Adaptive local searching and caching strategies for on-demand routing protocols in ad hoc networks

Zhao Cheng, Wendi B. Heinzelman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, NY 14627
(585) 275-{8078, 4053}
{zhcheng, wheinzel}@ece.rochester.edu

Abstract

On-demand routing protocols are widely used in mobile ad hoc networks due to their capability of adjusting to frequent network topology changes within acceptable routing overhead. In order to further reduce routing overhead, especially the overhead from the network-wide flooding in the route discovery phase, two techniques named route caching and searching localization are usually performed. In this paper, we reinvestigate these two techniques, in particular their joint effect on the routing overhead. For quantitative analysis purposes, we define one essential parameter for each technique: route caching validation probability and local searching radius. Based on the analytic results, we propose a new routing strategy that adapts to the current caching availability and is self-tunable towards the optimal performance. We demonstrate through extensive simulations that this routing strategy can reduce the routing overhead greatly under general scenarios.

1 Introduction

In ad hoc networks, there is no pre-existing fixed network architecture. Mobile nodes, typically with similar transmission and computational capabilities, cooperate by forwarding packets for nodes that are not in each other's direct transmission range. The properties of ad hoc networks such as node mobility, limited available bandwidth and the broadcast nature of the wireless medium make the design of efficient routing protocols for ad hoc networks more challenging than for traditional networks.

Routing protocols proposed for ad hoc networks can be roughly divided into two categories: table-driven (proactive) and on-demand (reactive). On-demand protocols typically have lower routing overhead than table-driven protocols, and thus attract more interest. This is because they only initiate a route discovery process when a packet is ready to be transmitted, which removes the necessity of persistent maintenance of a routing table using costy shortest path algorithms. Typical examples of these reactive protocols are DSR [1], AODV [2]. However, despite the reduced routing overhead using this reactive approach, the performance is still not satisfactory. Two primary techniques are introduced to improve the performance: route caching and searching localization.

Route caching plays an important role in reducing both the overhead and the discovery latency. After storing a route cache from the route discovery phase, a node is able to send a new packet without delay and respond to route requests from other nodes without further broadcasting. However, the routing overhead cannot be effectively reduced by those intermediate nodes due to the flooding nature of route requests. Also, a stale cache may bring about even more routing overhead and even longer packet delay.

Caching optimizations have been extensively researched to fully exploit benefits from caches. However, in this paper, we point out that in order to obtain full benefits, a local searching scheme must be performed in cooperation with the caching scheme. Also, a more accurate method for quantifying the quality of caches is required in order to avoid the adverse effects from stale caches. The major contribution of our paper is to provide a method to accurately measure the quality of caches and determine the optimal local search radius according to current caching conditions (see Section 3). Based on the analytical results, we present our local-searching and caching strategy that can adaptively adjust itself to the caching availability conditions and approach the optimal performance in Section 4. Finally, we implement this strategy in DSR and demonstrate its advantage through extensive simulations in Section 5.

2 Overview and related work

When a node has a packet to send but has no route to the destination node, it initiates a route discovery procedure by broadcasting a Route REQuest (RREQ) packet. Upon receiving a RREQ, if an intermediate node has a cached route for the destination or itself is the destination, it unicasts a Route RESponse (RRES) following the reversed route back to the source node. The discovered route will be stored in a route cache by the source node for future use. Intermediate nodes without route caches for the target attach their addresses in the RREQ packet and continue to flood the packet.

Caching strategies and caching designs for DSR are studied in [3]. The authors achieved the optimal choices for timeout and route cache capacity through exhaustive searching over specific scenarios. In our paper, we not only study the effects of caching, but also the joint effects of searching localization. Some optimizations, such as *salvaging*, *gratuitous route repair* and *promiscuous listening*, have been proposed and have been shown to be effective in reducing stale caches and improving the performance of route caching [1]. Some other proactive optimizations, such as *Negative caches*, *Wider error notification* and *Adaptive timeout selection*, are proposed in [4]. These schemes, although not taken into account in our analysis and simulations, can cooperate with our routing strategies seamlessly. Their existence only changes the caching availability conditions in the network, while our routing strategy is able to adjust itself based on the caching conditions and achieve the optimal performance.

Compared to the extensive studies on route caching, study in the searching localization area is relatively lacking. Although LAR [5] is able to localize its querying area, it requires geographical information, which we do not assume in our study. In DSR, the *non-propagating route request technique* is performed by the source node to search one-hop neighbors first before resorting to a network-wide flood. In AODV [2], an expansion ring searching scheme is proposed. A source node starts a route discovery process with an initial searching radius and increases the searching radius linearly upon each failure. A network-wide search is performed when the searching radius exceeds a predefined threshold. However, in [6], it is shown that when the existence of caching availability in the network is weak (e.g., in networks with infrequent traffic), using one-hop local searching has an only insignificant savings in overhead, while the expansion ring scheme has more overhead than a simple network-wide flooding. Another searching localization technique is also proposed in [7]. It utilizes prior routing histories but does not take route caches into account. Our scheme also utilizes prior route histories, but in a different manner, and our paper concentrates on the joint effects of the route caching and the local searching techniques rather than only one of these. Also, in contrast to the experimental study on cache validation and optimization schemes in [8], our study exposes the underlying relationship between routing overhead and caching optimization methods quantitatively.

The authors in [9] studied the effects of DSR with both caching and local searching, and they mentioned the possible ineffectiveness of the expansion ring technique under weak caching existence. They compared the performance of one specific expansion ring scheme with the one-hop local searching scheme and analyzed when a node should switch from one scheme to the other. In our paper, we do not consider the expansion ring scheme. Instead, we analytically determine the optimal local searching radius among all the possible choices in different scenarios and propose our protocol to realize it. To the best of our knowledge, this is the first study on finding the optimal performance of on-demand routing protocols for general scenarios with both techniques applied.

3 Model and analysis

3.1 Nomenclature and assumptions

Without loss of generality, let us consider N homogenous nodes with unit transmission range that are randomly distributed in a disk of radius R. N is large enough to form a network with good connectivity [10, 11]. Nodes move with the maximum speed of S_m in a random waypoint method. Each node has a total event rate of λ_T. In this paper, *event* indicates a one-way traffic flow towards a destination that is randomly selected from all the other nodes. The arrival of the events is a random variable that follows a poisson distribution, and each event lasts for a fixed lifetime T_l. During this lifetime, it is not necessary for the traffic to be continuous. For example, for an event with a lifetime of 10 seconds, there may be only 10 packets, or one packet per second. We denote **Src** as the source node and **D** as the destination node.

During our analysis, we assume that we are studying the DSR protocol with only the options of *gratuitous route repair* and *non-propagation route request* turned on. Without *gratuitous route repair*, after a RERR is received, the source node will receive invalid caches from intermediate nodes each time it resends the RREQ. The loop of RREQ-invalid cache-RERR will continue until the cache in the intermediate nodes expires. The performance of the protocol without this option is too poor to be studied. *Non-propagation route request* is the same as our local searching technique and will be fully studied. Furthermore, we assume that each node has at most one route cache entry for each destination. To avoid reply storms, we also assume that the destination only replies to the first route query packet.

In the remainder of this section, we will first introduce two crucial protocol parameters. Then we reveal the relationship between routing overhead and these two parameters. Finally, we give the optimal numerical results for these two parameters to maximize the overhead reduction. All these results will become the guideline for the protocol design in Section 4.

3.2 Route caching validation probability

The first term we are going to introduce is *route caching validation probability* P_v, which expresses the probability for a route cache to be valid. By valid route, we mean that a packet can follow this cached route to reach the destination. P_v is related to three factors: the maximum node speed S_m, the number of links L contained in the route and the elapsed time T since its last use. It is obvious that the larger the value of S_m, T and L, the less probability P_v for the route to remain valid. In other words, the function $P_v(S_m, L, T)$ decreases monotonically and continuously as its variables increase. $P_v(S_m, L, T)$ can be decomposed as

$$P_v(S_m, L, T) = P_{lv}^L(S_m T) \tag{1}$$

Here, $P_{lv}(t)$ is the probability for an originally connected link to remain valid after the two nodes of the link move for a time period t with a random speed from $[0, 1]$. This transformation simplifies the route validation problem into a unit speed link validation problem. This transformation is valid since S_m can be seen as a scale for time, and also, for a route cache to be valid, each of its independent L links must remain connected after time T. Although the lifetime of different routes may be correlated by certain common links, the lifetimes of different links within one specific route are independent of each other. This is validated by our simulations.

The definition of P_v has several practical meanings. First, it allows the source node to determine the quality of cache it requires by appending a validation threshold value p_t in the RREQ packets. Intermediate nodes with route caches only respond if they have a qualified route cache with its calculated P_v larger than p_t. By adjusting p_t, the source node is able to adapt to the current caching situation and reduce unnecessary RREQ packets. Second, P_v allows the source node to determine which route to choose when several RRES packets are received. Of course, the closer the P_v is to 1 and the shorter the route length is, the more likely this returned route is chosen. Third, P_v also helps with route caching management. Nodes can remove a route cache automatically if its P_v is lower than a certain value or remove the cache with the lowest P_v when the route caching table is full. Overall, the introduction of P_v provides more options for handling route caches more accurately than using cache lifetime alone.

3.3 Local searching radius

A local searching scheme must work with a caching scheme to be meaningful. In [6], it is shown that when there is no caching, even the optimal local searching scheme can reduce the searching overhead only by an amount of at most 8% while bringing about excessive latency.

In general, a local search has two-sided effects on the searching overhead. If a local search finds the target itself or it finds cached routes to the target in intermediate nodes, the network-wide flood can be avoided if the route cache is correct, and the searching overhead is reduced effectively. However, if this search fails to return any result, a network-wide search is still required and the overall searching cost is even more than a simple network-wide flood. Thus, the local searching radius k should be carefully chosen to achieve the most benefit from the local search. If the radius k is chosen too large, the probability of discovering a route returned from the destination itself is large and little benefit can be obtained from the route caches. If the radius k is chosen too small, the chance of discovering the destination or a cached route to the destination in this local search is also small and little benefit can be gained from this local search because the first round local search will be part of the total searching overhead. As we will show later, the radius k is a crucial parameter in determining the RREQ overhead and is also closely related to the amount of caching in the network.

3.4 Life periods of a cache

To understand how a node responds to other nodes' route requests, we need to clarify the life periods of a route cache first. The time interval between two traffic events from a certain **Src** to a certain **D** can be divided into three caching periods, as shown in Fig. 1, which are the guaranteed valid period I, the probable valid period II and the invalid period III. In different periods, a route cache is of different qualities and has different effects on the routing overhead and the routing performance.

Starting from the leftmost of Fig. 1, when a new event, say event i, just arrives, **Src** initiates a route discovery process and caches the discovered route for future use. During period I, all of the traffic packets of this event will follow this cached route. Meanwhile, if the route is broken due to node mobility, the route maintenance will detect it and initiate a new route discovery. Thus, during the event lifetime T_l, this node maintains a valid route for **D**. More strictly speaking, during period I, a node can respond to a RREQ from other nodes with a route cache whose validation probability P_v is very close to 1.

During life period II when there is no more traffic between **Src** and **D**, there are no more proactive methods such as route maintenance to refresh the cached route. As time

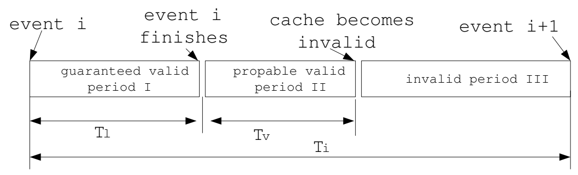

Figure 1. Time periods for a node's route caching between two events towards the same destination. In Period I, the cache is almost valid all the time. In the probable valid period II, the node has stopped sending traffic to the destination and therefore has only a probable valid route cache. In the invalid period III, the route cache is of very low quality.

elapses, this cached route becomes invalid gradually, i.e., P_v decreases. If a RREQ arrives with the validation threshold p_t, a route cache will be returned in response only if its validation probability P_v satisfies $p_t < P_v < 1$. In other words, when p_t is given, the time length T_v of life period II can be explicitly found by solving $P_v(S_m, L, T_v) = p_t$, or $P_{lv}^L(S_m T_v) = p_t$.

During Period III, P_v is below the threshold p_t. Nodes no longer respond to a RRES and we consider the cache invalid. This period lasts until the next event arrives and a new round starts.

When a route request arrives at the intermediate node **I** randomly at any time, it may fall into any of the above three periods. The probability of falling into each period is determined by the traffic pattern and evaluation of p_t. Also, the route cache validation probability varies for RRES packets from different periods.

3.5 Overhead reduction ratio (ORR)

The primary goal of this research is to achieve the minimum routing overhead, i.e., to reduce the number of RREQ and RRES packets as much as possible. We define *overhead reduction ratio* to measure the effectiveness of the combined caching and local searching schemes. Since RREQ packets are required to be flooded while RRES packets are unicasted, RREQ packets are dominant in the overall routing overhead and we only consider RREQ packets in the measurement of ORR. Suppose that with the caching and local searching schemes, the overhead is O, and without these schemes, the overhead is O_n. Then ORR is defined as

$$ORR = \frac{O_n - O}{O_n} \qquad (2)$$

Suppose there are $N_l(k)$ nodes in the k-hop local area, and the total number of nodes in the network is N. When doing a local searching with radius k, the expected RREQ overhead O falls into the following cases:

1. When node **D** is local: cost is local $N_l(k)$.
2. When node **D** is non-local and a guaranteed valid route cache is found locally: cost is local $N_l(k)$.
3. When node **D** is non-local and a guaranteed valid route is not found but a probable valid route is found, and it is valid: cost is local $N_l(k)$.
4. When node **D** is non-local and a guaranteed valid route is not found but a probable valid route is found, and it is invalid: cost is local plus network-wide $N_l(k) + N$.
5. When node **D** is non-local and no cache is found: cost is local plus network-wide $N_l(k) + N$.

The probability of the above five cases can be derived as well when k and p_t are given. Due to space constraints, we skip the analysis details here. We find that: $ORR(k, p_t) = (1 - \frac{N_l(k)}{N})(P_g + P_p p_t)$, where P_g denotes the probability of receiving at least one guaranteed valid route cache and P_p denotes the probability of receiving at least one probable valid route cache. This equation indicates that the overhead can be reduced when the target is not in the local area but either a guaranteed valid cache or a probable valid cache, which **is** valid, is found locally.

ORR is also related to other non-protocol related parameters such as network parameters, node mobility, traffic rates and the event life. All these parameters actually determine the caching availability conditions of the entire network. It is obvious that when the node mobility is low, or the traffic is frequent, or the event life is long, route caches will be abundant and of high qualities, and vice versa.

Our analysis shows that when the caching condition is moderate or weak, choosing $k = \lfloor \frac{M}{2} \rfloor$ and $p_t = 0.4$ will provide the largest overhead reduction (M is the network diameter and can be determined using methods provided in [6].) When the caching condition is abundant, numerical results show that the searching radius k and p_t should be set where a cache is very likely to be found to further reduce the overhead. In the next section, we will design a routing scheme based on these results that is able to dynamically adjust itself to approach the maximum performance under all possible scenarios.

4 Protocol description

Only minor changes are needed for existing on-demand routing protocols to fully attain the benefits of caches. Two primary parameters are needed, the caching validation probability threshold p_t and the local searching radius k. When the network is just set up, or a node just joins a network, these values should be set to $p_t = 0.4$ and $k = \lfloor \frac{M}{2} \rfloor$, assuming weak or moderate caching conditions. When more abundant caching conditions are detected based on history, k should be set to a smaller value accordingly. Also, p_t

can be adjusted larger to reduce unnecessary caches of low qualities. In this section, we propose an add-on protocol for existing routing protocols. Its three components, new data structures, protocol procedures and parameter adjustment rules, will be described in detail next.

4.1 New data structures

A new field for caching validation probability is required for both the RREQ and the RRES packets. For RREQ packets, the value of this field is calculated through the parameter adjustment rules described below and appended in the RREQ packets to serve as the caching validation threshold. For RRES packets, the value of this field is calculated by the node that initiates the RRES packet to indicate the cache's quality using equation 1.

Also, each node maintains a statistic such as the number of recent RREQ attempts, the values of k and p_t applied, the number of guaranteed valid caches and the number of probable valid caches received. This information is used to estimate the current caching condition to serve for the parameter adjustment rules. The counting of these numbers does not differentiate destinations and only needs a little extra storage. This non-differential counting is valid for uniform traffic scenarios. For biased traffic scenarios such as the client-server traffic model, a maintenance of the history of different destinations may provide more accurate parameter adjustment. The tradeoff is much larger extra storage for each destination node. In our current work, we utilize the general statistical method without destination differentiation.

4.2 Protocol procedure

When a source node needs to send a RREQ, it calculates the parameters k and p_t according to the parameter adjustment rules and attaches the values in the RREQ packet. Intermediate nodes calculate P_v for their cached route to the destination from equation 1 and return a RRES packet with P_v attached if P_v satisfies $P_v > p_t$. The source node picks the cached route with the largest P_v. When two cached routes have close P_v values, the one with a shorter route length is preferred. Each node refreshes the statistics each time it sends out a RREQ packet and receives RRES packets from intermediate nodes.

4.3 Parameter adjustment rules

The parameter adjustment rules determine the value of p_t according to the current caching situation. A node first calculates the average number of guaranteed valid route caches N_g and the average number of probable valid route caches N_p received from its history, say the last 100 RREQ attempts. Also, from the history it calculates the averages $\tilde{k}$ and $\tilde{p_t}$. These values indicate the current caching conditions. If k already equals $\lfloor \frac{M}{2} \rfloor$ and p_t is already 0.4 and N_p and N_g are still less than 1, there is no need to further increase k and p_t since this is a weak or moderate caching condition and the protocol is already running using optimal parameters. A running average over all the past RREQ attempts instead of the last 100 attempts requires less storage. However, it cannot represent the most recent caching conditions and is less accurate for the parameter adjustment. Therefore, there is a tradeoff between storage and parameter adjustment accuracy.

When N_g is larger than 1, guaranteed valid caches become the dominating factor. k should be primarily adjusted according to N_g towards the goal of receiving only one guaranteed cache by using $k = \frac{\tilde{k}}{\tilde{N_g}}$. In a more general case, the average number of guaranteed valid caches is much lower than 1 and the probable valid caches are prevailing. Thus, k should be adjusted towards the goal of $N_p = 1$ by using $k = \frac{\tilde{k}}{\tilde{N_p}}$. If k is already as low as 1 and there are still more than necessary caches returned, p_t should be adjusted larger to accept only more qualified caches by using $p_t = \frac{\tilde{p_t}}{\tilde{N_p}}$. This indicates a very abundant caching condition such as when the node speed is very low and traffic between nodes happens very often. In summary, k and p_t are adjusted towards receiving one guaranteed valid cache, or one probable valid cache when the chance of receiving guaranteed valid caches is small. However, the adjustment of k should not exceed $\lfloor \frac{M}{2} \rfloor$, and the adjustment of p_t should not be lower than 0.4. Exceeding these values only brings about more routing overhead although it may bring about more returned caches. However, during our simulations, we notice that the adjustment towards the goal of receiving only one probable valid cache from our analysis is a little conservative in finding qualified caches in some cases. This is because our analysis does not differentiate destinations in the non-local area for simplicity purpose. We simply use a more aggressive parameter adjustment method by setting $k = \frac{2\tilde{k}}{\tilde{N_p}}$ towards the goal of receiving two probable valid caches. This simple modification can provide good enough performance and avoid the excessive storage required by the per destination based statistic maintenance.

5 Performance evaluation

5.1 Assumptions, metrics and methodology

We simulate our routing scheme as an add-on to the DSR protocol in a mobile ad hoc network scenario. The simulations are performed using ns-2 [12]. In order to focus our study in the routing level, we programmed an ideal lower layer below the routing layer, which is able to send packets without collision and detect link failures automatically with no time and no cost. Working with this virtual bottom layer, the routing protocol can be approximately seen as working at low traffic. We believe that a realistic MAC will have negligible effect due to the broadcast nature of RREQ packets. For unicasting packets such as RRES, the MAC layer should have the same impact on our scheme as on traditional schemes since there is little modification on the packet structures.

We test all the scenarios in a square region of size $1400m \times 1400m$. There are 150 nodes inside this area, each moving in a random waypoint mobility pattern. The number of nodes is chosen large enough for good connectivity as well as to make it easy to investigate the performance difference between our scheme and the original DSR scheme. Each node has a transmission range of 250m, and the estimated maximum hop value is 7. The maximum node speeds of 1m/s or 10m/s are used. The total simulation time is 4500 seconds, long enough to remove the potential undesirable effects of starting the random waypoint mobility model simultaneously for all the nodes [13].

In our simulations, basic metrics commonly used by former studies [1] are investigated, which are routing overhead, successful delivery ratio, discovery latency and route optimality. However, in order to concentrate on our topic of reducing routing overhead, we only show in the figures the metrics that have significantly changed. Other metrics with little changes will just be briefly mentioned.

We study the performance of three routing schemes: the original DSR with No Caching (DSR-NC), DSR with Caching (DSR-C) and DSR with our scheme of Local searching and Caching added on (DSR-LC). We first validate our results of the selection of k and p_t through exhaustive simulations on DSR-LC. Then we simulate DSR-C and show that the selection of the timeout value has a similar impact on the performance as the selection of p_t in DSR-LC. Finally, we compare the performance of DSR-LC, DSR-C and DSR-NC under different scenarios.

5.2 DSR-LC, effects of k and p_t

In this part, we will validate the claim that $k = \lfloor \frac{M}{2} \rfloor$ and $p_t = 0.4$ are optimal values by testing all the possible k and p_t values in a scenario with moderate caching availability. First, we fix p_t at 0.4 and change k from 1 to 4. From the results shown in Fig. 2, we can see that $k = 3$ is optimal for the number of RREQ, which matches with our analytical result $k = \lfloor \frac{M}{2} \rfloor = \lfloor \frac{7}{2} \rfloor = 3$. The number of RRES is not of the same order as the number of RREQ.

Next, we fix k at 3 and change p_t from 0 to 0.6. The simulation results are shown in Fig. 3. From Fig. 3, some point between 0.3 and 0.4 is a good balance point for p_t. Before

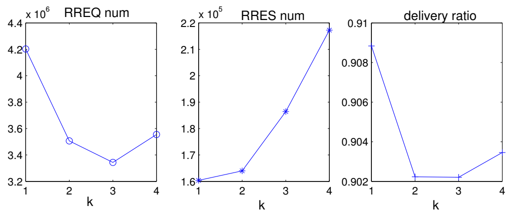

Figure 2. Performance of DSR-LC with p_t fixed at 0.4. The X-axis indicates the local searching radius k, ranging from 1 to 4. The optimal point is at $k = 3$ for the number of RREQ with almost no effect on other metrics.

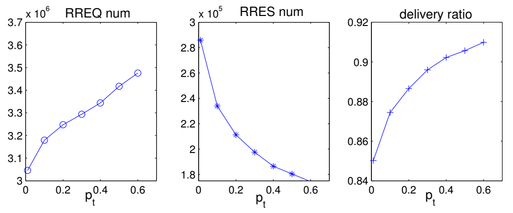

Figure 3. Performance of DSR-LC with k fixed at 3. The X-axis indicates the route caching validation probability threshold p_t, ranging from 0 to 0.6. The tradeoff is between the number of RREQ and the the number of RRES plus the delivery ratio. A good balance point is at $p_t = 0.4$.

this point, the increase of p_t leads to an approximately linear increase of RREQ while leading to a *faster* decrease of RRES and a *faster* increase of the packet delivery ratio. The knee of the curves of the RRES number and the packet deliver ratio is at around 0.4. Also, considering the delivery ratio to be larger than 90%, we choose p_t equal to 0.4 for the rest of the simulations. The study of p_t also shows that we can trade the routing overhead for the packet delivery ratio by adjusting p_t.

5.3 DSR-C, effects of timeout

We test DSR-C with the route cache timeout value of 5s, 10s, 20s and 30s, and the results are shown in Fig. 4. In order to compare the results with the selection of p_t in DSR-LC shown in Fig. 3, we reverse the order of the cache timeout values on purpose.

As shown in Fig. 4, there is a similar trend and tradeoff for the timeout value as there was for p_t shown in Fig. 3. The relationship between the timeout and p_t can be partially explained by equation 1. The larger the time for a route

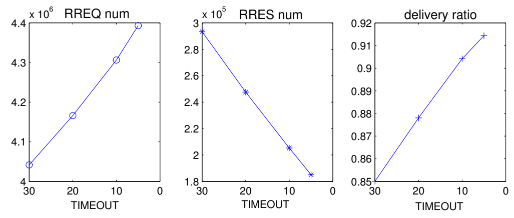

Figure 4. Performance of DSR-C with one-hop neighbor searching. The X-axis indicates the timeout value, ranging from 30 seconds to 5 seconds. The tradeoff is also between the RREQ number and the delivery ratio. Just like Fig. 3, the increase of TIMEOUT causes both metrics to decrease. A good balance point is at TIMEOUT=10 seconds.

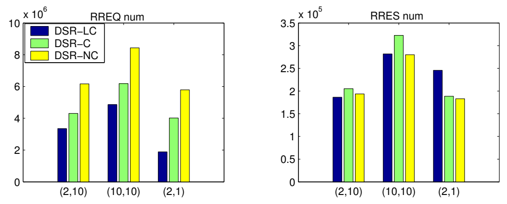

Figure 5. Routing overhead comparisons under a low event rate of 0.05 events/sec. The X-axis indicates different scenarios tested, from left to right stands for (event life, node speed) pairs of (2s, 10m/s), (10s,10m/s), (2s,1m/s). This figure shows the effects of event lifetime and the node speed on routing overhead in a moderate caching condition.

cache to be stale, the easier it is for a route request to be satisfied with certain cached routes. However, the sacrifice is a higher number of RRES packets and a lower packet delivery ratio due to stale routes. We pick the balance point TIMEOUT equal to 10s as the representative for DSR-C to ensure that the delivery ratio is larger than 90%.

5.4 DSR-LC, DSR-C and DSR-NC

In this part, we will compare these three routing schemes under different caching conditions by different traffic rates, node speeds, event lifetimes and traffic patterns. First, we experiment in a moderate caching condition with a low event rate of 0.05 events per second. We test the scenarios with the duplex [event lifetime, maximum node speed S_m] valued at [2s, 10m/s], [10s, 10m/s] and [2s, 1m/s]. These scenarios can all be categorized as moderate caching availability. From the results shown in Fig. 5, DSR-LC achieves a significantly lower routing overhead for this low event rate, despite the fact that the savings may vary depending on the other parameters.

Next, we test an extreme scenario with abundant caching availability. The event rate is as high as 0.5 events per second and the maximum node speed is as low as 1m/s. As shown in Fig. 6, the overhead reduction ratio of DSR-LC in this abundant caching scenario is as high as about 80% compared to DSR-NC, thanks to the caches. DSR-LC eventually adjusts its local searching radius to around 1 and p_t to around 0.65. DSR-C may achieve a closer number of RREQ packets if it adjusts its timeout value to 40 seconds (shown in the fourth column) instead of 10 seconds. However, the number of RRES packets increases correspondingly since more route caches are available for returning. DSR-LC, with the adjustment of both k and p_t, restrains the number of both RREQ and RRES in a satisfactory range.

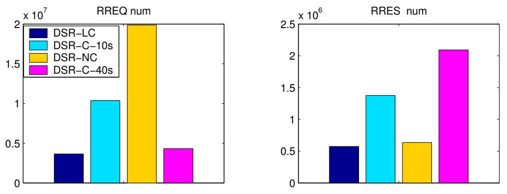

Figure 6. Routing overhead comparisons under a high event rate of 0.5 events/sec and a low maximum node speed of 1m/s. This figure shows the performance of different schemes in an abundant caching condition.

In the above experiments, each node has the same traffic pattern as other nodes and has events toward other nodes with equal probability. In contrast with this peer-to-peer traffic model, we experiment with a client-server traffic model. In this model, we choose 20% of the nodes as servers and 80% of the total traffic is towards these servers. The results shown in Fig. 7 are based on a maximum node speed of 10m/s and a total event rate of 0.05. As can be seen from this figure, the shift from a peer-to-peer model to a client-server model reduces the overhead reduction ratio of DSR-LC compared to DSR-C but increases the overhead reduction ratio of DSR-LC compared to DSR-NC. This is reasonable since the client-server model implies a more abundant cache availability. For this reason, DSR-LC eventually adjusts k to an average of around 1.3 in the client-server model, while it adjusts k to an average of around 2.5 in the peer-to-peer model. Thus, in the client-server model, DSR-C with local searching radius fixed at 1 is already close to the optimal value, and therefore, the number of RREQ packets in DSR-C is close to that in DSR-LC. However, for the

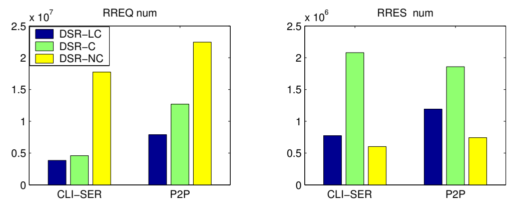

Figure 7. Routing overhead comparisons for peer-to-peer and client-server traffic models.

same reason illustrated in the last paragraph, DSR-C has a large number of RRES packets when route caches are abundant.

Overall, DSR-LC can achieve an overhead reduction up to 80% compared to DSR-NC and up to 40% compared to DSR-C, depending on the caching level in the network. When the route cache availability is moderate, DSR-LC has a larger ORR compared to DSR-C. When route caches are abundant, DSR-LC has less overhead reduction in RREQ packets compared to DSR-C while it has much larger reduction compared to DSR-NC. Besides, DSR-LC can restrain the number of RRES packets by adjusting p_t without degrading cache qualities, while DSR-C does not have an effective method to control the number of RRESs.

6 Conclusions and future work

The main contributions of this paper are to determine the optimal value for parameters of local searching radius and route caches valid probability threshold and to propose an adaptive routing strategy that can be easily added on to existing on-demand protocols. The new scheme adjusts the local searching radius and the required caching quality according to the caching conditions. It reduces routing overhead consistently and significantly, for both RREQ and RRES packets, with almost no effects on other performance aspects.

Further research on the parameter adjustment and the performance of our routing scheme working with specific MAC protocols is needed. One avenue of future work is to take into account more history information to aid in determining the local searching radius. Another avenue is to investigate how our scheme performs in a more realistic model. Currently, we ignore the existence of a MAC layer and the physical layer. In our future work, we will examine the performance improvement of our scheme over the popular ad hoc MAC protocol 802.11 and realistic wireless channels. However, as explained earlier in section 5, we believe that the impact of realistic MAC layers and wireless channels should be negligible.

References

[1] D.B.Johnson and D.A.Maltz. *Mobile Computing*, Chapter Dynamic source routing in ad hoc wireless networks, pages 153-181. Kluwer Academic Publishers, Imielinski and Korth edition, 1996.

[2] C.Perkins and E.M.Royer. "Ad hoc on-demand distance vector routing" *Proceedings of IEEE WMCSA'99*, pp. 90–100, Feb. 1999.

[3] Y.-C. Hu and D. B. Johnson, "Caching strategies in on-demand routing protocols for wireless networks," *Proc. ACM/IEEE MobiCom*, August 2000.

[4] M. K. Marina and S. R. Das, "Performance of Route Caching Strategies in Dynamic Source Routing," *Proc. of WNMC in conjunction with ICDCS*, pp. 425–432, 2001.

[5] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks,"*In ACM/IEEE Int. Conf. MobiCom'98*, Oct. 1998

[6] Z. Cheng, W. Heinzelman,"Flooding strategy for target discovery in wireless networks," *Proc. of the 8th MSWIM 2003*, Sep 2003.

[7] R. Castaneda and S. Das, "Query localization techniques for on-demand routing protocols in ad hoc networks," *in Proc. ACM/IEEE MOBICOM '99*, 1999, pp. 186–194.

[8] N. Panchal and N. B. Abu-Ghazaleh, "Active Route Cache Optimization for On-Demand Ad Hoc Routing Protocols", *ICN 2004*

[9] J. Sucec and I. Marsic, "An Application of Parameter Estimation to Route Discovery by On-Demand Routing Protocols", *Proc. ICDCS 2001*, pp. 207–218.

[10] Feng Xue and P. R. Kumar. "The number of neighbors needed for connectivity of wireless networks" Manuscript, 2002. Available from http://black1.csl.uiuc.edu/prkumar/postscript files.html

[11] B. Krishnamachari, S. Wicker, R. Bejar, and M. Pearlman, "Critical Density Thresholds in Distributed Wireless Networks," *in Communications, Information and Network Security, Kluwer Publishers*, 2002.

[12] http://www.isi.edu/nsnam/ns/ns-documentation

[13] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *WCMC: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol 2, pp. 483–502*, 2002.