Science
University of Massachusetts
Amherst, MA, USA
Email: kurose@cs.umass.edu

## I. Introduction

Wireless and mobile ad-hoc networks (MANETs) are distinguished by time-varying link characteristics and network topology. In such dynamic environments, the network must accommodate these changes, providing end-end packet delivery while at the same time incurring low control overhead. Yet this ideal is difficult to meet in practice: end-end delivery requires some form of end-end (potentially global) coordination, and frequent changes make adaptation to each and every change costly. Consider then a routing protocol that does not adapt globally to every change: can such a protocol perform well (delivering a high fraction of messages to their destination) while at the same time incurring low control overhead?

In this work, we specifically investigate the control overhead incurred by two MANET routing protocols: (i) dynamic source routing (DSR) [2] and (ii) a braid routing algorithm [5] that locally forwards packets within the constraints of a routing subgraph ("braid") constructed around the DSR shortest path. As defined in [5], a $k$-hop braid comprises a "best" (e.g., shortest) path plus all nodes within $k$ hops of the best path, as shown in Figure 1. If a link failure occurs, a braid can potentially adapt to the change by locally routing packets around the failure, with the scope of the forwarding constrained by the braid. Consequently, network-wide flooding of control messages to alert nodes about the link failure and to repair the route need not be (immediately) incurred. *The key idea from [5] is that by using a braid for routing, routes can be re-computed less frequently, and so less control overhead will be incurred.*

In this work, we describe one way to implement braid routing efficiently. Comparing with DSR, we show that braid routing can significantly decrease control overhead while only minimally degrading the number of packets delivered, with gains dependent on node density.

## II. Braid Routing

The braid routing algorithm of [5] performs the following steps every $T$ time-steps: (1) identify the shortest path, (2) build a $k$-hop braid around the shortest path, and (3) perform local forwarding within the braid. We use DSR [2] to construct the shortest path. Similar to AODV backup routing [3], we then leverage overheard route requests and replies to obtain information about nodes adjacent to the shortest path. A node

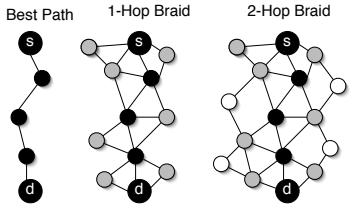


Fig. 1. Example best path, 1-hop braid, and 2-hop braid between a source (s) and destination (d).

using DSR automatically adds all source routes contained in any overheard route requests, route replies, or data packets to its route cache, as well as all sub-paths contained within those routes. Thus, nodes using DSR already have 1-hop braid information in their route caches, and so no additional control overhead is incurred. Some additional control overhead may be necessary, however, to construct a $k$-hop braid for $k > 1$.

To understand how to use a braid for routing, suppose that a node $i$ experiences a link break to a node $j$ while transmitting a packet. In DSR, first, any routes in node $i$'s route cache that use the link from node $i$ to $j$ would be deleted. Then, a route error would be broadcast indicating that the link from node $i$ to $j$ is broken. Finally, node $i$ would attempt to salvage the packet by checking for an alternate path to the destination.

As in DSR, braid routing would also first delete any routes in node $i$'s route cache that use the failed link. Next, unlike in DSR, node $i$ would check its route cache for a braid path to any of the remaining nodes on the packet source route (not just for an alternate path to the destination). If no braid path is available, braid routing defaults to the DSR procedure of broadcasting errors (and re-constructing the path, if there are further packets to send). The frequency with which this occurs determines $T$. If a braid path is available, the packet is sent out over the first hop of the braid path. The packet will still contain the broken source route, but the packet itself will be flagged as a braid packet so that the source route contained in the packet is not added to the route caches of nodes overhearing or receiving the packet. A node $k$ recognizes that it is a "braid node" whenever it receives a packet destined for itself and node $k$ is not included in the packet source route. Whenever node $k$ recognizes that it is being used as a braid node, node $k$ will forward the packet to the appropriate next hop in the

packet source route.

## III. Results

We use QualNet, with $N$ nodes moving in a 2000m$\times$2000m area according to Gauss-Markov mobility [4]. Mobility traces were generated using BonnMotion [1] and fed into QualNet: we set the angle standard deviation to 1 radian, the maximum node speed to 2 m/s, the minimum node speed to 0.5 m/s, the speed standard deviation to 0.2 m/s, and the speed and angle update frequency to 100s; nodes bounce at boundaries. Simulation runs are of 1,000,000s; an additional initial 200,000s was removed for the transient phase. To model network traffic, we use one constant bit rate flow between two nodes; one packet is generated every 0.25s (4,000,000 packets total). The MAC protocol is 802.11b and the connectivity range is 400m.

Figure 2(a) compares the number of link failures for DSR and braid routing, and also shows the number of braid paths attempted by braid routing. As the node density increases, Figure 2(a) shows that the number of link failures increases for both algorithms and that braid routing also attempts more braid paths. Figure 2(a) also shows that braid routing has more link failures: we believe that this is due to braid routing attempting braid paths which eventually fail, in addition to attempting links on the DSR shortest path which also fail.

Next, Figures 2(b) to (d) show the breakdown of the total control overhead incurred for each algorithm according to route errors, requests, and replies. Observe that the route error plots in Figure 2(b) are similar in shape to the link breakage plots in Figure 2(a). This is because link breakages significantly determine the amount of control overhead incurred by an algorithm. Figures 2(a) to (d) then show for $N \leq 40$ that DSR and braid routing incur about the same amount of control overhead. As node density further increases, however, DSR incurs increasingly more control overhead than braid routing, with DSR incurring about $25\%$ more total control overhead for $N = 80$. Figures 2(b) to (d) indicate that braid routing reduces not just route errors, but also route requests and replies.

Next, Figure 2(e) compares the delivery ratios for DSR and braid routing. Figure 2(e) shows that for both algorithms, as the node density increases, the percentage of packets delivered increases. For higher node densities, braid routing delivers slightly fewer packets than DSR: this is due in part to braid routing's use of longer braid paths which also eventually fail.

Finally, Figure 2(f) compares the average packet delay for DSR and braid routing. Figure 2(f) shows that for both algorithms, as node density increases, the average delay decreases. The average delay, however, decreases significantly more for DSR than it does for braid routing: again we believe this is due to braid routing's use of longer braid paths.

## IV. Summary

This paper investigates the control overhead incurred by DSR and braid routing. Comparing with DSR, we show that braid routing can significantly decrease control overhead while only minimally degrading the number of packets delivered, with gains dependent on node density.

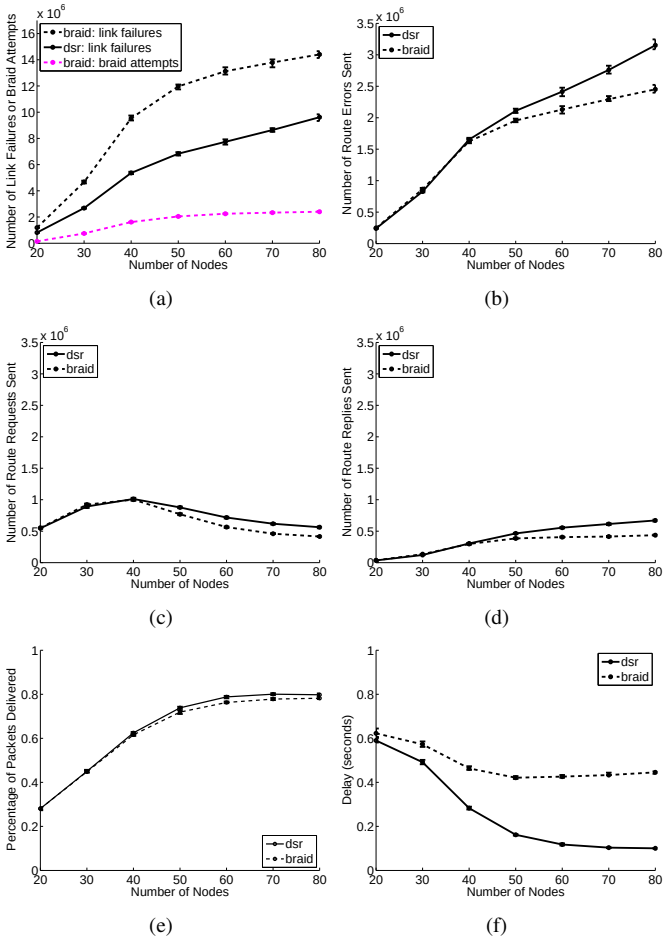


Fig. 2. Performance of braid routing vs DSR. 95% confidence intervals are shown, computed over 10 simulation runs.


## Acknowledgments

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.


## References


[1] Institute of Computer Science IV, University of Bonn. BonnMotion: A mobility scenario generation and analysis tool. *http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/*, 2005.

[2] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 1996.

[3] S.-J. Lee and M. Gerla. AODV-BR: Backup routing in ad hoc networks. In *IEEE Wireless Communications and Networking Conference*, 2000.

[4] B. Liang and Z. Haas. Predictive distance-based mobility management for multidimensional pcs networks. *Transactions on Networking*, 11(5), 2003.

[5] V. Manfredi, R. Hancock, and J. Kurose. Robust routing in dynamic MANETs. In *Annual Conference of the International Technology Alliance*, 2008.