PERFORMANCE ANALYSIS AND ENHANCEMENT OF DYNAMIC SOURCE ROUTING FOR MOBILE AD HOC NETWORKS

APPROVED BY SUPERVISING COMMITTEE:

Dr. Rajendra V. Boppana, Thesis Advisor

Dr. Hugh B. Maynard, Thesis Reader

Dr. Kleanthis Psarris, Department Chair

PERFORMANCE ANALYSIS AND ENHANCEMENT OF DYNAMIC SOURCE ROUTING FOR MOBILE AD HOC NETWORKS

by

Anket Mathur

Presented to the Faculty of
The University of Texas at San Antonio
in partial fulfillment
of the requirements for

HONORS THESIS IN COMPUTERS SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Division of Computer Science
October 2005

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Rajendra V. Boppana, for his time, encouragement and insights. His guidance has been very valuable for this thesis and for myself. I would like to thank Carlos Cardenas, a graduate student in the CS department, whose help throughout the project was invaluable. I would also like to thank fellow undergraduate student Andrew Rodriguez, who had some interesting insights on the topic. I am grateful to Dr. Hugh B. Maynard for serving on my committee.

Abstract

Dynamic Source Routing (DSR) is one of the most commonly used routing protocols for mobile ad hoc networks. Several features of DSR tend to hurt the performance, especially in high mobility and low traffic situations. This issue has been studied by various researchers, and DSR is shown to perform better with certain features turned off. In this thesis, I show that DSR's performance is unsatisfactory even when these features are turned off.

I suggest three simple and intuitive changes to the routing protocol. Using simulations, I show that these changes provide significant performance improvements for different network densities and traffic loads. To illustrate the relative significance of various factors, the three proposed techniques, traffic load and network density, I present 2^k factorial analyses of the simulation data. Based on the statistical analysis, I show that the three proposed techniques not only improve performance, but have the largest impact on performance among the various factors.

Contents

1. Introduction

A mobile ad-hoc network (MANET) is a multi-hop wireless network formed by a group of mobile nodes that have wireless capabilities and are in proximity of each other. MANETs can facilitate communication among mobile users in situations where no fixed wired infrastructure is available, because it may not be either economically feasible or physically possible to provide the necessary infrastructure, or because the expediency of the situation does not permit its installation. For instance, a group of soldiers may need to interact while on a combat mission in unknown territory, friends or business associates may run into each other in an airport terminal and wish to share files, or a group of emergency rescue workers may need to be quickly deployed after a calamity.

If all the hosts that want to communicate are within transmission range of one another, no routing protocol or routing decisions would be necessary. In many cases, however, two hosts that want to communicate may not be within the transmission range of each other, but can communicate if other hosts between them are willing to forward packets for them. The routing problem in mobile ad-hoc networks gets complicated due to the inherent non-uniform propagation characteristics of wireless transmissions and due to the possibility that the nodes may change their location at any time.

Several routing protocols have been proposed for mobile ad-hoc networks. One such routing protocol is the *Dynamic Source Routing* (DSR) [1,4]. In this thesis, I will analyze the performance of DSR using simulations. Though DSR is designed to be adaptive and suitable for mobile networks, it has been shown to perform poorly at low traffic loads and high node mobility situations [12]. To remedy this, several researchers have proposed to turn off certain features of DSR that tend to create stale routes and cause poor performance [5,8,9]. In this thesis, I investigate the performance impact of these changes. I also introduce and evaluate three additional modifications to improve performance. Finally, I present a factorial analysis in order to determine the relative impact of different techniques on the performance of DSR.

The rest of the thesis is organized as follows. Chapter 2 describes the DSR protocol. Chapter 3 provides an analysis of the original DSR and corrections suggested by other researchers. Chapter 4 describes three proposed techniques to improve performance and chapter 5 presents the results from simulations. In chapter 6, I perform a factorial analysis of different factors affecting the performance of DSR. I draw my conclusions in chapter 7, where I also mention my plans for future work on related topics.

2. Dynamic Source Routing (DSR)

The Dynamic Source Routing (DSR) protocol is a simple and robust routing protocol designed specifically for use in multi-hop wireless ad-hoc networks of mobile nodes.

The key features of DSR are:

- Source Routing – The sender of a data packet knows the complete hop-by-hop route to the destination. These routes are stored in a *route cache.* Data packets sent by the source node carry the complete route in the packet header. Intermediate nodes forward the packet based on the route in its header. In most cases, the only modification that an intermediate node may make to the header of a packet is to the *hop count* field. The fact that all data packets are routed from the source has widely perceived security benefits [3,4]. This may make DSR a strong contender as a routing protocol for many applications.

- On-Demand – DSR attempts to reduce routing overhead by only maintaining routes between nodes taking part in data communication. The source discovers routes on-demand by initiating a route discovery process only when it needs to send a data packet to a given destination. As a result, a significant amount of routing overhead is eliminated.

2.1. Basic Operation

To send a packet to another host, the sender constructs a *source route* in the packet's header, giving the address of each node in the path to its destination. The sender then transmits the packet over its wireless network interface to the first hop identified in the source route. When a node that is not the final destination receives such a data packet, it simply transmits the packet to the next hop indicated in the source route. Once the packet reaches its final destination, it is delivered to the transport layer on that host.

Each node in the network maintains a *route cache* in which it caches the routes it has learned. When a source node needs to send data to another node, it first checks its route cache for a route to the destination. If a route is found, the sender uses this route to transmit the packet. If no route is found, the sender buffers the packet and obtains a new route using the route discovery process indicated below.

2.1.1. Route Discovery and Maintenance

A node that needs to send a packet to another node in the network but does not have a route to the destination in its cache initiates the route discovery procedure. The source broadcasts a *route request* packet to all nodes within wireless transmission range of it. In addition to the addresses of the source and the destination nodes, a route request packet contains a *route record,* which is an accumulated record of the sequence of hops taken by the route request packet as it is propagated through the ad hoc network during this route discovery. When a node receives a route request packet, it does the following:

a) If it has seen the same request before (determined by the source address and the request sequence number), the request is ignored.

b) If the destination address of the request matches its own address, then the route record in the packet contains the route by which the request reached this node from the source. A copy of this route is sent back to the source in a *route reply* packet by following the same route in reverse order.

c) Otherwise, if the node has a route to the destination in its cache table, it creates a route reply packet with the route from its cache, and sends it back to the source using the path in the route request packet header, in reverse order. Such replies are called *intermediate-node replies*.

d) Otherwise, it appends its own address to the route record, increments the hop count by one, and rebroadcasts the request.

When the source receives a route reply, it checks its send buffer and transmits any packets that had been waiting for this route. It then adds this route to its cache table for future use.

If any link on a source route is broken (detected by the data link layer of the node attempting to send the packet to its next hop), a *route error* packet is generated. The route error is unicast back to the source using the part of the route traversed so far, erasing all entries in the route caches along the way that contain the broken link. A new route discovery must be initiated by the source, if this route is still needed and no alternate route is available in its cache.

2.1.2. Optimizations

Several optimizations to this basic protocol have been proposed and have been evaluated to be very effective by the authors of the protocol [1]. Some of them are:

a) Data Salvaging – If an intermediate node encounters a broken link and has an alternate route to the destination in its cache, it can try to salvage the packet by sending it via the route from its cache.

b) Gratuitous Replies – When a node overhears a packet not addressed to itself, it checks to see if the packet could be routed via itself to gain a shorter route. If so, the node sends a *gratuitous reply* to the source of the route with this new, better route.

c) Route Snooping – A node that overhears a data packet and does not have the route indicated in the packet's header in its own cache, adds the route to its cache for future use.

3. Critique of DSR

By virtue of source routing, nodes have access to a large amount of routing information. For instance, using a single request-reply cycle, the source can learn routes to each intermediate node on the route in addition to the intended destination. Each intermediate node can also learn routes to every other node on the route. Snooping of data packets transmitted in the vicinity also gives access to a significant amount of routing information. In particular, a node that overhears a data packet can learn routes to every node in the path indicated in the packet’s header.

DSR makes use of route caching aggressively. A destination replies to every route request that it receives. Thus a source can learn, and store many alternate routes to a destination. A forwarding node also caches any source route in a packet it forwards for possible future use.

Without an effective mechanism to remove stale entries, route caches may contain routes that are invalid. Then, route replies from intermediate nodes may carry stale routes. Data transmissions using stale routes result in wastage of channel bandwidth. Other caches are also polluted when packets with stale routes are snooped. Several previous studies describe the stale route problem and its harmful effects [5,6,7].

3.1. Analysis of route ages in DSR

Route staleness has been identified as a major problem affecting the performance of DSR. The major cause of the stale route problem is that there is no concept of the *freshness* or *staleness* of routes. For instance, even after a stale cache entry is removed from a route error, a subsequent forwarded or snooped data packet carrying the same stale route can put that entry right back in. This possibility increases at high data rates, as there will be a large number of data packets carrying the stale route.

In order to determine the freshness of a route, we introduce a mechanism of time stamping the routes with a *request time* and a *reply time*. In our modification, each request packet includes the 'request time', which is the time when the request was sent by the source node. The destination node includes the request time, as indicated in the request packet, and the time of the reply (denoted as ‘reply time’) in the reply packets. When the reply packet reaches the source, it copies both the request time and the reply time to the cache table entry that will contain the route. The route request and reply times are stored with each route and are indicated in the route replies by the intermediate nodes. The route age information that these fields provide can be very useful when choosing the best route for a given destination and reducing the average age of the routes used.

3.2. DSR ‘features’ that hurt performance

Certain features of DSR have previously been shown to hurt its performance [5,8,9,14].

- **Intermediate-Node Replies –** When a route request packet is heard by a node that is not the destination itself, but has a route to the destination in its cache, a route reply packet is sent back to inform the source of the route. Intermediate-Node replies make the route learning process faster because all route requests do not need to travel all the way to the destination if one of the intermediate nodes already has the desired route. They also reduce the number of RREQs transmitted.

  However, the drawback with Intermediate-Node replies is that the route in the intermediate node could be arbitrarily old when the route request is heard. Hence, the source has no guarantee of the validity of a route even immediately after the route reply is received. In a highly mobile network, intermediate-node replies are very likely to be invalid. When a source receives a bad route reply, it tries to send the waiting data packet along the route. Upon failure of one of the links along the route, a route error packet is propagated back to the source. The source then issues a new route request, starting the process all over again. So, an invalid route reply from an intermediate node can be very costly in terms of time and bandwidth.

- **Data Salvage -** When a node in the path of a route experiences a link failure, it checks its own route cache table to see if it can *salvage* the data packet by sending it via an alternate route from its route cache. Data Salvage can be useful in relatively static networks because the time spent in learning a new route can be saved when it is possible to salvage a packet using a route from an intermediate node.

However, in a network in which the nodes are highly mobile, it is likely that the route in the intermediate node's cache was older, and hence, also invalid. Trying to salvage a data packet by using another bad route would result in a waste of time and bandwidth.

To confirm that the above features do in fact hurt the performance of DSR, I ran simulations comparing the performance of *original DSR* with variations of DSR in which 1 or both of the above features were turned off. The gratuitous replies optimization, described in Section 2.1.2, introduces several security vulnerabilities [3] and was therefore turned off in all variations of DSR that I simulated.

To turn off intermediate-node replies, I modified DSR such that an intermediate node that hears a route request broadcasts it further, without replying to it even if a route to the destination exists in its cache. Only the destination nodes to route requests are allowed to reply. The idea behind this is that it might be a good idea for a route request to travel a few hops further to the destination, and get a reply that is assured to be both fresh and valid.

To turn off Data Salvage, I modified DSR such that an intermediate node that experiences a link failure does not try to salvage the data. Instead, it sends a route error back to the source, and drops the data packet.

3.3. Performance Analysis of original DSR

Simulation Environment

All simulations were performed using the Glomosim network simulator [10]. The modifications were made to the implementation of DSR written for Glomosim. The mobility model used was *random waypoint* [11] in a rectangular field. In random waypoint, each node starts its journey from its current location to a random location within the field. The speed is randomly chosen to be between 1-19 m/sec. Once the destination is reached, another random destination is targeted after a specified pause. Continuous node mobility is achieved by keeping the pause time as 0 seconds, which is the default used in my simulations.

Traffic sources used were CBR (constant bit-rate) over UDP. The source-destination pairs were spread randomly over the network. Only 512 byte data packets were used. The number of source-destination pairs was fixed to 25 and the packet sending rate in each pair was varied to change the *offered load* in the network. Simulations were run for 600 simulated seconds. A 100 node network in a field size of 1000m x 1000m was used.

Performance metrics

I evaluated two key performance metrics:

1. **Delivery Rate:** The delivery rate is calculated as a ratio of the data packets delivered to the destination to those generated by the CBR sources.

2. **Average Route Age** – The route age is calculated by subtracting the route request time from the current time when a route is used to send a data packet. The average of all the route ages is the second metric used in my analysis. For example, suppose a source node A sends a packet to destination B at 120 seconds from the start of the simulation. Also assume that the route request time indicated in the route used is 80 seconds. The age of the route in this case is (120-80) or 40 seconds. For a given data packet, this calculation is done only at the source. At the end of the simulation, the sum of all the route ages used is divided by the number of times the above calculation was done to get the average route age.

Simulation Results

Figure 1 shows a comparison of delivery rates achieved by original DSR and the variations of DSR in which one or both of the above features are turned off. Figure 2 shows a comparison of the average route ages by the same set of DSR variations.

'IN OFF' refers to the variation of DSR in which Intermediate-Node replies are turned off, 'DS OFF' refers to the variation in which Data Salvage is turned off, and 'BOTH OFF' refers to the variation in which both Intermediate-Node replies and Data Salvage are turned off.

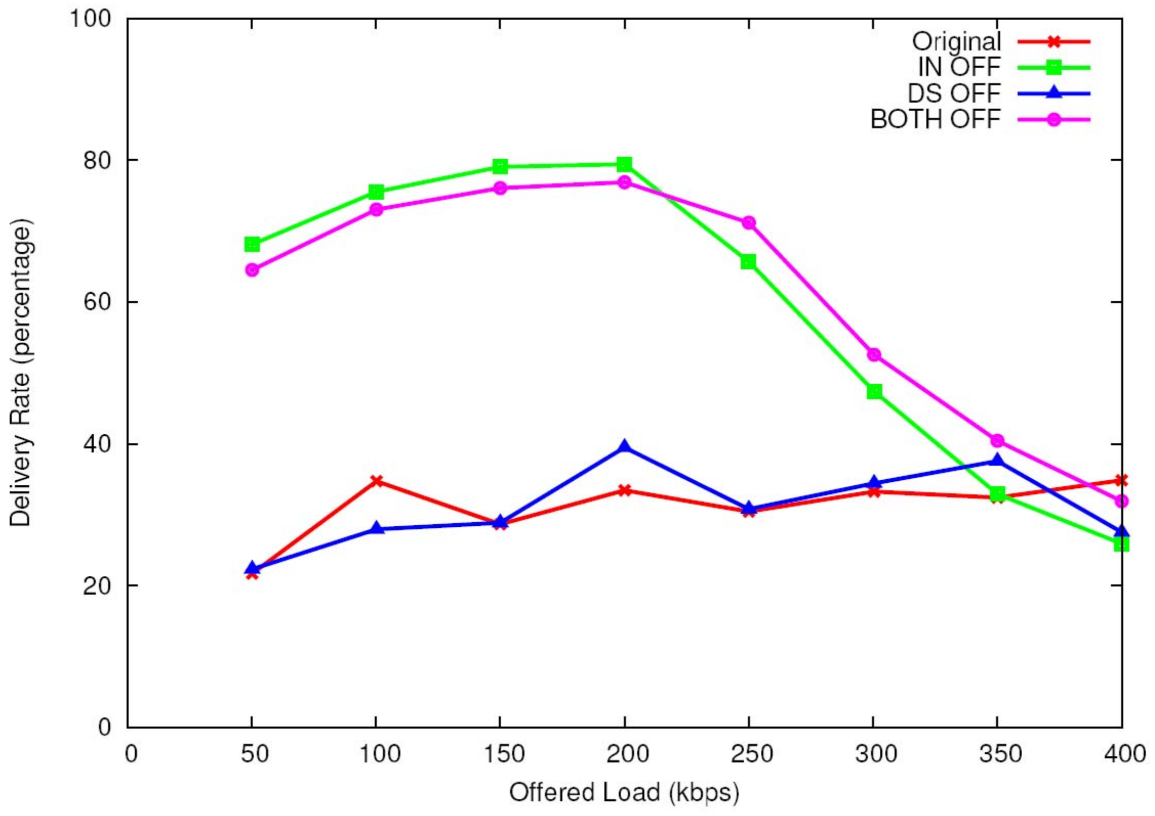

Figure 1: Delivery rate vs. Load

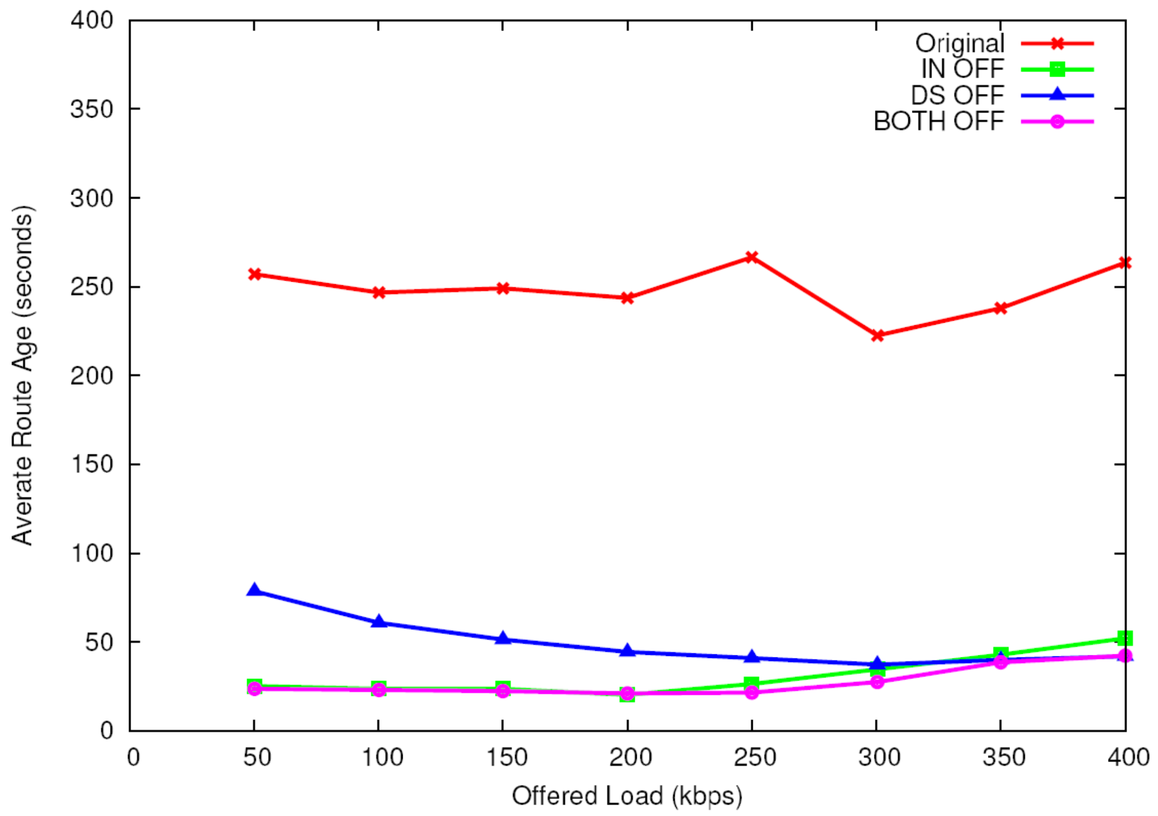

Figure 2: Average route age vs. Load

Figures 1 and 2 suggest some correlation between the average age of the routes used and the throughput achieved. Turning off intermediate-node replies reduces the average route age by a considerable amount. This results in significantly improved delivery rates as shown by Figure 1. The performance improvement is less significant when data salvage is turned off. However, given that malicious node detection becomes harder with data salvage [3], turning it off is preferable.

Because of the performance and security concerns described above, I will define ‘Base DSR’ as the version of DSR with both Intermediate-Node replies and Data-Salvage turned off, for the analysis that follows.

4. Proposed techniques to improve performance

Even after intermediate-node replies and data-salvage are turned off, the performance of DSR is not satisfactory. The throughput achieved by Base DSR at a load of 50 Kbps in a typical network configuration is about 32 Kbps (64% delivery rate), whereas, the throughput achieved by AODV (another routing protocol designed for Ad Hoc Networks) in a similar configuration is close to 40 Kbps (80% delivery rate).

To further improve the performance of the Base DSR, I evaluate the following techniques based on my observations of AODV and other protocols:

Technique 1: Limiting Replies from Destination
In the original implementation of DSR, a destination node replies to every route request packet that it hears. This, however, results in a lot of unnecessary route replies when the same route request is heard by a destination multiple times. This can also result in ‘bad’ routes being added to the route cache of the source. For instance, consider 2 route requests that take the same number of hops, but different paths to reach the destination at different times. The request that reaches the destination late possibly took a path that was more congested. Instead of being discarded, this request is also replied to, and because it had the same hop count as the previous request, it is added to the top of the route cache of the source. Hence, when a data packet is to be sent, a congested route is tried before the route that was not congested.

I modified DSR such that destination nodes will reply only if

a) The last route request from the given source was older than the current one, or
b) The last route request was made at the same time (the same route request took different routes to the destination) but the current request took fewer hops than the last one.

The destination now sends a route reply only if it is a new route request or a better route for a route request to which it has already replied.

Technique 2: Giving preference to 'fresher' routes in cache
When a node learns multiple routes to a destination, DSR stores them in its cache table ordered by hop count. This, however, often results in the use of older, invalid routes over routes that are fresh, and valid. While hop count may be a good metric to determine the order of routes in the cache, it should probably not be the only criteria.

I modified the cache structure of DSR such that it would maintain routes to a particular destination in the following order:

a) A route with a later request time is given preference over a route with an earlier request time.
b) If the request times of two routes are the same, then a route with shorter hop count is given preference over a longer route.
c) If both the request time and the hop count of two routes are the same, then a route with a later reply time is given preference over a route with an earlier reply time.

Technique 3: Keeping a single route per destination
If routes are ordered by freshness (as described under Technique 2), and the first route fails, it is very likely that the older routes stored in the cache will also fail. By trying all the routes in the cache before sending a new route request, a lot of time and bandwidth is wasted.

I modified the cache structure such that a node will now keep only the best route to any given destination (based on the criteria described under technique 2). If that route fails, the node will issue a route request, and learn a fresh route.

5. Performance Evaluation of proposed techniques

I used a detailed simulation study to evaluate the effectiveness of the techniques for the Base DSR (DSR with IN and DS turned off; See Section 3.3). The simulation environment used was as described in Section 3.3.

Besides packet delivery fraction and average route age, I also evaluate routing overhead as a performance metric. The routing overhead is the total number of control packets that are sent every second throughout the network.

5.1. Simulation Results

Figures 3, 4 and 5 show a comparison of ‘Base DSR’ with variations of DSR ‘enhanced’ by one or more of the techniques described above. Figure 3 shows a comparison of the Packet Delivery Fraction (in percentages) achieved by the different variations of DSR. Figure 4 shows a comparison of the average route age, and figure 5 shows a comparison of the routing overhead incurred per second in the 3 modified variations of DSR and ‘base’ DSR.
In figures 3, 4 and 5, ‘Base’ refers to the version of DSR with Intermediate-Node replies and Data Salvage turned off. ‘Base + LR’ refers to the variation in which technique 1 described above is implemented. ‘Base + LR + FR’ refers to the variation in which both technique 1 and technique 2 are implemented, and ‘Base + FR + 1R’ refers to the variation in which all 3 of the techniques described above are implemented.

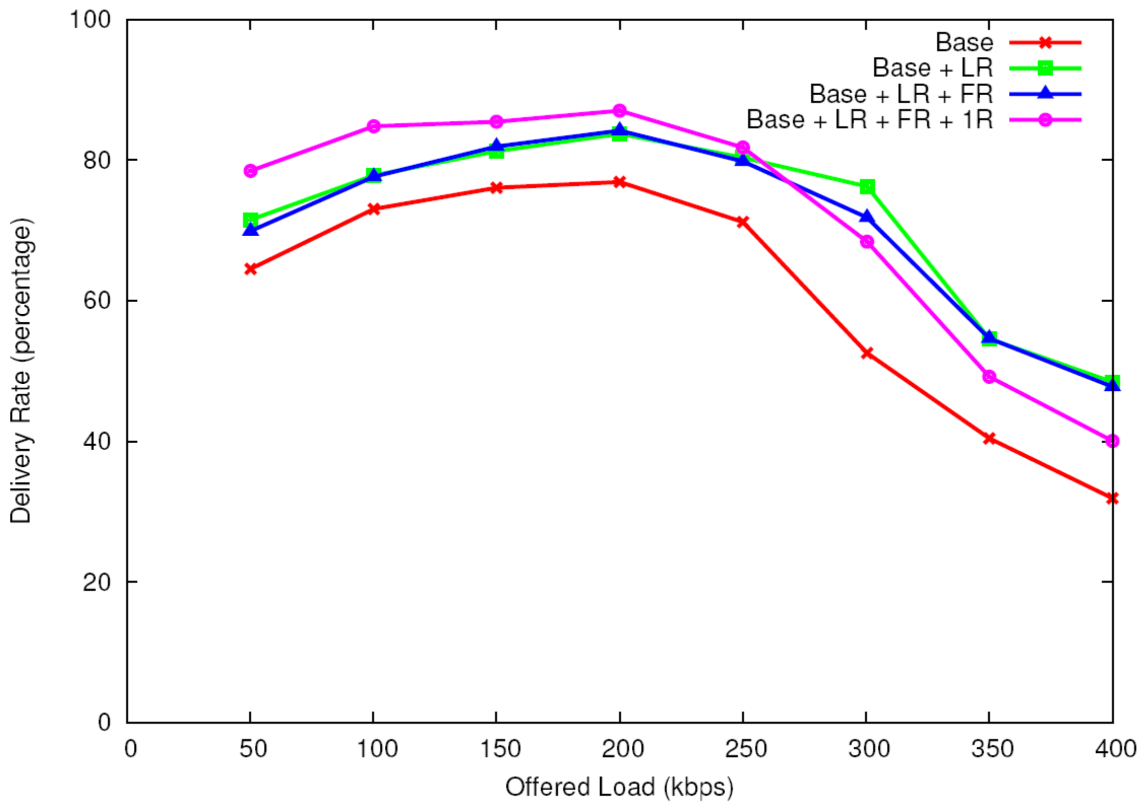

Figure 3: Performance analysis of suggested techniques (Delivery Rate vs. Load)

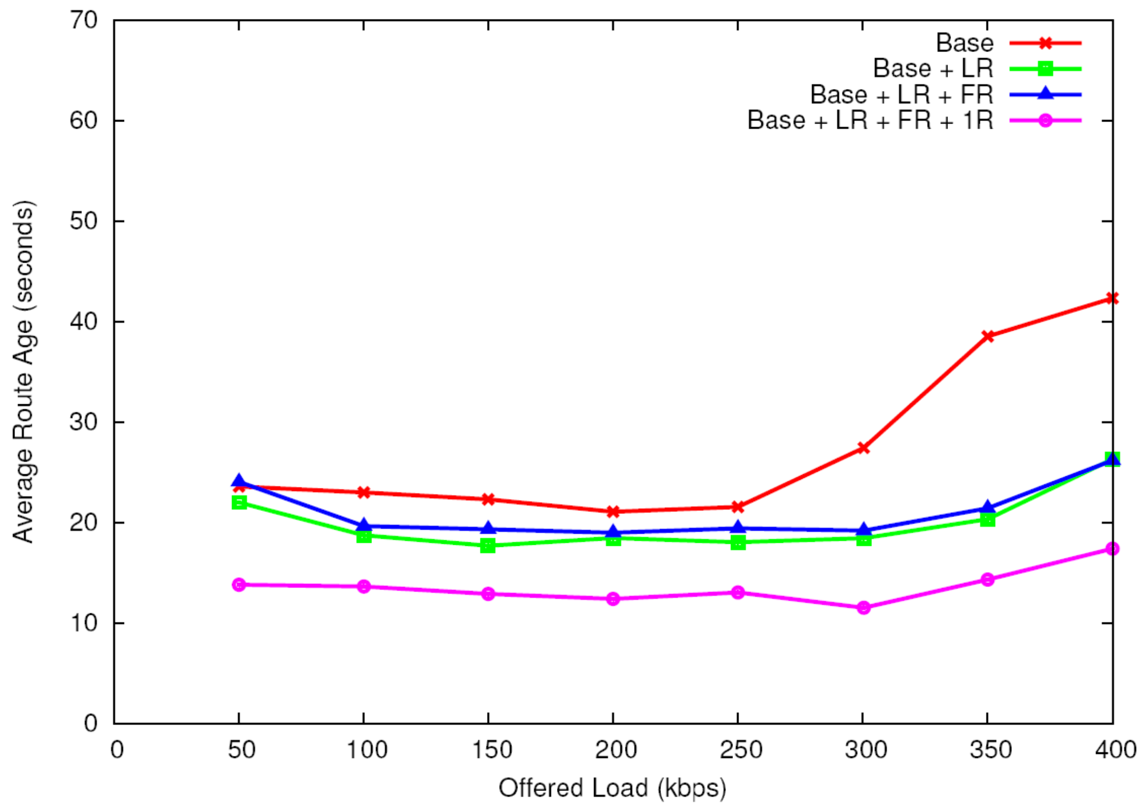

Figure 4: Effect of suggested techniques on the average route age

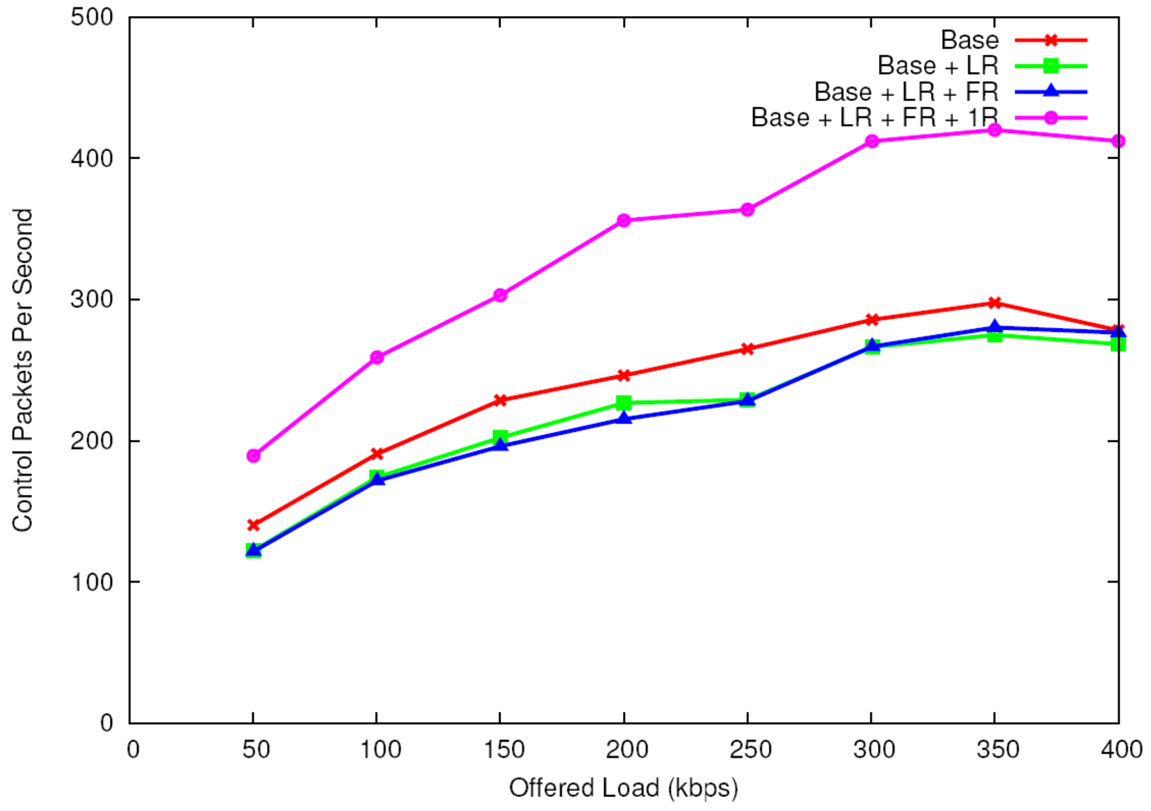

Figure 5: Number of control packets sent per second throughout the network

In Figure 3, we notice that 'Base + LR + FR + 1R' achieves the best performance for low values of offered load. The decrease in delivery rate for higher loads can be explained by the near saturation of the network, a situation where most routes are congested. In such a scenario, congested links could be wrongly identified as 'broken', resulting in route errors and route requests propagating throughout the network. Keeping only 1 route incurs an increase in the routing overhead (for example, when the load is increased from 250 Kbps to 300 Kbps in Figure 5), and can hurt the performance in a saturated, or almost saturated network.

Figure 4 shows that the average age of the routes used is decreased in all 3 of the 'enhanced' versions. The lowest average route age is achieved by 'Base + LR + FR + 1R'. This is not surprising because this version tries only the 'freshest' route that it has learned. The low average age probably results in the improved performance.

Figure 5 gives us a picture of the routing overhead incurred by the different techniques. 'Base + LR' and 'Base + LR + FR' have less routing overhead than 'Base DSR' because 'limited replies' reduce the number of route replies in the network. However, 'Base + LR + FR + 1R' has the maximum amount of overhead among all the version of DSR simulated. This is to be expected due to the increased number of route requests caused by having only 1 route in the cache.

6. Statistical Analysis

While graphs such as those presented in the previous section can give a general overview of the performance changes caused by the different techniques, a more rigorous analysis is needed in order to determine the relative impacts of the techniques on performance.

I used a 2^3 factorial design [13] in order to determine the effect of the 3 techniques proposed. Such a factorial design makes it easy to analyze and helps in sorting out the factors in the order of impact. This can help us decide which of the 3 techniques has the most impact on performance, which has the least impact, and how a combination of 2 or more of the factors affects performance.

In order to analyze behavior in different network conditions, I simulated 3 different field sizes: 1000m x 1000m (Medium density), 2200m x 600m (Low density) and 1500 x 300 (High density). The number of nodes was kept constant at 100 and the radio range was kept at 250 m. The node density is computed as follows:

$$\textit{Node density} = (N \times \Pi r^2)/(L \times W) \textit{ nodes/radio range}$$

Where N is the number of nodes, r is the radio range, and L and W are the length and width of the field, respectively.

The node densities simulated are:
Medium density: 19.6 nodes/radio range
Low density: 14.8 nodes/radio range
High density: 43.6 nodes/radio range

Each factor used in the analysis is varied between two values (denoted as levels -1 and +1). A total of 2^k simulations need to be conducted varying the factors of interest systematically, keeping all other input parameters fixed, for the analysis. Tables 1, 2 and 3 show the results when load was kept constant at 100 Kbps (Low Traffic). The factors considered and their values at each level are given below.

Symbol	**Factor**	**Level -1**	**Level 1**
A	Limited Replies	Destination replies to all requests	Destination replies to “limited” replies
B	Cache Route Order	By Hop Count	By Freshness
C	# of Routes in Cache	Multiple	One

There are eight rows corresponding to each combination of factor levels; the achieved delivery rate for each case is indicated in the last column (labeled y). Each column labeled with a combination of A, B, and C indicates a factor combination. The effect of each factor combination on the performance is calculated as

Dot Product{Column for the factor, Column y} / 8.

For factor A in Table 1, it is

$$(-1)(73.2) + (1)(78) + (-1)(71) + (1)(77.8) + (-1)(82.7) + (1)(84) + (-1)(82.8) + (1)(85)$$
$$= 1.89.$$

The value calculated for column ‘I’ is the average delivery rate, denoted y_{av}. In Table 1, it is 79.3.

The total variation of y or sum of squares total (SST) is calculated as

$$\text{Total variation of } y = SST = \Sigma(y_i - y_{av})^2.$$

The variations of various combinations of factors are calculated. For example,

$$\text{Sum of squares due to A (SSA)} = (\text{Effect of A})^2 \times 2^3$$
$$= 28.5$$

Now, the fraction of total variation due to each factor (or combination of factors) can be calculated as the ratio of sum of squares value for that combination and SST. For example,

$$\text{Fraction of variance due to A} = SSA/SST$$
$$= 28.5/188$$
$$= 0.15$$

Table 1 Factorial Analysis of three routing techniques at 100 Kbps load for medium network density

		Limited Replies	Cache Route Order	# Routes in Cache					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	73.20
	1	1	-1	-1	-1	-1	1	1	78.00
	1	-1	1	-1	-1	1	-1	1	71.00
	1	1	1	-1	1	-1	-1	-1	77.80
	1	-1	-1	1	1	-1	-1	1	82.70
	1	1	-1	1	-1	1	-1	-1	84.00
	1	-1	1	1	-1	-1	1	-1	82.80
	1	1	1	1	1	1	1	1	85.00
	634.50	15.10	-1.30	34.50	2.90	-8.10	3.50	-1.10	
Factor effect	79.31	1.89	-0.16	4.31	0.36	-1.01	0.44	-0.14	
Sum of squares	1.88E+02	2.85E+01	2.11E-01	1.49E+02	1.05E+00	8.20E+00	1.53E+00	1.51E-01	
Variation explained by factor	1.00	0.15	0.00	0.79	0.01	0.04	0.01	0.00	

Table 2 Factorial Analysis of 3 routing techniques at 100 Kbps for low network density

		Limited Replies	Cache Route Order	# Routes in Cache					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	64.50
	1	1	-1	-1	-1	-1	1	1	67.90
	1	-1	1	-1	-1	1	-1	1	62.20
	1	1	1	-1	1	-1	-1	-1	68.50
	1	-1	-1	1	1	-1	-1	1	70.20
	1	1	-1	1	-1	1	-1	-1	72.20
	1	-1	1	1	-1	-1	1	-1	70.80
	1	1	1	1	1	1	1	1	71.60
	547.90	12.50	-1.70	21.70	1.70	-6.90	1.70	-4.10	
Factor effect	68.49	1.56	-0.21	2.71	0.21	-0.86	0.21	-0.51	
Sum of squares	8.75E+01	1.95E+01	3.61E-01	5.89E+01	3.61E-01	5.95E+00	3.61E-01	2.10E+00	
Variation explained by factor	1.00	0.22	0.00	0.67	0.00	0.07	0.00	0.02	

Table 3 Factorial Analysis of three routing techniques at 100 Kbps for high network density

		Limited Replies	Cache Route Order	# Routes in Cache					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	74.70
	1	1	-1	-1	-1	-1	1	1	80.10
	1	-1	1	-1	-1	1	-1	1	72.30
	1	1	1	-1	1	-1	-1	-1	78.50
	1	-1	-1	1	1	-1	-1	1	81.10
	1	1	-1	1	-1	1	-1	-1	86.30
	1	-1	1	1	-1	-1	1	-1	80.60
	1	1	1	1	1	1	1	1	86.90
	640.50	23.10	-3.90	29.30	1.90	-0.10	4.10	0.30	
Factor effect	80.06	2.89	-0.49	3.66	0.24	-0.01	0.51	0.04	
Sum of squares	1.78E+02	6.67E+01	1.90E+00	1.07E+02	4.51E-01	1.25E-03	2.10E+00	1.12E-02	
Variation explained by factor	1.00	0.37	0.01	0.60	0.00	0.00	0.01	0.00	

Tables 1, 2 and 3 show that the highest fraction of variance in low traffic environments is explained by the number of routes in the cache (0.79, 0.67 and 0.6 of the variation in medium, low and high density networks, respectively). Keeping a single route in the cache has the largest impact and gives consistently better performance. Limited Replies also have some impact on performance (0.15, 0.22 and 0.37). A small fraction of the variance is explained by the combination of having limited replies and keeping a single route in the cache at the same time. The impact due to the cache route order is negligible in all 3 cases.

In tables 4, 5 and 6, the same factors are analyzed, but the load is changed to 200 Kbps (Medium Traffic). The network densities are also the same as before (Medium, Low and High in tables 4, 5, and 6 respectively). Loads higher than 200 Kbps were not analyzed because the network seems to reach saturation at loads higher than that.

Table 4 Factorial Analysis of three routing techniques at 200 Kbps for medium network density

		Limited Replies	Cache Route Order	# Routes in Cache					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	76.90
	1	1	-1	-1	-1	-1	1	1	83.75
	1	-1	1	-1	-1	1	-1	1	74.20
	1	1	1	-1	1	-1	-1	-1	84.20
	1	-1	-1	1	1	-1	-1	1	70.30
	1	1	-1	1	-1	1	-1	-1	86.90
	1	-1	1	1	-1	-1	1	-1	70.45
	1	1	1	1	1	1	1	1	87.05
	633.75	50.05	-1.95	-4.35	3.15	16.35	2.55	-3.15	
Factor effect	79.22	6.26	-0.24	-0.54	0.39	2.04	0.32	-0.39	
Sum of squares	3.53E+02	3.13E+02	4.75E-01	2.37E+00	1.24E+00	3.34E+01	8.13E-01	1.24E+00	
Variation explained by factor	1.00	0.89	0.00	0.01	0.00	0.09	0.00	0.00	

Table 5 Factorial Analysis of 3 techniques at 200 Kbps for low network density

		Limited Replies	Cache Route Order	# Routes in Cache					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	60.55
	1	1	-1	-1	-1	-1	1	1	70.15
	1	-1	1	-1	-1	1	-1	1	55.85
	1	1	1	-1	1	-1	-1	-1	73.05
	1	-1	-1	1	1	-1	-1	1	43.30
	1	1	-1	1	-1	1	-1	-1	72.40
	1	-1	1	1	-1	-1	1	-1	45.30
	1	1	1	1	1	1	1	1	73.00
	493.60	83.60	0.80	-25.60	6.20	30.00	4.40	-9.00	
Factor effect	61.70	10.45	0.10	-3.20	0.77	3.75	0.55	-1.13	
Sum of squares	1.09E+03	8.74E+02	8.00E-02	8.19E+01	4.80E+00	1.13E+02	2.42E+00	1.01E+01	
Variation explained by factor	1.00	0.80	0.00	0.08	0.00	0.10	0.00	0.01	

Table 6 Factorial Analysis of 3 techniques at 200 Kbps for high network density

		Limited Replies	Cache Route Order	# Routes in Cache					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	58.95
	1	1	-1	-1	-1	-1	1	1	82.40
	1	-1	1	-1	-1	1	-1	1	58.25
	1	1	1	-1	1	-1	-1	-1	82.70
	1	-1	-1	1	1	-1	-1	1	36.50
	1	1	-1	1	-1	1	-1	-1	89.75
	1	-1	1	1	-1	-1	1	-1	35.20
	1	1	1	1	1	1	1	1	87.55
	531.30	153.50	-3.90	-33.30	0.10	57.70	-3.10	-1.90	
Factor effect	66.41	19.19	-0.49	-4.16	0.01	7.21	-0.39	-0.24	
Sum of squares	3.50E+03	2.95E+03	1.90E+00	1.39E+02	1.25E-03	4.16E+02	1.20E+00	4.51E-01	
Variation explained by factor	1.00	0.84	0.00	0.04	0.00	0.12	0.00	0.00	

From Tables 4, 5, and 6, a substantial fraction of variance in medium traffic environments is explained by having limited replies (0.89, 0.8 and 0.84 in medium, low and high density networks, respectively). The impact due to the number of routes, as the load becomes high, is much less (0.01, 0.08 and 0.04). The combination of having limited replies and a single route in the cache is slightly more significant (0.09, 0.1, and 0.12). The impact due to the cache route order, once again, is negligible.

It is evident that the cache route order has little to no impact in all the 6 network configurations simulated (with varied network density and load). It might be interesting to see, however, how the other 2 factors interact with load and what effect that has on performance. Therefore, I have analyzed limited replies, number of routes in the cache, and load as the three factors in tables 7, 8 and 9. Routes are ordered by freshness in all 3 cases. The values for limited replies and route cache order for levels -1 and 1 are the same as used in tables 1-6. For load, -1 represents 100 Kbps and 1 represents 200 Kbps.

Table 7 Factorial Analysis of limited replies and number of routes for medium network density

		Limited Replies	# Routes in Cache	Load					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	71.00
	1	1	-1	-1	-1	-1	1	1	77.80
	1	-1	1	-1	-1	1	-1	1	82.80
	1	1	1	-1	1	-1	-1	-1	85.00
	1	-1	-1	1	1	-1	-1	1	74.20
	1	1	-1	1	-1	1	-1	-1	84.20
	1	-1	1	1	-1	-1	1	-1	70.45
	1	1	1	1	1	1	1	1	87.05
	632.50	35.60	18.10	-0.70	2.00	17.60	-19.90	11.20	
Factor effect	79.06	4.45	2.26	-0.09	0.25	2.20	-2.49	1.40	
Sum of squares	3.04E+02	1.58E+02	4.10E+01	6.13E-02	5.00E-01	3.87E+01	4.95E+01	1.57E+01	
Variation explained by factor	1.00	0.52	0.13	0.00	0.00	0.13	0.16	0.05	

Table 8 Factorial Analysis of limited replies and number of routes for low network density

		Limited Replies	# Routes in Cache	Load					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	62.20
	1	1	-1	-1	-1	-1	1	1	68.50
	1	-1	1	-1	-1	1	-1	1	70.80
	1	1	1	-1	1	-1	-1	-1	71.60
	1	-1	-1	1	1	-1	-1	1	55.85
	1	1	-1	1	-1	1	-1	-1	73.05
	1	-1	1	1	-1	-1	1	-1	45.30
	1	1	1	1	1	1	1	1	73.00
	520.30	52.00	1.10	-25.90	5.00	37.80	-22.30	16.00	
Factor effect	65.04	6.50	0.14	-3.24	0.63	4.73	-2.79	2.00	
Sum of squares	6.98E+02	3.38E+02	1.51E-01	8.39E+01	3.13E+00	1.79E+02	6.22E+01	3.20E+01	
Variation explained by factor	1.00	0.48	0.00	0.12	0.00	0.26	0.09	0.05	

Table 9 Factorial Analysis of limited replies and number of routes for high network density

		Limited Replies	# Routes in Cache	Load					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	72.30
	1	1	-1	-1	-1	-1	1	1	78.50
	1	-1	1	-1	-1	1	-1	1	80.60
	1	1	1	-1	1	-1	-1	-1	86.90
	1	-1	-1	1	1	-1	-1	1	58.25
	1	1	-1	1	-1	1	-1	-1	82.70
	1	-1	1	1	-1	-1	1	-1	35.20
	1	1	1	1	1	1	1	1	87.55
	582.00	89.30	-1.50	-54.60	28.00	64.30	-34.90	27.80	
Factor effect	72.75	11.16	-0.19	-6.83	3.50	8.04	-4.36	3.48	
Sum of squares	2.23E+03	9.97E+02	2.81E-01	3.73E+02	9.80E+01	5.17E+02	1.52E+02	9.66E+01	
Variation explained by factor	1.00	0.45	0.00	0.17	0.04	0.23	0.07	0.04	

From Tables 7, 8, and 9, limited replies seem to have the most impact on performance (0.52, 0.48 and 0.45) in all 3 network densities, irrespective of the number of routes in cache and load. The number of routes in the cache has little to no impact by itself (0.13, 0, 0). The impact of having a high load combined with one of the other factors is substantial (0.13, 0.26 and 0.23 when combined with limited replies).

We have shown that 2 of the 3 techniques suggested have a substantial impact on performance. The improvement caused due to the 2 techniques is also dependent on the offered load. While the third technique does not have a significant impact, it does not hurt the performance. For a more complete analysis, it might be useful to analyze how our enhanced DSR (with all 3 techniques implemented) compares with Base DSR in different loads and network densities.

Table 10 presents such a factorial analysis with the following factors and levels.

Symbol	**Factor**	**Level -1**	**Level 1**
A	DSR	Base	Enhanced
B	Traffic Load	100 Kbps (Low)	200 Kbps (Medium)
C	Network Density	Low (2200 x 600)	High (1500 x 300)

Table 10 Factorial analysis of routing techniques, traffic load and network density

		DSR	Load	Density					
	I	A	B	C	AB	AC	BC	ABC	y
	1	-1	-1	-1	1	1	1	-1	64.50
	1	1	-1	-1	-1	-1	1	1	71.60
	1	-1	1	-1	-1	1	-1	1	60.55
	1	1	1	-1	1	-1	-1	-1	73.00
	1	-1	-1	1	1	-1	-1	1	74.70
	1	1	-1	1	-1	1	-1	-1	86.90
	1	-1	1	1	-1	-1	1	-1	58.95
	1	1	1	1	1	1	1	1	87.55
	577.75	60.35	-17.65	38.45	21.75	21.25	-12.55	11.05	
Factor effect	72.22	7.54	-2.21	4.81	2.72	2.66	-1.57	1.38	
Sum of squares	8.30E+02	4.55E+02	3.89E+01	1.85E+02	5.91E+01	5.64E+01	1.97E+01	1.53E+01	
Variation explained by factor	1.00	0.55	0.05	0.22	0.07	0.07	0.02	0.02	

From Table 10, it appears that enhanced DSR is a significant improvement over base DSR irrespective of load or network density. The version of the DSR used and network density explain for most of the impact on the delivery rate (0.55 and 0.22 respectively) in the analysis in Table 10.

6. Conclusions and future work

I have identified route staleness as the major cause of poor performance by DSR. By analyzing the average route age and delivery rate at different loads, I was able to demonstrate that simply removing certain features of DSR (intermediate node replies and data salvage) results in significantly lower route ages, and consequently, better performance.

I then proposed 3 techniques in order to further enhance the performance of DSR. Two of these techniques (limiting replies and keeping a single route in the cache) are shown to have a significant positive impact on performance. The third technique (ordering routes by freshness) is not as significant in comparison, but does improve the performance.

A factorial analysis demonstrates that while load and network density have an impact on performance, the impact due to the routing protocol is much more significant. Enhanced DSR performs better than original DSR in different kinds of network environments, and has the most positive impact when compared with factors such as load and density.

In my future work, I plan to investigate better metrics for caching routes. For instance, route discovery time may be a better metric than hop count as it would indicate distance as well as congestion between 2 nodes. I also plan to research the impact of varying mobility on network performance. Because DSR is likely to be used as a protocol where data security is important, I also plan to investigate secure communication using DSR, and the impact of security overhead on performance.

7. References

1. D. Johnson, D. Maltz and Y. Hu. The dynamic source routing protocol for mobile ad hoc networks. IETF MANET Working Group, Internet Draft 2003.
2. T. Jiang, Q. Li, and Y. Ruan. Secure Dynamic Source Routing Protocol. In Proceedings of the Forth International Conference on Computer and Information Technology (CIT), 2004.
3. R. N. Mir and A. M. Wani. Security Analysis of Two On-Demand Routing Protocols in Ad Hoc Networks. In Proceedings of ACM MOBIHOC 2001.
4. D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, edited by Tomasz Emilienski and Hank Korth, Kluwer Academic Publishers, 1996.
5. M. K. Marina and S. R. Das. Performance of Route Caching Strategies in Dynamic Source Routing. In Proceedings of Int'l Workshop on Wireless Networks and Mobile Computing (WNMC), 2001.
6. Y. Hu and D. Johnson. Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (Mobicom), 2000.
7. W. Lou and Y. Fang. Predictive Caching Strategy for On-Demand Routing Protocols in Wireless Ad Hoc Networks, Wireless Networks, vol. 8, no. 6, 2002.
8. T. Goff et al., Preemptive Routing in Ad Hoc Networks. In Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (Mobicom), 2001.
9. D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In Proceedings of MobiCom 2003.
10. X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: A library for parallel simulation of large-scale wireless networks," in *Workshop on Parallel and Distributed Simulation*, 1998, pp. 154–161.
11. T. Camp, J. Boleng, V. Davies. A survey of mobility models for Ad Hoc Network Research. In Wireless Communication and Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications. vol. 2, no. 5, 2002.
12. C. E. Perkins, E. M. Royer, S. R. Das, M. K. Marina, Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks, IEEE Personal Communications 8 (1) (2001) 16–28.
13. Raj Jain. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons, 1991.
14. M.K. Marina and S. R. Das, Impact of Caching and MAC Overheads on Routing Performance in Ad Hoc Networks, Computer Communications, 2003.

Vita

Anket Mathur was born in New Delhi, India on December 20, 1982, the son of Rajendra K. Mathur and Madhu Mathur. After graduating from Delhi Public School in 2000, he attended Georgia Institute of Technology from 2000 to 2003. At Georgia Institute of Technology, he was part of the co-op program and worked with Erdas, inc. in alternating semesters between 2001 and 2002. In Summer 2004, he joined the University of Texas at San Antonio to pursue a Bachelor of Science degree in Computer Science.

Publications in Computer Science

R. V. Boppana and A. Mathur. Analysis of the Dynamic Source Routing Protocol for Ad Hoc Networks. To appear in *First IEEE International Workshop on Next Generation Wireless Networks 2005 (IEEE WoNGeN '05)*, Dec. 2005