Improving the Performance of Query-Based Routing Protocols Through "Diversity Injection"

Marc R. Pearlman and Zygmunt J. Haas
School of Electrical Engineering
Cornell University
Ithaca, NY 14853
E-mail: {pearlman,haas}@ee.cornell.edu

Abstract-**In this paper, we investigate the diversity properties of routes acquired through traditional route query floods. The on-demand nature of query-based (reactive) routing is appropriate for networks with dynamic network topologies (such as ad-hoc networks). An intrinsic feature of route query protocols is the support for multiple route replies. While it is widely assumed that the set of discovered routes is fairly diverse, an examination of the query propagation behavior reveals that these routes typically have many nodes/links in common. Without sufficient diversity, the routing protocol is limited in its ability to promptly recover from (or even prevent) interrupted service caused by route failures.**

Based on our understanding of query propagation behavior, we propose a novel mechanism that uses information from terminated query threads to "inject" diversity into the collection of route replies. We study the impact of our diversity injection scheme on the demanding ad-hoc networking environment. Our results demonstrate how diversity injection can leverage a small investment of temporary cache to reduce both the rate of route failures and the bandwidth consumed by overhead query traffic.

I. Introduction

An ad-hoc network is designed to support communication between its mobile users, without any dependence on pre-existing infrastructure. The constraints of portable wireless communications place limits on channel bandwidth and transmission power. Such a network requires a routing protocol that can deal with dynamic network topologies, without consuming valuable network resources.

Traditional wired networks use *proactive* routing protocols, like RIP [4] and OSPF [5] to maintain up-to-date routes to all networks nodes. Continuously tracking the frequent topology changes in a practical ad-hoc network can produce an overwhelming amount of control traffic. Even worse, most of the acquired route information is never used, making the proactive control traffic a poor investment of bandwidth. In contrast, *reactive* routing protocols only initiate a global, query-based, route discovery as routes are needed. While some delay is incurred in route acquisition, the amount of overhead traffic is generally much less than proactive routing protocols, because routing information is not wasted. For this reason, reactive protocols are generally viewed as being more suitable than proactive routing protocols for the power / bandwidth limited mobile ad-hoc network (for example [3][6][7]).

An inherent disadvantage of reactive protocols is that they only provide a partial snapshot of network connectivity, and do not automatically adapt to changes in topology. Hybrid reactive/proactive protocols, like the ZRP [2], can improve the stability of *each* reactively acquired route by proactively tracking a node's local topology (allowing some link failures to be bypassed). The robustness of reactive protocols can be further improved by ensuring that each route discovery returns multiple routes that are both stable and diverse.

II. Overview Of Reactive Routing

The reactive route discovery process consists of two phases: the route *query* phase and the route *reply* phase. The route query phase is initiated when a node requires a route to a destination, but does not have the route stored in its route table. This query source issues a route query packet and sends this packet to each of its neighbors[1]. When the query destination[2] receives the query, it may respond with a reply. Otherwise, it forwards the query packet to its neighbors.

In order to prevent route queries from cycling through the network indefinitely, a route query termination criteria is needed. A simple method of query termination is to include a time-to-live (TTL) field in each query packet. Each time that the packet is forwarded by a node, the TTL is decremented. When the TTL reaches 0, the packet is discarded. If the TTL is initialized to the network diameter (in hops) or larger, the query is guaranteed to reach every network node. In fact, if only the route destination responds to the query, a query packet will traverse every possible route between the source and destination (including the minimum cost route). However, this is achieved at great expense, as each

[1] In the case of single shared channel networks, the query packet may be transmitted through a single neighbor broadcast. For multiple channel networks, a separate transmission is required on each outgoing link.

[2] or any node with a valid route to the destination

0-7803-5669-1/$10.00 (c) 1998 IEEE

network node may relay a query many times. Given the limited network resources, a better approach is to regulate the route query so that each node only forwards a route query once (Figure 1). In order to do this, each node must maintain a temporary query cache, which stores the unique query identifier[3] of each recently forwarded route query. This termination criteria still permits robust route discovery, as all reachable links in the network are traversed. However, not all *paths* are traversed, so it is possible that the minimum cost route may not be discovered.

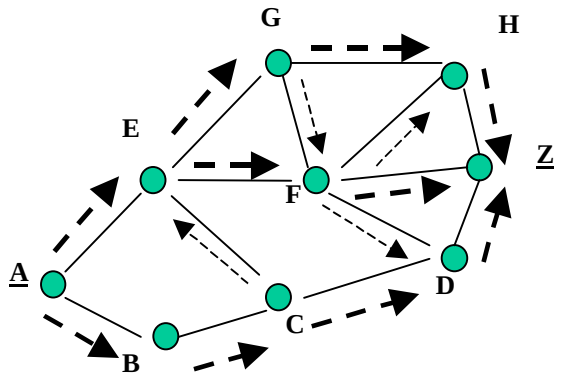

Figure 1: Propagation of a route query issued by A for Z. Bold arrows represent the paths traveled by the successful query packets.

When the route query successfully discovers a route, a reply, containing information about the discovered route, is sent back to the query's source. Therefore, the route query needs to accumulate route information as it spreads through the network. Before forwarding a query packet, a node appends its address to the packet. When a query packet reaches the destination, the sequence of recorded nodes represents a route from the source to the destination. The route may be reversed and used to send the reply back to the query source[4]. Transmission resources can be saved during the route query flood, by distributing accumulated route information among the temporary query caches of each node, instead of appending them to increasingly longer packets. Each node records the hop count and "previous hop" address to the query source. A similar approach can be used during the reply phase. The query source can receive an entire source route to the query destination, or each route node can record the next-hop address to the destination in its routing table.

III. Multiple Routes And Route Diversity

If a destination node is reachable from a query source, a route query will result in at least one route reply. A single reply is sufficient to establish connectivity between source and destination. However, there are benefits to obtaining additional routes. Multiple routes contain extra information about the network topology, which can be used to increase the average connectivity time provided by a route query. This can contribute to a reduction in route discoveries, and therefore less routing control traffic. When multiple routes are used in parallel (i.e. routes alternated on a per-packet or per "stream" basis) further benefits can be achieved. Parallel route usage helps to avoid network congestion by distributing traffic over multiple paths. Parallel route usage also contributes to data security: because packets travel over different paths, eavesdropping becomes more difficult. Finally, coding can be used in conjunction with parallel routing, enabling the destination to automatically recover from some route failures, *without disruption in service*. Of course, the impact of these benefits depends on the diversity of the discovered routes.

Because many paths usually exist between a querying source and destination, it is natural to assume that the route query process will return a diverse set of routes. Contrary to intuition, the multiple discovered route replies tend to share many links in common. Query packets experience variable delays as they are forwarded through the network. For example, short-term variations in the amount of local traffic can cause significant changes in the transmission queue waiting times and channel access delays. Consequently, it is common for one query thread to arrive at destination's "neighborhood" ahead of other query threads. Descendents of this thread propagate throughout the destination region, arriving at the

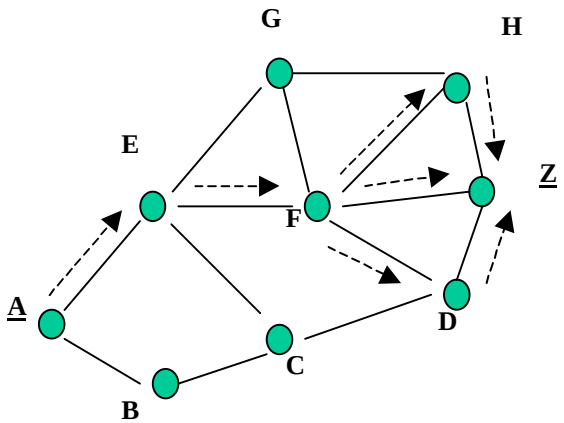

Figure 2: Actual paths traveled by successful route query packets.

[3] The query identifier consists of the query source address and a sequence number used to uniquely identify that all queries from that source

[4] Wireless network links are not necessarily bi-directional, due to differences in node transmission power, receiver sensitivity, co-channel interference levels. For our purposes, we assume that only bi-directional are used, because reliable link-layer communication requires the bi-directinoal exchange of data and acknowledgements.

destination node and spreading outwards to intercept other threads (Recall that a node forwards the first query packet it sees, and discards subsequent copies). As a result, the set of discovered routes are nearly identical, except for some very limited diversity in the last few hops preceding the destination.

The route diversity can also be affected by "topological bottlenecks" in the network. A topological bottleneck exists if two network regions are connected through a single node (the removal of this node would result in network partitioning). If a query source and a query destination are separated by a topological bottleneck, then all successful route queries are descended from the single query packet to cross the bottleneck. Any route diversity that may have been accumulated by other query threads is lost when subsequent query packets are discarded at the bottleneck.

One way of improving route diversity is to allow a node to forward multiple route queries. However, this is not an attractive approach for bandwidth limited ad-hoc networks, as it requires a significant increase in control traffic. A better approach would be to somehow make use of the information accumulated in *all* received query packets, while still relaying the query just once.

IV. Diversity Injection

Although route diversity may exist in the network, traces of this diversity are lost when queries are terminated and their accumulated route information discarded. This lost information can be reclaimed through a technique that we refer to as "*diversity injection*". When a node receives a query packet, it records the accumulated route information in the temporary query cache, regardless of whether the packet is then forwarded or discarded. Each time that the node receives a reply packet, a return path is selected from its temporary query cache, and the reply is forwarded to the source accordingly.

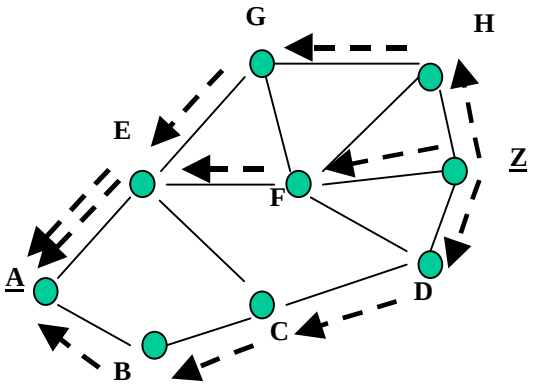

Figure 3: Route replies are injected with the paths of discarded route query packets

The application of diversity injection to in-packet route query accumulation is fairly straightforward. When a node receives a query packet, the entire accumulated route is recorded in the temporary query cache. For each reply that is received, the remaining path back to the query source is replaced with the shortest cached route that has been "injected" the fewest number of times, *and* does not form a loop. Loop detection is trivial, since the entire sequence of nodes from query source to query destination is available. Furthermore, since the received reply contains a loop-free route, there is guaranteed to be at least one cached entry (i.e. the entry that also appears in the reply packet) that can be injected to form a loop free route.

Diversity injection can be applied to distributed route query accumulation in a similar manner. However, the "next hop" route information stored in the temporary route cache is not sufficient to prevent loop formation as the reply proceeds back to the query source. As is the case with route queries, when a loop is detected in a route reply, the reply is discarded. It is conceivable that all route replies might form loops and be discarded, resulting in a route discovery failure.

Fortunately, a simple modification can be made to the diversity injection, to ensure that at least one loop-free route reply is returned to the query source. The node that issues the route reply can specify that the reply explicitly follow the path traveled by the successful query packet. Increasing the number of explicit route replies leads to a tradeoff between route success and route diversity.

V. Performance Evaluation

The diversity injection scheme was evaluated based on an event driven simulation of 50 independent ad-hoc networks. Each network consists of 200 mobile nodes, whose initial positions are chosen from a uniform random distribution over an area of 2000 [m] by 2000 [m]. All nodes move at a constant speed, v, with an initial direction,[5] θ, which is uniformly distributed between 0 and 2π. When a node reaches the edge of the simulation region, it is reflected back into the coverage area, by setting its direction to $-\theta$ (horizontal edges) or $\pi-\theta$ (vertical edges). The magnitude of the velocity is held constant at 20 [m/s].

Data transmission takes place over a single physical channel that is used throughout the network. Nodes contend for access to the data channel using the Dual Busy Tone Multiple Access (DBTMA) protocol [1]. In the absence of a packet collision, we assume that background channel interference and receiver noise effectively limit successful packet and busy tone reception to a physical radius of d_{xmit} = 500 [m]. For the given node distribution, this transmission range results in roughly six neighbors per node.

[5] Direction is measured as an angle relative to the positive x-axis.

Route queries are forwarded to all neighbors by means of a reliable neighbor broadcast service. A node will rebroadcast a packet until an acknowledgement has been received from each neighbor. For simplicity, we assume that an ideal neighbor discovery process provides each node with an up-to-date view of its neighbors.

The reactive routing protocol implemented for this simulation uses distributed cache route accumulation during the route query phase, and in-packet route accumulation during the reply phase. We refer to the maximum number of temporary query cache entries that a node can provide per route discovery as the temporary query cache (TQC) capacity. The performance of diversity injection is measured over a range of TQC capacities ranging from 1 [entry] (corresponding to the traditional reactive routing with no diversity injection) up to ∞ [entries].

The performance of the diversity injection scheme is evaluated based on both static and time-dependent properties of each set of discovered routes. We quantify the diversity of each route set based on the ratio of unique links / total links[6]. In the best case, all routes are distinct, yielding a diversity measure of 1. In the worst case, all routes are identical, and the diversity measure = 1/# of routes. Also of interest is the lifetime of the discovered route set. The route set lifetime represents the amount of time until all routes in the set become invalid. Larger route set lifetimes result in more stable end-to-end connectivity and lead to less routing traffic. The potential benefits of diversity injection are weighed against its cost, name the average amount of memory consumed by the temporary query cache.

VI. Performance Results

The motivation behind the diversity injection scheme is to address an apparent lack of diversity in discovered route sets. Figure 4 underscores this diversity problem. We note that, without diversity injection, the level of route diversity is very close to the lower bound. All discovered routes essentially follow the same path, except for slight variations around the destination. Diversity injection dramatically improves the route diversity, doubling or tripling the amount of information returned from a route discovery.

As explained in section II, route query protocols that forward just one query packet do not necessarily discovery the shortest possible route between two nodes. In fact, the variations in query forwarding delay make the discovery of a minimum distance route an uncommon event. For example, the shortest discovered route

[6] Alternatively, the ratio of unique *nodes*/total *nodes* can be considered. We have found that the two ratios are nearly identical.

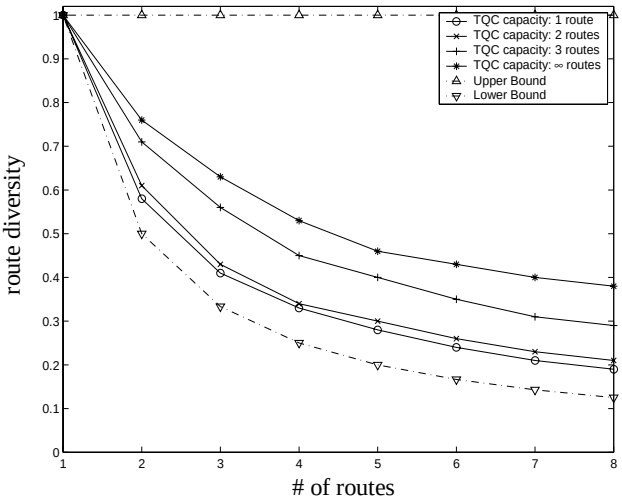

Figure 4: Diversity of Discovered Route Sets

between two nodes separated by 10 hops is an average of 15% longer than optimal. Temporarily caching route information from all received route query packets (rather than just the first) increases the possibility that a reply packet will be able to return to the query source along a shorter path. Figure 5 shows that diversity injection reduces the *excess* length of the shortest discovered route by an average of 60%.

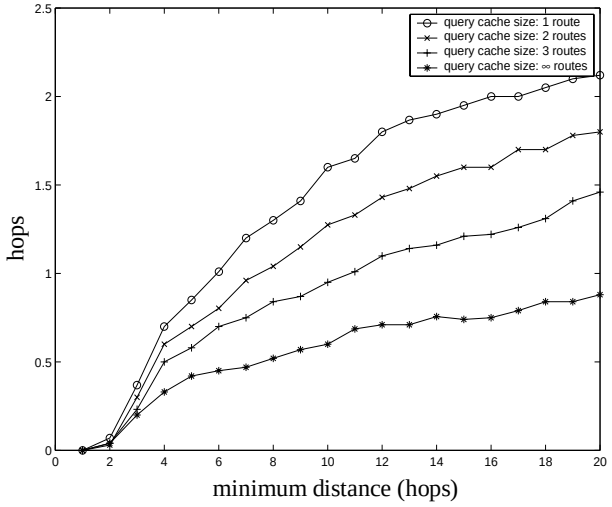

Figure 5: Excess Length of Shortest Discovered Route (compared to minimum distance route)

The combination of increased route set diversity and shorter least-hop routes results in a noticeable improvement in the discovered route set lifetimes. From figure 6, we observe that the diversity injection scheme can increase the discovered route set lifetime by a factor of two or three. Even when only one extra query path is cached in the TQC, the route set lifetime is extended by

about 50%. The increased route set lifetime is perceived by the network user as fewer session disruptions and improved throughput (as less bandwidth is consumed by the routing protocol for route maintenance/rediscovery)

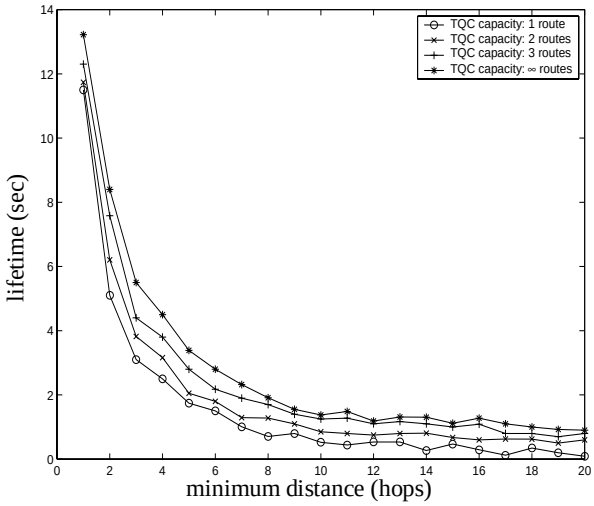

Figure 6: Average lifetime of discovered route set

The cost of implementing diversity injection is the memory required for the temporary query cache (TQC) storage. As the previous figures have confirmed, the benefits of diversity injection are only partially realized when the TQC capacity is limited to a few entries. Much better performance is achieved when no limits are placed on the TQC capacity. Fortunately, infinite TQC capacity does not place a large burden on the nodes. From Table 1, we see that an infinite capacity TQC only holds an average of five entries per query, for this network. More generally, for each route discovery, the TQC will hold, at most, one entry sent from each of its neighbors.

Table 1: Storage Requirement of Temporary Query Cache (TQC)

TQC Capacity	Average TQC Size
1	0.91
2	1.70
3	2.30
∞	5.02

The diversity injection scheme does *not require* a node to store extra entries in its TQC. This is important, because not all network devices may have sufficient memory resources to meet the network's route discovery demands, especially when these resources and demands vary greatly over time.

VII. Conclusion

Because reactive routing protocols return multiple routes in response to a single route query, it is often assumed that this set of routes is diverse. In fact, the discovered routes are almost always identical, except for some minor path variations near the route destination. Without sufficient diversity, the routing protocol is limited in its ability to promptly recover from or even prevent interrupted service caused by route failures.

The proposed diversity injection scheme addresses the route diversity problem by temporarily caching path information from *all* received route query packets. These paths can be "injected" into the returning route reply packets. This allows the replies to return to the source along different paths, even though the successful queries may have traveled along a common path.

Simulation results confirm that diversity injection significantly increases the diversity of discovered route sets. Furthermore, the availability of extra cached route information provides more opportunities for constructing shorter routes. The combination of these factors contributes to a two to three times increase in the lifetime of the discovered routes. The combination of improved route set diversity and lifetime translates to fewer service interruptions and more available bandwidth.

The benefits of diversity injection rely on an investment in short term memory. Although many machines can easily accommodate the full storage load of diversity injection, each node has the option of limiting its participation in the diversity injection process (or not participating altogether), without affecting the proper operation of the routing protocol.

References

[1] Deng, J. and Haas, Z.J., "Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks," IEEE ICUPC'98, Florence, Italy, Oct. 5-9, 1998.

[2] Pearlman, M.R. and Haas, Z.J., "Determining the Optimal Configuration of the Zone Routing Protocol," to appear in *IEEE JSAC*, vol. 17, num. 6, June 1999.

[3] Johnson, D.B., and Maltz, D.A. , "Dynamic Source Routing in Ad-Hoc Wireless Networking," in *Mobile Computing*, T. Imielinski and H. Korth, editors, Kluwer Academic Publishing, 1996.

[4] Malkin, G., "RIP version 2," IETF RFC 2453, Nov. 1998.

[5] Moy, J., "OSPF version 2," IETF RFC 2328, April 1998.

[6] Park, V.D., and Corson, M.S. "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *IEEE INFOCOM '97*, Kobe, Japan, 1997.

[7] Perkins, C.E. and Royer, E.M., "Ad Hoc On-Demand Distance Vector Routing," *IEEE WMCSA'99*, New Orleans, LA, Feb. 1999.