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

Zhao Cheng* and Wendi B. Heinzelman

Department of Electrical and Computer Engineering,
University of Rochester, Rochester, NY 14627, USA
Fax: (585) 275-8078 Fax: (585) 275-4053
E-mail: zhcheng@ece.rochester.edu E-mail: wheinzel@ece.rochester.edu
*Corresponding author

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 this paper, we reinvestigate two techniques, route caching and searching localisation, in particular their joint effect on reducing routing overhead. We first analyse and present the optimal value for two essential parameters: route caching validation probability and local searching radius. We then propose a new routing strategy that adapts to the current caching availability and is self-tunable towards the optimal performance. Simulation shows that routing overhead is thus greatly reduced.

Keywords: cache; local searching; adaptive.

Reference to this paper should be made as follows: Cheng, Z. and Heinzelman, W.B. (2006) 'Adaptive local searching and caching strategies for on-demand routing protocols in ad hoc networks', *Int. J. High Performance Computing and Networking*, Vol. 4, Nos. 1/2, pp.51–65.

Biographical notes: Zhao Cheng is a PhD candidate in the Department of Electrical and Computer Engineering at the University of Rochester. He received a BS Degree in Radio Engineering from Southeast University, China in 2000 and MS Degree in Electrical and Computer Engineering from University of Rochester in 2003. His current research interests lie in the areas of sensor networks, Quality of Service (QoS) and reliability for mobile ad-hoc networks, and efficient discovery strategies for mobile ad-hoc networks.

Wendi B. Heinzelman is an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Rochester. She received a BS Degree in Electrical Engineering from Cornell University in 1995 and MS and PhD Degrees in Electrical Engineering and Computer Science from MIT in 1997 and 2000, respectively. Her current research interests lie in the areas of sensor networks, Quality of Service (QoS) and reliability for mobile ad hoc networks, and multimedia communication. She is a member of Sigma Xi, the IEEE, and the ACM.

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). Typical examples of table-driven protocols are DSDV (Perkins and Bhagwat, 1994), OLSR (Jacquet et al., 2001) and WRP (Murthy and Garcia-Luna-Aceves, 1996). These protocols require nodes to maintain a route table for all the other nodes so that a route is always available when a packet is ready to be transmitted. However, on-demand protocols attract more interest than table-driven protocols because they only initiate a route discovery process when a packet is ready to be transmitted. Without the necessity of persistent maintenance of a routing table, where shortest path algorithms are usually applied, on-demand protocols typically have lower routing overhead than table-driven protocols. Typical examples of these reactive protocols are DSR (Johnson and Maltz, 1996), AODV (Perkins and Royer, 1999) and TORA (Park and Corson, 1997). However, despite the reduced routing overhead using this reactive approach, the performance is still not satisfactory. The main reason is packet flooding in the route discovery process. First, the Route REQuest (RREQ) has to be flooded through the whole network, which leads to both large overhead and congestion. Second, packets have to be queued until the route query process finishes and, therefore, the latency for these packets is increased. Two primary techniques are introduced to solve these problems: route caching and searching localisation.

Route caching plays an important role in reducing both the overhead and the discovery latency. With a route caching scheme, once a route is discovered, a node stores it in a route caching table. This route is cached for two purposes. First, when the node has another packet for the same destination, it may refer to this route cache instead of initiating a new route query process. Second, when the node hears a route query from another node for the same destination, it may return the cached route to the querying node instead of continuing to forward the query. Route caching reduces the routing overhead originating from the source node as well as reducing the discovery latency for other nodes.

However, applying caching alone is not as effective as expected. This is mainly caused by the flooding property of route queries and the frequent network topology changes. First, in a well-connected network, even if an intermediate node returns a cached route and stops forwarding the query, it cannot quench the flooding. The flooded query will get around this node through other directions and continue to flood the whole network, just like water flowing down from a mountaintop will reach the ground even though some boulders may block the way. Therefore, an intermediate node cannot effectively reduce the routing overhead by returning a cached route. Second, although the discovery latency of the first packet that triggers the route discovery process is decreased by faster responses from intermediate nodes, this gain may be counteracted by even larger propagation latency for the following packets. This is because if the route cache is non-optimal (i.e., not the shortest path), the remaining packets that utilise this non-optimal route will have a larger latency than necessary. Overall, fast responses from route caching only benefit the first several packets while impairing the rest of the packets if the applied cached route is non-optimal. Third, the worst case happens when the returned route cache is invalid, (that is, the cached route is broken). If such a cached route is applied in sending data packets, the source node has to restart the route discovery process after receiving a route error message returned from the broken link. The unwanted consequence is that both the routing overhead and latency are increased, plus several data packets are lost due to the broken route.

Caching optimisations 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 main 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. Through analysis, we quantify the overall routing overhead based on two parameters that characterises the local searching strategy and the caching strategy. 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. We then implement this strategy in DSR and demonstrate its advantage through extensive simulations.

The reminder of the paper is organised as follows. Section 2 gives an overview of on-demand protocols and details previous studies on the route caching and searching localisation techniques. Section 3 presents our analysis on optimising the protocol parameters. In Section 4, we propose our routing strategy based on earlier analysis results. Section 5 validates our analysis and demonstrates the performance of DSR with our routing strategy by comparing with the original DSR. Section 6 concludes the paper.

2 Overview and related work

An on-demand routing protocol is composed of two phases, the route discovery phase and the route maintenance phase, both of which operate reactively and entirely on demand. In the route discovery phase, taking DSR for example, a source node floods a RREQ when it has a packet to send but has no route to the destination node. Upon receiving the RREQ packet, intermediate nodes without route caches for the target attach their addresses in the RREQ packet and continue to flood the packet. If an intermediate node has a cached route for the destination or the destination is reached by the RREQ packet, it unicasts a Route RESponse (RRES) following the reversed route back to the source node. After discovering a route and placing it in the cache table, the source node switches into the maintenance phase. In this phase, packets follow the cached route instead of initiating a new discovery process. If a packet fails to be forwarded through one of the links indicated in the source route, the intermediate node at the upstream of this broken link will unicast a Route ERRor (RERR) packet to the source node, indicating the broken link. The source node then removes all the route caches that contain the broken link and initiates a new route discovery process for an alternate route.

Caching strategies and caching designs for DSR are studied in Hu and Johnson (2000). 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 effects of searching localisation. Furthermore, our caching strategies are mainly based on analysis and the simulations are for validation purposes.

Some optimisations have been proposed and have been shown to be effective in reducing stale caches and improving the performance of route caching (Johnson and Maltz, 1996). The *salvaging* technique allows intermediate nodes to use an alternate route cache of its own when the attached route in the packet is broken. The *gratuitous route repair* technique helps intermediate nodes to remove stale caches by piggybacking the last RERR information in the new RREQ packet. The *promiscuous listening* technique takes advantage of the broadcast property of the wireless medium and helps overhearing nodes to learn the network topology without directly participating in the routing process. Some other proactive optimisations are proposed in Marina and Das (2001). *Wider error notification* can be performed to further reduce the existence of stale caches. *Negative caches* can reside in nodes to prevent them from adding invalid caches. *Adaptive timeout selection* can avoid removing valid caches by observing route stability and dynamically choosing timeout values. These schemes, although not taken into account in our analysis and simulations, can cooperate with our routing strategies directly. Their existence only changes the caching availability conditions in the network, while our routing strategy is able to adjust itself adaptively based on the caching conditions and achieve the optimal performance.

Compared to the extensive studies on route caching, study in the searching localisation area is relatively lacking. Although LAR (Ko and Vaidya, 1998) is able to localise 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 neighbours first before resorting to a network-wide flood. In AODV (Perkins and Royer, 1999), 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, these two techniques are proposed without solid theoretical support. In Cheng and Heinzelman (2003), it is shown that when the existence of caching availability in the network is weak (i.e., 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. When the existence of caching is moderate, using one-hop local searching is likely to be too conservative and a larger searching radius can reduce the routing overhead even more, as shown later in this paper. When route caches are abundant, one-hop local searching becomes a good scheme because there is no need to search farther for route caches in this case. Although we do not analyse the performance of the expansion ring scheme in this paper, we find that using only one local search may increase the discovery latency significantly. The discovery latency is likely to be too large to be acceptable with an expansion ring approach. Another searching localisation technique is also proposed in Castaneda and Das (1999). It utilises prior routing histories but does not take route caches into account. Our scheme also utilises 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 optimisation schemes in Panchal and Abu-Ghazaleh (2004), our study exposes the underlying relationship between routing overhead and caching optimisation methods through quantitative analysis.

The authors in Sucec and Marsic (2001) 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 analysed 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 realise 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 (Xue and Kumar, 2002; Krishnamachari et al., 2002). Nodes move with the maximum speed of S_m in a random waypoint method. Each node has a total event rate of λT. 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 s, there may be only ten packets, or one packet per second.

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 derive the general formula of the overhead reduction ratio using these two parameters. Finally, we determine the optimal values for these two parameters to maximise the overhead reduction, and we show some numerical examples. All these results will be utilised in the next section for the protocol design.

Table 1 lists the symbols, definitions and variables that are used throughout the paper. We will extend our discussion on these assumptions and other possible assumptions in Section 6.

Table 1 Notations used throughout the paper

Src	Source node
I	Intermediate node
D	Destination node
S	Node speed
M	Maximum hops
N	Total number of nodes
P_v	Route caching validation probability
λ	Event rate
T_l	Event lifetime
T_v	Route cache valid time
ORR	Overhead Reduction Ratio

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. As long as a cache is valid, *optimality* can be used to further measure the quality of this cache. However, in this paper, we mainly focus on the study of caching validation P_v since an invalid cache will lead to a new route discovery process and can be seen as much worse than the adverse effects of a non-optimal route.

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 $P_v(S_m, T, L)$ decreases monotonically and continuously as its variables increase. $P_v(S_m, T, L)$ 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 with each other. This is validated by our simulations.

The derivation of the closed form for $P_{lv}(t)$ in a two-dimensional space is non-trivial and we will discuss this later in Section 3.6. In order to not deviate from the main theme, for now, we just take $P_{lv}(t)$ and the corresponding $P_v(S_m, T, L)$ as known functions.

The definition of P_v has several practical meanings. First, it allows the source node to determine the quality of cache it requires before sending the RREQ packet. The source node just needs to determine the validation threshold p_t and appends this value 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 cache to choose after receiving RRES packets. 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. Also, when a new route cache arrives and the route caching table is already full, this new route cache can simply replace the route cache with the lowest P_v.

In brief, the introduction of P_v provides more options for handling route caches more accurately than using cache lifetime alone. The validation threshold p_t mentioned above is a very important parameter for the source node to control the quality of to-be-returned caches and the propagation of RRES packets. In the next section, we will introduce the other parameter *local searching radius*, which is the major parameter used by the source node to control the propagation of the RREQ packets.

3.3 Local searching radius

A local searching scheme must work with a caching scheme to be meaningful. In Cheng and Heinzelman (2003), 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 our study where route caches are available, a local searching scheme becomes much more effective and promising.

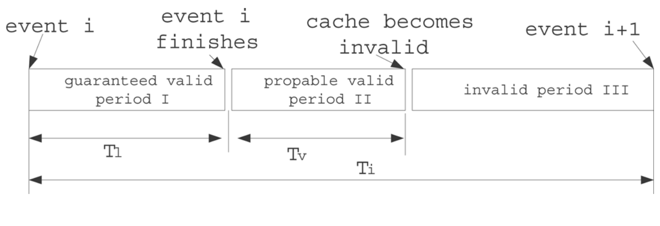
In general, a local search has two-sided effects on the searching overhead. If a local search finds the target itself or it finds cached routes to the target in intermediate nodes, the network-wide flood can be avoided if the route cache is correct, and the searching overhead is reduced effectively. 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. How to determine k and p_t to reduce the routing overhead and enhance the routing performance is the major objective of the rest of our analysis.

3.4 Life periods of a cache

To understand how a node responds to other nodes' RREQs, 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 Figure 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 Figure 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**. If the node receives a RREQ for **D** from another node, it can guarantee to return a valid route. Ideally, during this period, the route cache validation probability P_v for **D** equals 1. However, in practice, the traffic is in the form of discrete packets instead of continuous data flow and there may be time intervals between two consecutive packets. Within these time intervals, P_v is actually less than 1. Also, a RREQ packet may arrive when the route happens to be broken and **Src** is still in the route maintenance phase, although the probability for this case to happen is quite small. Thus, strictly speaking, during period I, a node can respond with a route cache whose P_v is very close to 1.

Figure 1 Time periods for a node's route caching between two events towards the same destination. During the guaranteed valid period I, the node has traffic and maintains the cache. 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 caching valid probability is so low that this route cache may be discarded



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 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 and we consider the cache no longer valid. When a RREQ with the caching validation p_t arrives, this node will not return the route cache. This period lasts until the next event arrives and a new round starts. Also, a node may remove a cache proactively when its P_v is very small. However, since each RREQ may arrive with different values of p_t, correspondingly, T_v may vary for different RREQs as well. A route cache not satisfying one RREQ packet does not necessarily indicate that it does not satisfy other RREQ packets. Therefore, a cache can only be removed when it is guaranteed to be of no future use, i.e., P_v is very small.

When a RREQ arrives at the intermediate node **I** randomly at any time, it may fall into any of the above three periods. The probability that it falls into Period I, in other words, the probability P_I for this intermediate node to return a guaranteed valid route is

$$P_I = \frac{T_l}{T_i}. \tag{2}$$

From a node's view, its total event rate of λ_T is evenly distributed towards the other $N - 1$ nodes. Thus the event rate towards a certain node is $\lambda = (\lambda_T/N - 1) \approx \lambda_T/N$. The average time interval T_i between two events for the same source destination pairs is $T_i = (1/\lambda) = N/\lambda_T$. Thus equation (2) becomes

$$P_I = \frac{T_l \lambda_T}{N}. \tag{3}$$

The probability P_{II} for the RREQ to fall in Period II, that is, for the intermediate node to return a probable valid route is

$$P_{II} = \frac{T_v}{T_i}. \tag{4}$$

As mentioned earlier, T_v in equation (4) can be solved from $P_v(S_m, L, T_v) = p_t$, or $P_{lv}^{L}(S_m T_v) = p_t$.

Note that except p_t, all the other variables in equations (3) and (4) are parameters related to the network pattern or the traffic pattern, and they are not controllable by the protocols. It is the source node's responsibility to determine a good p_t so that there will be route caches returned by intermediate nodes and these route caches are of good qualities.

3.5 Overhead Reduction Ratio (ORR)

The primary goal of this paper is to achieve the minimum routing overhead, i.e., to reduce the number of RREQ and RRES packets as much as possible. We define *ORR* 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

$$ORR = \frac{O_n - O}{O_n}. \tag{5}$$

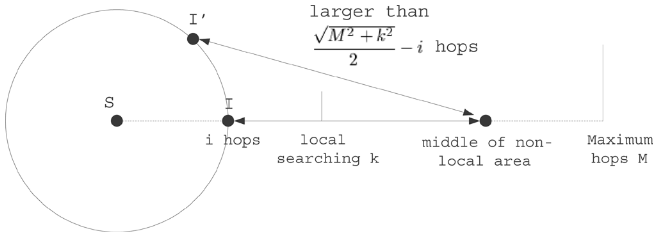
Next, we will derive the overhead *O* as a function of the variables of p_t and *k* and other network and traffic parameters. The analysis of *O* can be briefly described as follows. If the destination is non-local, when a RREQ is flooded locally with radius *k*, it may fall into different life periods of the required route caches. Different caches with different validation probabilities P_v may be returned, and different routing overhead occurs correspondingly. Therefore, the expected overhead can be calculated based on the validation probability of these returned caches.

The propagation of a RREQ packet in a local searching scheme can be illustrated as in Figure 2. The source node **Src** floods a RREQ packet looking for node **D** locally with radius (hops) *k* and route cache validation probability threshold p_t. Each node inside the *k* hops range may return a guaranteed valid route cache with probability P_I. Suppose there are N_i nodes at exactly *i* hops away from node **Src**. The probability P_g for node **Src** to receive at least one guaranteed cache is

$$P_g = 1 - \prod_{i=1}^{k} (1 - P_I)^{N_i}. \tag{6}$$

This equation can be explained as the probability of obtaining no cache at all occurs only when not a single local node has the cache, and by subtracting this value from 1 we find the probability of obtaining at least one cache.

Figure 2 Local searching example. The shortest distance from the intermediate node that is *i* hops away from the source node to the middle of the non-local area is $\sqrt{M^2 + k^2}/2 - i$. *k* is the local searching radius and *M* is the maximum hop of the network



Similarly, each node inside the *k*-hop range may return a probable valid route cache if the RREQ falls into the cache life period II. Thus, the probability of receiving at least one probable valid route cache P_p is

$$P_p = 1 - \prod_{i=1}^{k} (1 - P_{II})^{N_i}. \tag{7}$$

The difference between equations (6) and (7) is that P_I is a constant value for all the nodes, while P_{II} is distinct for each RREQ. In equation (7) we categorise nodes at the same hop-distance from node **Src** for a simpler expression, e.g., nodes **I** and **I'** from Figure 2 are considered to have the same value for P_{II}. Although this categorisation simplifies the expression and the analysis, it may bring error in the final results. We will discuss the error and how to compensate it in Section 4.3.

Suppose there are $N_l(k)$ nodes in the *k*-hop local area, and the total number of nodes in the network is *N*. We can categorise the expected RREQ overhead *O* into the following cases:

- when node **D** is local: cost is local $N_l(k)$
- when node **D** is non-local and a guaranteed valid route cache is found locally: cost is local $N_l(k)$
- 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 $N_l(k)$
- 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 $N_l(k) + N$
- when node **D** is non-local and no cache is found: cost is local plus network-wide $N_l(k) + N$.

Considering the probability for a node to be local as $N_l(k)/N$ and non-local as $1 - (N_l(k)/N)$, we can transform the above descriptions into the equation of the expected routing overhead *O*:

$$\begin{aligned} O &= \frac{N_l(k)}{N} N_l(k) + \left(1 - \frac{N_l(k)}{N}\right) P_g N_l(k) \\ &\quad + \left(1 - \frac{N_l(k)}{N}\right)(1 - P_g) P_p P_v N_l(k) \\ &\quad + \left(1 - \frac{N_l(k)}{N}\right)(1 - P_g) P_p (1 - P_v)(N_l(k) + N) \\ &\quad + \left(1 - \frac{N_l(k)}{N}\right)(1 - P_g)(1 - P_p)(N_l(k) + N) \\ &= N_l(k) + (N - N_l(k))[(1 - P_g)(1 - P_p P_v)]. \end{aligned} \tag{8}$$

Compared to a simple network-wide search without caching whose overhead O_n is the total number of nodes *N*, the *ORR* is

$$\begin{aligned} ORR &= \frac{O_n - O}{O_n} \\ &= \frac{N - N_l(k) - (N - N_l(k))[(1 - P_g)(1 - P_p P_v)]}{N} \\ &= \left(1 - \frac{N_l(k)}{N}\right)(P_g + P_p P_v - P_g P_p P_v). \end{aligned} \tag{9}$$

Equation (9) informs us that the caching scheme saves only when the target is in the non-local area and when a valid route cache is found locally, either from a guaranteed valid route cache or from a probable valid route cache which *is* valid. The last item $P_g P_p P_v$ needs to be deducted since there is a chance that both guaranteed valid caches and probable valid caches are received while only one of them can be chosen.

Let us first look at a scenario where P_g and P_p are much less than 1. This is a typical scenario in which the route cache availability is weak or moderate. Now *ORR* can be further expressed as

$$\begin{aligned} ORR &= \left(1-\frac{N_l(k)}{N}\right)(P_g + P_pP_v - P_gP_pP_v) \\ &\approx \left(1-\frac{N_l(k)}{N}\right)(P_g + P_pP_v) \\ &\approx \left(1-\frac{N_l(k)}{N}\right)(P_g + P_pp_t). \end{aligned} \tag{10}$$

The expressions of P_g and P_p from equations (6) and (7) are too complex to be directly input to equation (10) for analysis. However, with the assumption of P_g and P_p being much less than 1, P_g and P_p can be transformed to

$$\begin{aligned} P_g &\approx 1-\left(1-\sum_{i=1}^{k} N_iP_I\right) = \sum_{i=1}^{k} N_iP_I \\ P_p &\approx 1-\left(1-\sum_{i=1}^{k} N_iP_{II}\right) = \sum_{i=1}^{k} N_iP_{II}. \end{aligned} \tag{11}$$

Now, we have the *ORR* as

$$\begin{aligned} ORR &\approx \left(1-\frac{N_l(k)}{N}\right)(P_g + P_pp_t) \\ &= \left(1-\frac{N_l(k)}{N}\right)\left(\sum_{i=1}^{k} N_iP_I + \sum_{i=1}^{k} N_iP_{II}p_t\right) \\ &= ORR_1(k) + ORR_2(k, p_t). \end{aligned} \tag{12}$$

$ORR_1(k)$ is the overhead reduction from local guaranteed valid caches, and it is only related to the local searching hop number k. $ORR_2(k, p_t)$ is the overhead reduction from local probable valid caches. Besides k, ORR_2 is also related to p_t since an appropriate value of p_t can not only prevent the source node from receiving bad caches by an overly large p_t, but also avoid receiving no cache and achieving no benefits from caching schemes by an overly small p_t. With equation (12), we are able to determine the optimal parameters k and p_t to maximise the overall *ORR*.

3.6 Optimise the parameters: k and p_t

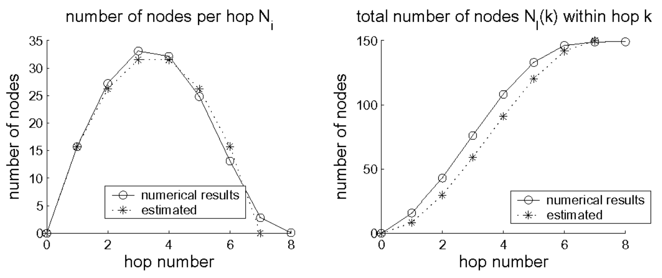
The *ORR* in equation (12) is composed of two items. The first item represents the overhead reduction achieved from the caches in life period I. The second item represents the overhead reduction achieved from the caches in life period II. In this section, we will optimise the parameters k and p_t to maximise *ORR*1 and *ORR*2, and correspondingly, to maximise *ORR*.

First, from statistical results, in a network of limited size, the number of nodes at certain hops N_i and the corresponding number of nodes $N_l(k)$ in the local searching area of radius k can be estimated as

$$\begin{aligned} N_i &\approx \frac{6(Mi-i^2)}{M^3}N \\ N_l(k) &= \sum_{i=1}^{k} N_i = N\frac{6}{M^3}\left(\frac{Mk^2}{2}-\frac{k^3}{3}\right). \end{aligned} \tag{13}$$

where M is the maximum number of hops in the network. Figure 3 shows how close the above two estimations are to the numerical results. In this experiment, 150 nodes are randomly distributed in a unit circle area. Each node has a transmission range of 0.35. The results are based on 30 simulations. As can be seen, the estimated number of nodes in the left plot is close to that of the numerical results. Their major difference is located at the trailer part for hop 7 and 8. At those hops, there may still be some nodes in rare cases. However, these values are so small (less than 5) that we believe our estimation is accurate enough. Also, the estimation for the number of nodes within hop i is accurate enough for our analysis purpose.

Figure 3 The number of nodes at each hop N_i (left) and the number of nodes within k hops (right). Both the experimental results and their estimates are shown



Plugging these values into ORR_1, we have

$$\begin{aligned} ORR_1(k) &= \left(1-\frac{N_l(k)}{N}\right)\sum_{i=1}^{k} N_iP_I \\ &= \left(1-\frac{6((Mk^2/2)-(k^3/3))}{M^3}\right)\frac{6}{M^3}\left(\frac{Mk^2}{2}-\frac{k^3}{3}\right)T_I\lambda_T. \end{aligned} \tag{14}$$

It is easy to determine that ORR_1 reaches its maximum at $k = M/2$. In other words, setting $k = M/2$ will lead to the largest overhead reduction achieved from the guaranteed valid life period I.

The second item $ORR_2(k, p_t)$ becomes

$$\begin{aligned} ORR_2(k,p_t) &= \left(1-\frac{N_l(k)}{N}\right)\sum_{i=1}^{k} N_iP_{II}p_t \\ &= \left(1-\frac{N_l(k)}{N}\right)\sum_{i=1}^{k}\frac{6(Mi-i^2)}{M^3}N\frac{T_v(S_m,L,p_t)}{T_i}p_t. \end{aligned} \tag{15}$$

In the above equation, L and $T_v(S_m, L, p_t)$ are still needed to be clarified. From Figure 2, L is the hop distance from the intermediate node to the destination node. Since the destination node is randomly distributed within the non-local area, we just simply take the middle of the non-local area from hop k to hop M. Since the number of nodes is proportional to the area and thus proportional to the square of the hops, thus we have the middle hop k' that separates the non-local area equally as

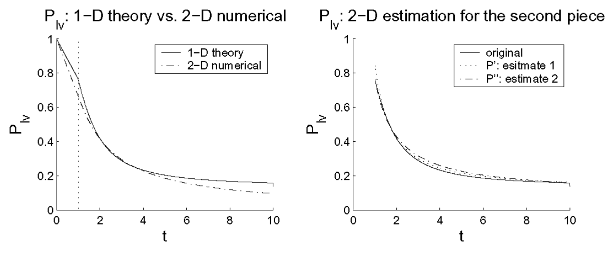
$$\begin{aligned}&(k')^2 - k^2 = M^2 - (k')^2\\ &\Rightarrow k' = \frac{\sqrt{M^2+k^2}}{2}\\ &\Rightarrow L = k' - i = \frac{\sqrt{M^2+k^2}}{2} - i.\end{aligned} \tag{16}$$

L is the average shortest hop distance from the intermediate node located at hop i to this middle point.

Now that S_m, L, and p_t are known, T_v, the time length of the cache period II, can be determined through the function $P_v(S_m, L, T_v) = p_t$. Next, we will determine the expression for P_v to solve for T_v.

In the earlier section when we introduced P_v, we mentioned that $P_v = P_{lv}^L(S_m T)$. Let us restate the definition of $P_{lv}(t)$. A link is formed by two nodes of random distance D uniformly distributed from [0,1]. Suppose that they both have a velocity with a random speed from [0,1] and a random direction from $[0,2\pi]$. If we say that the link is broken when the distance becomes larger than 1, then $P_{lv}(t)$ indicates the probability for the link to remain connected at time i. Since P_{lv} is not linearly related with the caching lifetime t, as seen from Figure 4, our measurement for cache qualities is more accurate than traditional cache timeout schemes.

Figure 4 The $P_{lv}(t)$ in 1-D and 2-D cases. On the left figure, the theoretical results of $P_{lv}(t)$ in a 1-D case and the numerical results in a 2-D case are shown. They both have a two piecewise tendency. On the right figure, two estimations for the 2-D numerical results are shown



We solved the P_{lv} for a one-dimensional case in which the nodes of the link move either towards each other or away from each other. The closed form of this one-dimensional case is

$$P_{lv}(t) = \begin{cases} 1 - \dfrac{t}{3} & t < 1 \\ \dfrac{1}{t} - \dfrac{1}{3t^2} & t \geq 1 \end{cases}. \tag{17}$$

The deduction of the closed form for the two-dimensional case is very difficult since there are two more angle random variables involved. Numerical results show that the 2-D case has a two piecewise tendency, similar to the 1-D results, shown in the left part of Figure 4. When t starts from zero, P_{lv} decreases linearly. After $t = 1$, the decreasing tendency starts to flatten. From this, we conjecture that the closed form of the 2-D case is of the same form as the 1-D case. For the first linear part where $t < 1$, it is easy to derive the estimated form from the numerical results

$$P_{lv}(t) = 1 - 0.2338t \quad 0 < t < 1. \tag{18}$$

As for the second part where $t > 1$, we estimate its form as $(a/t^m) + (b/t^n)$, similar to the form in the 1-D case in equation (17). Two estimated functions P' and P'' are shown in the right part of Figure 4.

$$\begin{aligned}P'_{lv}(t) &= 0.0998t^{0.08} + \frac{0.7518}{t^{1.2400}},\\ P''_{lv}(t) &= 0.0919 + \frac{0.6762}{t}.\end{aligned} \tag{19}$$

The first curve $P'_{lv}(t)$ has a smaller mean square error than $P''_{lv}(t)$. However, we take the second curve $P''_{lv}(t)$ for our later analysis because it has a much simpler expression for analysis and the error is not too large from the numerical results. Note that we do not consider the curve after $t > 10$ because P_{lv} is already lower than 0.2, which indicates a cache of too low quality P_v to be utilised.

Since L and the function of $P_v(S_m, L, T_v)$ have been determined, ORR_2 can be rewritten as a deterministic form

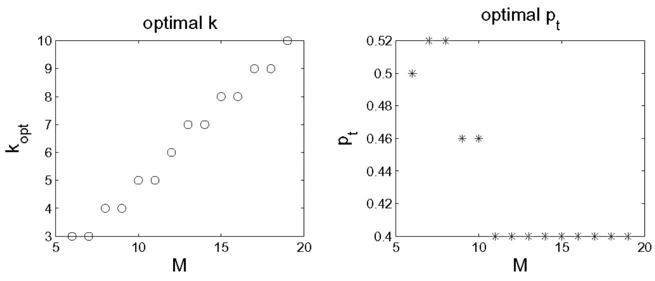
$$\begin{aligned}ORR_2(k, p_t) &= \left(1 - \frac{N_l(k)}{N}\right) \sum_{i=1}^{k} \frac{6(Mi - i^2)}{M^3} N \frac{T_v(S_m, L, p_t) p_t}{T_i}\\ &= \left(1 - \frac{N_l(k)}{N}\right) \sum_{i=1}^{k} \frac{6(Mi - i^2)}{M^3} N \frac{(P'')_{lv}^{-1}\left(p_t^{1/((\sqrt{M^2+k^2}/2)-i)}\right)}{T_i S_m} p_t.\end{aligned} \tag{20}$$

In the above equation, $P''_{lv}(t)$ is

$$P''_{lv}(t) = \begin{cases} 1 - 0.2338t & 0 < t < 1 \\ 0.0919 + \dfrac{0.6762}{t} & t \geq 1 \end{cases}. \tag{21}$$

Although ORR_2 is expressed as a deterministic function with only two variables k and p_t, it is still difficult to find the maximum value of ORR_2 and the corresponding maximum point k and p_t theoretically. Again, we use numerical methods to find the optimal points. The results are shown in Figure 5. As we can see, when the maximum hops $M < 12$, k should be set to $\lfloor M/2 \rfloor$, and when $M \geq 12$, k should be set to $\lfloor (M+1)/2 \rfloor$. For p_t, the optimal value is around 0.4–0.5 depending on the maximum number of hops. Thus, to achieve a maximum overhead reduction from route caching life period I and period II consistently, we choose $k = \lfloor M/2 \rfloor$ and $p_t = 0.4$.

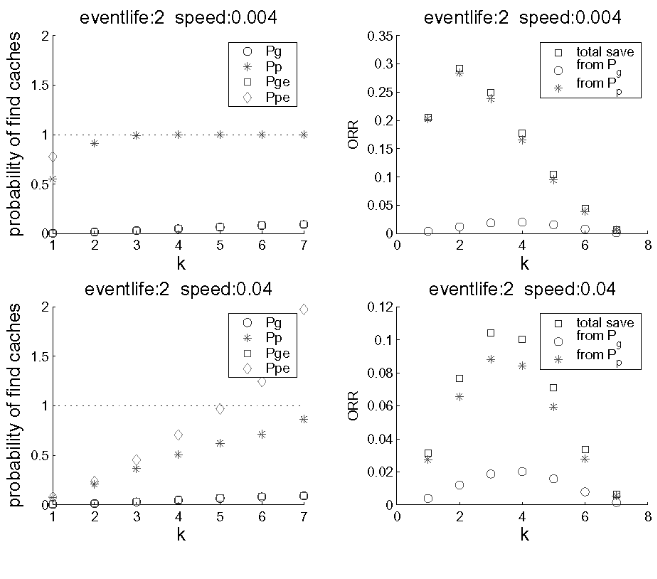
Figure 5 The optimal parameters k and p_t. The optimal k value is consistently around $M/2$. The optimal p_t is around 0.4–0.5, depending on the maximum hop radius M



3.7 Numerical examples

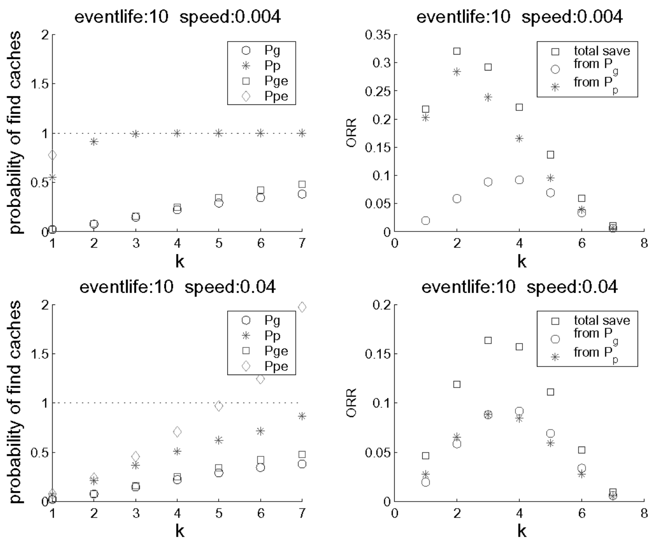
So far, we have found the optimal parameters for p_t and k, which are $p_t = 0.4$ and $k = \lfloor M/2 \rfloor$, to maximise *ORR*. These results are based on the scenarios in which P_g and P_p are much less than 1. Typically, this kind of scenario represents low caching availability at low event rate, fast node speed and low traffic lifetime. In other scenarios where P_g and P_p are comparable to 1, the calculation of *ORR* should be performed by resorting to equations (6) and (7) instead of the approximations in equation (11). In order to have a complete understanding of the relationship between *ORR* and the scenario parameters, we demonstrate some numerical examples in Figures 6 and 7.

Figure 6 The scenarios for event lifetime as low as 2 s. The maximum node speed is either 0.004 (upper part) or 0.04 (lower part). The X-axis indicates the local searching radius k and the Y-axis indicates P_g and P_p and their estimations P_{ge} and P_{pe} in left figures and *ORR* in right figures. The estimation of P_p go beyond 1 at hop 2 and cannot be seen in the upper left plot



In both figures, p_t is fixed at 0.4. The total event rate for each node is 0.05. The total number of nodes is 150 and the estimated maximum hop M for the network is 7. P_g and P_p are shown in the left part of each figure, together with their approximations P_{ge} and P_{pe} calculated from equation (11). *ORR* is shown in the right part of each figure. Figure 6 shows the results for the event lifetime as low as 2 s and the maximum node speed of 0.004 m/s (the upper part) and 0.04 m/s (the lower part). Figure 7 shows the results for the event lifetime as high as 10 s and the maximum node speed of 0.004 m/s (upper part) and 0.04 m/s (lower part). If we map the maximum link distance 1 m into the transmission range of 250 m, the relative speeds of 0.004 m/s and 0.04 m/s can be mapped to the low speed of 1 m/s and the high speed of 10 m/s. Thus, the simulation scenarios to be used in the simulation part correspond to the scenarios here.

Figure 7 The scenarios for event lifetime as high as 10 s. The maximum node speed is either 0.004 m/s (upper part) or 0.04 m/s (lower part). The X-axis indicates the local searching radius k and the Y-axis indicates P_g and P_p and their estimations P_{ge} and P_{pe} in left figures and *ORR* in right figures. The estimation of P_p go beyond 1 at hop 2 and cannot be seen in the upper left plot



As can be seen from the lower part of Figures 6 and 7, when the number of P_g and P_p is less than one, the approximation equations are accurate and *ORR* reaches its maximum when k is near $\lfloor M/2 \rfloor$, the same as the conclusions from the last sections. However, when the node speed is as low as in the upper part of both figures, the approximation from equation (11) is no longer valid for P_p and *ORR* reaches its maximum at $k = 2$. P_p becomes close to 1 and the approximate P_p value using equation (11) becomes an unrealistic value larger than 1. We do not show these unrealistic approximate values of P_p in the upper left part of Figures 6 and 7 once they become larger than 1. There are abundant caches available in this low speed network and it only adds to unnecessary searching cost to apply a large radius. The number of caches from probable valid period II dominates the number of caches from guaranteed valid period I, and the local radius should be adjusted based on the dominating factor. Although we could not determine the optimal k and P_t in a closed form in this abundant caching scenario, we know that k should be adjusted smaller than $\lfloor M/2 \rfloor$ and p_t should be adjusted larger than 0.4 to reduce the overhead. This criteria will be realised in the protocol design in the next section.

Now we have a complete understanding of the relationship between *ORR* and the network parameters. In summary, when caching availability is moderate, the optimal parameters from analysis should be applied to achieve the maximum overhead reduction. When caching is abundant, the local searching radius should be set where a cache is very likely to be found. With these analytical results, in the next section we will design a routing scheme that is able to dynamically adjust itself to approach the maximum performance under all possible scenarios.

4 Protocol description

Although the analysis is involved, the final conclusions are quite simple. Only 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 $k = (M/2)$, assuming weak or moderate caching conditions. When more abundant caching conditions are detected based on history, k should be set to a smaller value according to the dominant factor, such as P_p, the probable valid caching period, shown in the upper part of Figures 6 and 7. Also, p_t can be adjusted larger to reduce unnecessary caches of low qualities.

Therefore, only minor changes are needed for existing on-demand routing protocols to fully attain the benefits of caches. 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 equations (1) and (21).

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, 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 utilise the general statistical method without destination differentiation.

4.2 Protocol procedure

When a source node needs to send a RREQ, it calculates the parameters of 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 $(M/2)$ 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 $k = (\tilde{k} / N_g)$. For example, if the source node uses $\tilde{k} = 4$ to achieve $\tilde{N}_g = 2$ guaranteed valid caches in the history, it should use $k = (4/2) = 2$ to expect to receive only one guaranteed cache this time. p_t is set to a value at 0.9, which indicates only guaranteed valid caches (or almost guaranteed, more strictly speaking) are needed.

In a more general case, the average number of guaranteed valid caches is much lower than 1 and the probable valid caches are the dominating factor such as in the examples shown in Figures 6 and 7. Thus, k should be adjusted towards the goal of $N_p = 1$ by using $k = (2\tilde{k} / N_p)$. 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 $p_t = (\tilde{p}_t / \tilde{N}_p)$. 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 $M/2$, 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 is a little conservative in finding qualified caches in some cases. There may be several reasons for this. First, in our analysis part, we use the shortest hop distance from the intermediate node **I** to the destination **D** for all the intermediate nodes with the same hop distance *i* from the source node. However, most of these intermediate nodes, such as node **I'** in Figure 2, have longer distance and less qualified caches than the shortest hop distance from node **I**. Thus, the results drawn for probable valid caches are overly optimistic in finding route caches. Second, each destination has a different hop distance towards the source node. In our analysis, we simply take the middle point of the non-local area to represent all the non-local destinations. This makes it unfair for some farther destinations and makes the local searching radius not large enough for those destinations. One solution, as mentioned earlier, is to maintain a more detailed statistic for all the possible destinations and take the difference among destinations into account during the parameter adjustments. However, in our implementation, we use a simpler but more aggressive parameter adjustment method by setting $k = (2\tilde{k} / N_p)$ towards the goal of receiving two probable valid caches. This simple modification can provide enough good 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 (http://www.isi.edu/nsnam/ns/ns-documentation). 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. As pointed out by Castaneda and Das (1999), although it is a limitation of the simulation model, it is able to make a very good qualitative evaluation at the packet level without being confined by the specific design of the MAC layer. We believe that realistic MAC will have negligible effect due to the broadcast nature of RREQ packets. And 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 1400 m × 1400 m. Although the region is not of the disk shape as in the analysis part, this square scenario is roughly close to the circular scenario, and it is easier for the setup of the node mobility model. 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 250 m, and the estimated maximum hop value is 7. The maximum node speed of 1 m/s can be mapped to 0.004 m/s for unit transmission range (10 m/s maps to 0.04 m/s), which enables us to compare the simulation results with the analytical results found in Figures 6 and 7 in Section 3.

The total simulation time is either 4500 s or 2000 s, depending on the event rate. The simulation time is chosen longer than 1000 s to remove the potential undesirable effects of starting the random waypoint mobility model (Camp et al., 2002) simultaneously for all the nodes. Also the simulation time is long enough for each pair of nodes to have an event rate larger than 1. For example, if a node has a total event rate of 0.05 events/seconds, it needs 4500 s to have an average of $(0.05/150) \times 4500 = 1.5$ events toward each other node.

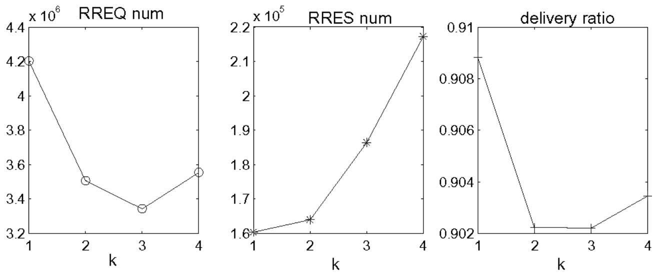
In our simulations, basic metrics commonly used by former studies (Johnson and Maltz, 1996) are investigated, which are routing overhead, successful delivery ratio, discovery latency and route optimality. However, in order to concentrate on our topic of reducing routing overhead, we only show in the 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. We show that our scheme DSR-LC is able to adjust to all the circumstances and shows an ORR up to about 80% compared to DSR-NC and up to about 40% compared to DSR-C, depending on the scenarios studied.

5.2 DSR-LC, effects of *k* and p_t

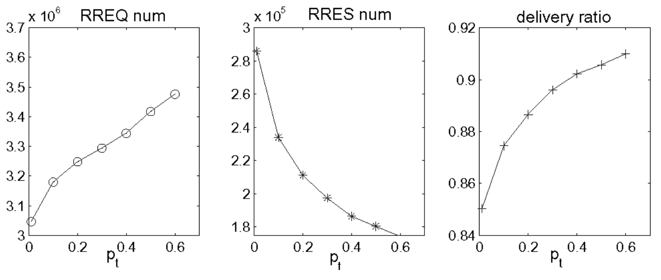
In this part, we will validate the claim that $k = \lfloor M/2 \rfloor$ and $p_t = 0.4$ are optimal values by testing all the possible *k* and p_t values in a scenario with moderate caching availability. First, we fix p_t at 0.4 and change *k* from 1 to 4. From the results shown in Figure 8, we can see that there is an optimal point for the selection of *k*. In the tested scenario, it is $k = 3$, which matches with our analytical result $k = \lfloor M/2 \rfloor = \lfloor 7/2 \rfloor = 3$. Although the number of RRES also increases as *k* increases, this increase is not of the same order as the number of RREQ. In addition, the decrease of the packet delivery ratio at $k = 3$ is small (less than 1%). Another trend which is not shown but worth pointing out is the increasing path optimality with increasing *k* (the difference is also very small, however). With a larger local searching radius, it is more possible to find the target itself. Therefore, the path optimality increases correspondingly.

Figure 8 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



Next we fix *k* at 3 and change p_t from 0 to 0.6. The simulation results are shown in Figure 9. When the threshold p_t is set low, intermediate nodes tend to return a route cache and stop forwarding the query packet. Thus the number of RREQ packets is lower while the number of RRES packets becomes higher. However, the reduction of RREQ packets by lowering p_t may not be desirable because the packet delivery ratio also decreases. With a low p_t, the quality of the routes returned from intermediate nodes tend to be low. It is more possible to fail to deliver packets by using these less valid routes.

Figure 9 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 number of RRES plus the delivery ratio. A good balance point is at $p_t = 0.4$.



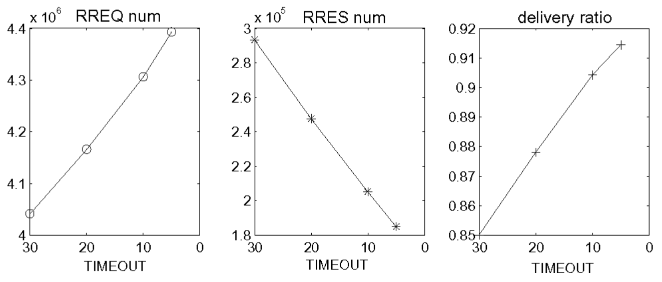
From Figure 9, some point between 0.3 and 0.4 is a good balance point for p_t. Before this point, the increase of p_t leads to an approximately linear increase of RREQ while leading to a *faster* decrease of RRES and a *faster* increase of the packet delivery ratio. The knee of the curves of the RRES 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 provides us a method to trade the routing overhead for the packet delivery ratio by adjusting p_t.

So far, we have validated our analytic results by simulation. In the rest of this section, we will apply DSR-LC with an initial value of $k = 3$ and $p_t = 0.4$.

5.3 DSR-C, effects of timeout

We test DSR-C with the route cache timeout value of 5 s, 10 s, 20 s and 30 s, and the results are shown in Figure 10. In order to compare the results with the selection of p_t in DSR-LC shown in Figure 9, we reverse the order of the cache timeout values on purpose.

Figure 10 Performance of DSR-C with one-hop neighbour searching. The X-axis indicates the timeout value, ranging from 30 s to 5 s. The tradeoff is also between the RREQ number and the delivery ratio. Just like Figure 9, the increase of TIMEOUT causes both metrics to decrease. A good balance point is at TIMEOUT = 10 s



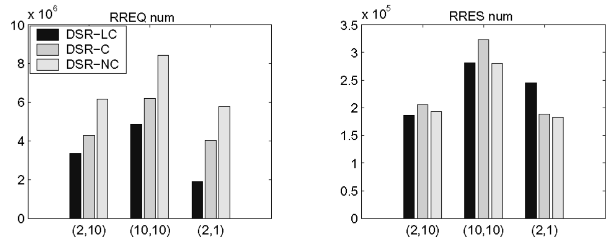
As shown in Figure 10, there is a similar trend and tradeoff for the timeout value as there was for p_t shown in Figure 9. The relationship between the timeout and p_t can be partially explained by equation (1). The larger the time for a route cache to be stale, the easier it is for a RREQ to be satisfied 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 10 s 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 traffic rates, different node speeds, different event lifetimes and different traffic patterns.

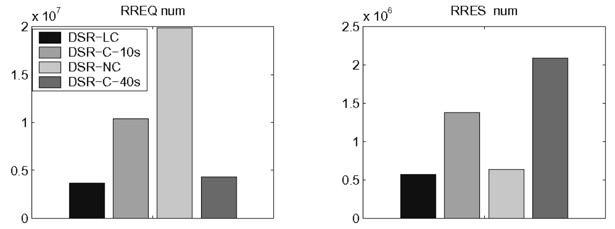
First, we experiment with a low event rate of 0.05 events/second. We test the scenarios with the duplex (event lifetime T_l, maximum node speed S_m) valued at (2 s, 10 m/s), (10 s, 10 m/s) and (2 s, 1 m/s). These scenarios can all be categorised as moderate caching availability. From the results shown in Figure 11, 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.

Figure11 Routing overhead comparisons under a low event rate of 0.05 events/second. The X-axis indicates different scenarios tested, from left to right stands for (event life, node speed) pairs of (2 s, 10 m/s), (10 s, 10 m/s), (2 s, 1 m/s). This figure shows the effects of event lifetime and the node speed on routing overhead in a moderate caching condition



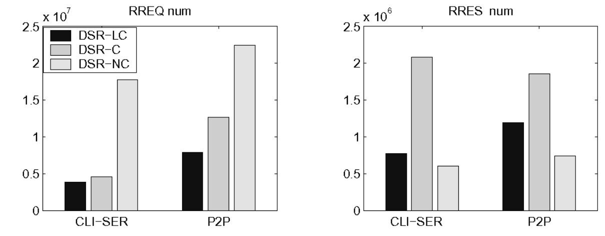
Next, we test an extreme scenario with abundant caching availability. The event rate is as high as 0.5 events/second and the maximum node speed is as low as 1 m/s. As shown in Figure 12, the ORR of DSR-LC in this abundant caching scenario is as high as about 80% compared to DSR-NC, thanks to the caches. 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 s (shown in the fourth column) instead of 10 s. 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 12 Routing overhead comparisons under a high event rate of 0.5 events/second 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 Figure 13 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 ORR of DSR-LC compared to DSR-C but increases the ORR 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 same reason illustrated in the last paragraph, DSR-C has a large number of RRES packets when route caches are abundant.

Figure 13 Routing overhead comparisons for peer-to-peer and client-server traffic models



Another metric that is not shown in the figures but worth mentioning is the number of forwarded data packets. DSR-LC shows up to 10% less data forwarding than DSR-C. This implies that the path chosen in DSR-LC is closer to the optimal path than DSR-C. That is because in DSR-LC, nodes can differentiate the quality of a route cache from the value of P_v and determine whether they should replace the old route cache or not. In DSR-C, only the route cache lifetime is checked, which cannot accurately reflect the real quality of a cached route.

If any of the caching optimisations mentioned in Section 2 are turned on, they just increase the caching availability and the quality of the caches, similar to how the client-server model increases the quality of the caches. Our routing scheme adjusts its parameters in response to the caching availability conditions. That is why in Section 2 we claim that our scheme works fine with all these optimisations.

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.

Two parameters are required during our analysis, the maximum hop of the network M and the maximum node speed. The maximum hop of the network can be estimated according to the network size, and the maximum node speed can be estimated by the target network applications, i.e., applications for pedestrians or for vehicles. Inaccurate estimation of these values may provide erroneous initial values for k and p_t. Also, applications with different mobility patterns may adversely affect the choices of k and p_t. However, as in the client-server model case, our protocol adjustment is mainly based on the caching conditions for moderate and abundant caching conditions. The analytical results only set up an initial value for weak caching conditions. Therefore, erroneous k would not affect the performance much since even the optimal saving is limited in weak caching conditions. The worst results of overly estimating the network diameter M, as may happen for the random way point model, are to flood the network during the startup phase. Afterwards, the searching radius would reduce quickly according to our adjustment rule $k = (\tilde{k} / N_g)$ as caches accumulate. As for p_t, as we can see from Figure 9, using an erroneous p_t value results in trading off the number of RRES packets with the delivery ratio, hence not an adverse factor at all considering the number of RRES packets is not of the same scale as the number of RREQ packets. Also, although we assume that nodes do not pause during the simulations, which leads to a conservative threshold value p_t and more than necessary caches to be returned, p_t will be adjusted to a proper value eventually upon receiving these excess caches.

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. In the current scheme, a node only needs to know the number of RREQ packets and the number of returned RRES packets of its last several attempts. Thus, a little extra storage is needed. If a node keeps more information of the past attempts, such as the destination and the last attempt time, it can predict its next searching radius for the same destination, especially when the destination is a server. Combined with the current scheme, this prediction scheme is expected to provide an even more accurate searching radius and save more routing overhead, especially for high caching availability and the client-server model. 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

Camp, T., Boleng, J. and Davies, V. (2002) 'A survey of mobility models for ad hoc network research', *Wireless Communications and Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, Vol. 2, pp.483–502.

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

Cheng, Z. and Heinzelman, W. (2003) 'Flooding strategy for target discovery in wireless networks', *Proc. 8th International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2003)*, September, San Diego, CA.

Hu, Y-C. and Johnson, D.B. (2000) 'Caching strategies in on-demand routing protocols for wireless networks', *Proc. ACM/IEEE MobiCom*, August, pp.231–242.

Jacquet, P., Multethaler, P., Qayyum, A., Laouiti, A., Viennot, L. and Clausen, T. (2001) *Optimized Link State Routing Protocol*, Internet Draft, Internet Engineering Task Force, March, http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-04.txt.

Johnson, D.B. and Maltz, D.A. (1996) *Mobile Computing, Chapter Dynamic Source Routing in Ad Hoc Wireless Networks*, Imielinski and Korth ed., Kluwer Academic Publishers, pp.153–181

Ko, Y-B. and Vaidya, N.H. (1998) 'Location-Aided Routing (LAR) in mobile ad hoc networks', *ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom'98)*, October, Dallas, Tx.

Krishnamachari, B., Wicker, S., Bejar, R. and Pearlman, M. (2002) 'Critical density thresholds in distributed wireless networks', in Bhargava, H., Poor, H.V., Tarokh, V. and Yoon, S. (Eds.): *Communications, Information and Network Security*, Kluwer Publishers, December, pp.1–15.

Marina, M.K. and Das, S.R. (2001) 'Performance of route caching strategies in dynamic source routing', *Proceedings of the Int'l Workshop on Wireless Networks and Mobile Computing (WNMC) in Conjunction with Int'l Conf. on Distributed Computing Systems (ICDCS)*, Phoenix, AZ, pp.425–432.

Murthy, S. and Garcia-Luna-Aceves, J.J. (1996) 'An efficient routing protocol for wireless networks', *ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks*, October, pp.183–197.

Panchal, N. and Abu-Ghazaleh, N.B. (2004) *Active Route Cache Optimization for On-Demand Ad Hoc Routing Protocols*, ICN 2004.

Park, V.D. and Corson, M.S. (1997) 'A highly adaptive distributed routing algorithm for mobile wireless networks', *Proc. IEEE INFOCOM*, Kobe, Japan, pp.1405–1413.

Perkins, C. and Bhagwat, P. (1994) 'Highly dynamic destination-sequenced Distance-Vector Routing (DSDV) for mobile computers', *Proc. ACM SIGCOMM*, October, London, UK, pp.234–244.

Perkins, C. and Royer, E.M. (1999) 'Ad hoc on-demand distance vector routing', *Proceedings of IEEE WMCSA'99*, February, New Orleans, LA, pp.90–100.

Sucec, J. and Marsic, I. (2001) 'An application of parameter estimation to route discovery by on-demand routing protocols', *Proc. ICDCS 2001*, Phoenix (Mesa), AZ, pp.207–218.

Xue, F. and Kumar, P.R. (2002) *The Number of Neighbors Needed for Connectivity of Wireless Networks*, Manuscript, Available from http://black1.csl.uiuc.edu/prkumar/postscript files.html.

Website

http://www.isi.edu/nsnam/ns/ns-documentation.