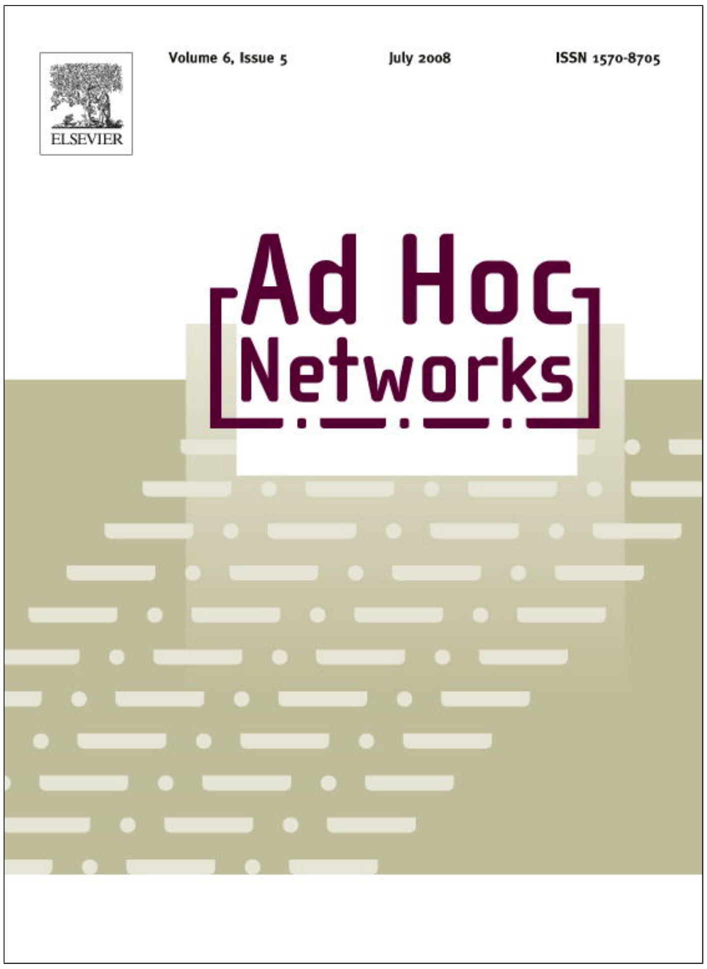
Volume 6, Issue 5
July 2008
ISSN 1570-8705
ELSEVIER
Ad Hoc Networks

Available online at www.sciencedirect.com

ScienceDirect

Ad Hoc Networks 6 (2008) 661–674

www.elsevier.com/locate/adhoc

Discovering long lifetime routes in mobile ad hoc networks

Zhao Cheng *, Wendi B. Heinzelman

Department of Electrical and Computer Engineering, University of Rochester, 328 Hopeman, Box 270126, Rochester, NY 14620, United States

Received 8 June 2006; received in revised form 27 March 2007; accepted 11 June 2007
Available online 16 June 2007

Abstract

In mobile ad hoc networks, node mobility causes frequent link failures, thus invalidating the routes containing those links. Once a link is detected broken, an alternate route has to be discovered, incurring extra route discovery overhead and packet latency. The traffic is also interrupted at the transport layer, and proper traffic recovery schemes have to be applied. To reduce the frequency of costly route re-discovery procedures and to maintain continuous traffic flow for reliable transport layer protocols, we suggest discovering long lifetime routes (LLR). In this paper, we first propose g-LLR, a global LLR discovery algorithm, that discovers LLRs of different route lengths for any given pair of nodes. We then propose a distributed LLR discovery scheme (d-LLR) that discovers two of the most desirable LLRs through one best-effort route discovery procedure. Simulations show that the lifetimes of the routes discovered by d-LLR are very close to those discovered by g-LLR. Simulations also show that the performance of different transport layer protocols is greatly improved by using LLRs. More importantly, traffic can remain continuous using the provided LLRs. D-LLR can be implemented as an extension to existing ad hoc routing protocols, and it improves the performance of transport layer protocols without modifications on them.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Long lifetime routes; Mobile and wireless ad hoc networks; Routing protocols; Cross-layer interactions; Performance analysis and simulation of protocols

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. On-demand routing protocols based on shortest path algorithms, such as DSR [1], AODV [2] and TORA [3], are usually applied due to their relatively lower routing overhead compared with that of table-driven protocols. Node mobility is the major factor that affects the performance of these routing protocols. Since a link break from node mobility invalidates all the routes containing this link, alternate routes have to be discovered once the link is detected as broken. This new discovery phase incurs network-wide flooding of routing requests and extended delay for packet delivery. Furthermore, the upper transport layer may mistake this temporary route break as long term congestion and

* Corresponding author. Tel.: +1 585 2758078.
E-mail address: zhcheng@ece.rochester.edu (Z. Cheng).

1570-8705/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2007.06.001

execute unnecessary backoffs. Since ad hoc routing protocols usually have their own retransmission scheme for route discovery, failure of synchronization between the routing and transport layers often occurs, resulting in poor overall performance.

Discovering long lifetime routes (LLRs) can reduce the impact of node mobility and improve the overall performance compared to using randomly chosen shortest-path routes. When a route with a longer lifetime is chosen, less frequent route discovery, which usually involves expensive network-wide flooding, is required, thus less routing overhead is incurred. The impact of long lifetime routes on upper layer protocols is also obvious. First, an LLR can reduce the chance of a route break, thus reducing the chance for abnormal TCP transmission behaviors observed in [4]. If two LLRs can be provided at a time, the routing protocol can save the longer LRR as a backup and use the shorter LRR, which usually has a shorter route length and is thus more energy-efficient, as the primary route for transmissions. The routing protocol can switch to the longer LRR to maintain the flow of traffic when the shorter LLR breaks. Meanwhile, a new route discovery procedure can be initiated, and when the newly discovered LLRs are returned, the old LLRs can be replaced.

A preliminary study on the effectiveness of LLRs has been performed in [5]. Among the many possible routes between a given pair of nodes, a route with the longest lifetime can be found at different route lengths. The trend shows that the lifetime of these LLRs increases almost linearly with the route length. Although an LLR with a long route length reduces the frequency of route breaks, it is inherently inefficient since a longer route length implies more packet forwarding for each packet. The benefit from less frequent route discovery can be easily diminished by the increased number of hops to forward data to the destination. It is suggested in [5] that only LLRs with short route lengths are desirable since they can reduce route break frequency and there is no sacrifice on packet delivery efficiency from using more hops.

In this paper, we first present a global LLR algorithm (g-LLR) that discovers the longest lifetime route at each different route length for a given pair of nodes. This algorithm requires global knowledge and provides the optimal LLRs for analysis. We then propose a distributed long lifetime route (d-LLR) discovery approach that finds two LLRs, termed as the primary LLR and the auxiliary LLR, in one best-effort discovery procedure. The primary LLR is the LLR at the shortest route length, and the auxiliary LLR is the LLR that contains one more hop than the shortest route. Simulations show that these two LLRs are very similar with the LLRs discovered using g-LLR and greatly improve the overall routing performance. Based on these two LLRs, we also propose a fast-switch scheme that maintains continuous traffic flow for upper transport layers. This is crucial for reliable transport layer protocols and stream-based applications where the interruption of traffic may cause abnormal behaviors and deteriorate the overall performance.

The rest of this paper is organized as follows. Section 2 provides an overview of previous efforts in discovering stable routes and stresses the design lessons learned from the previous long lifetime route studies. Section 3 illustrates how g-LLR obtains the LLRs at different route lengths and how d-LLR achieves the goal of finding desirable LLRs in a distributed manner. Section 4 evaluates the performance of LLR by comparing with DSR. Various transport layers are tested. Section 5 concludes the paper.

2. Overview and related work

Shortest-path routing is the most common algorithm in existing ad hoc routing protocols [1,2]. However, as pointed out by De Couto et al. [6], shortest-path routing is not good in terms of link stability, even for static multi-hop wireless networks. In mobile ad hoc networks, links are even more fragile due to node mobility. A good metric to enable adaptive routing protocols, as suggested by Boleng et al. [7], is link duration, or as termed in this paper, link lifetime.

Several methods have been proposed for link lifetime estimation in different scenarios. In cellular systems, signal strength provides hints for node mobility patterns and probable connection loss. If nodes are equipped with GPS, link lifetime can be calculated from node distance and node speed. A theoretical link lifetime prediction method is proposed in [8]. This prediction method uses the current link age to predict the residual lifetime. The lifetime distribution for various mobility patterns, which can be used for link lifetime prediction, is achieved through experiments [9]. In our study, we do not intend to repeat these research topics and invent new methods for link lifetime estimation. Instead, we will propose a route discovery method built on existing link lifetime estimation methods to discover routes with long lifetimes in a distributed manner.

The idea of finding a ‘‘good route’’ rather than a random route is not new. Several routing protocols based on link stability have been proposed, such as ABR [10] and SSA [11]. They both determine a link to be good if it has existed longer than a certain period. The common idea behind these approaches is to prefer stable links or strongly connected links rather than transient links during route setup. However, these protocols overly stress on link qualities and neglect the fact that a route quality is restricted by the quality of its weakest link. Discovering routes with good quality is more difficult than discovering good quality links due to the difficulty in quantifying route quality and the lack of global information for discovery.

Many protocols have been proposed and studied to improve routing performance. Some protocols attempts to use disjointed multiple paths to improve the overall throughput [12]. Some other protocols attempts to reduce the overhead from route discovery by optimizing and reusing the route caches within the network [13–15]. Multiple route cache storage is often used as well to reduce the route discovery latency for faster route recovery. The performance of transport layer protocols (especially TCP) with ad hoc routing protocols has also been researched [4,16–18]. Bad performance was observed and cross-layer designs have been proposed to improve the performance [17,19,20]. The basic idea is to let TCP obtain network conditions from the routing protocol, thus adjusting its behavior accordingly.

During our research, we learned several lessons for the above approaches. First, route caches should be used very cautiously. Route caches in relay nodes may be inefficient or obsolete. Considering the large number of relay nodes, many misleading route replies may be returned on one route query. Route caches located in the source node may also be misleading. When one path breaks, it is likely that the other paths also break. However, the source node cannot recognize the obsolete caches until they have tried them. Second, multi-path is difficult to apply since these paths are very likely to interfere with each other. To avoid interference by using disjointed paths, routes with more hops have to be used, which adds up to the data routing overhead. Finally, cross-layer design, if necessary, should be limited at the bottom layers where it is usually implemented as hardware/firmware and full control is easy to obtain. Cross-layer designs through many layers bring about layered overhead, and sometimes is unlikely to implement, especially when the upper layer protocols have already been standardized.

Following these guidelines, we propose a routing protocol d-LLR that only attempts to discover LLRs with short route lengths. After one discovery procedure, two routes will be found. The first route is the one which, among all shortest-path routes, has the longest lifetime. The second route is one hop longer than the shortest path but has an even longer route lifetime.

Our protocol is designed to focus more on the performance of data routing. It uses the shortest path with the longest lifetime for data routing most of the time, thus saving energy on both hop forwarding and route recovery. It does not use excessive route caches to avoid unnecessary storage and trial failures. It maintains two routes with increasing route lifetimes to maintain continuous flow of traffic. It uses existing link lifetime estimation schemes, and the functionality is transparent to the upper transport layers. By avoiding exotic route discovery, complex route caching schemes and interactions with transport layers, our protocol is easy to implement and provides a direct data routing performance improvement.

3. Long lifetime route discovery: g-LLR and d-LLR

In this section, we first present g-LLR, a global LLR algorithm that discovers all the LLRs for a given pair of nodes. The key idea of g-LLR is to pick the link with the longest lifetime and add it to the initially empty network. The algorithm notes the changes of route length and route lifetime between the given pair of nodes until every link has been added to the network. The LLR at each route length can thus be recorded. This algorithm requires global knowledge of all the link lifetimes, and thus it is not practical to implement directly in routing protocols. However, the optimal LLRs obtained from this global algorithm can be used as a benchmark to evaluate the performance of other distributed LLR approaches.

We then propose d-LLR, a distributed LLR approach, that discovers LLRs of short route lengths in a distributed manner. D-LLR provides best-effort discovery of a primary LLR and an auxiliary LLR in one discovery procedure. The primary LLR is the LLR with the shortest route length, and the auxiliary LLR is one-hop longer than the primary LLR. D-LLR applies the lessons from g-LLR and only attempts to discover LLRs with short route lengths. As mentioned earlier, only short LLRs can provide both energy efficiency for packet delivery and routing overhead reduction for route discovery. Lifetime extension from using LLRs with long route lengths cannot compensate for the adverse effects from the excessive hops of packet forwarding. Thus, LLRs with long route lengths are not preferred.

G-LLR is a global algorithm that picks the link with the longest link lifetime from a global view. D-LLR, on the contrary, is a distributed protocol with only local information. In this protocol, each individual node waits for a proper time before it forwards a route request so that each forwarded route request contains the LLRs seen from each individual node's point of view. In other words, when a node rebroadcasts the LLR request packet (LLR-REQ), the routes contained in the request packet indicate the LLRs from the source node to itself. In this way, when the LLR-REQ reaches the destination, the destination simply chooses the best LLRs from all the received route requests and returns them in one LLR-RES packet. To achieve this, a lifetime-aware delay scheme that associates a forwarding delay with the current primary lifetime is implemented.

3.1. A review of link lifetime estimation

From the literature, there are two general methods to quantify the quality of a link using link lifetime. The first method expresses link lifetime in a stochastic manner. A link break probability $P(t)$ indicates the probability that a link is broken at time t. An example of $P(t)$ in a Gauss-Markov scenario can be found in [8]. $P(t)$ is a non-decreasing function starting from 0. Obviously, as time elapses, the probability that a link will break increases.

The second method expresses link lifetime in a deterministic manner. Link lifetime can be estimated through the link break probability given an estimation rule, such as from now to when the link break probability is higher than a certain threshold. The quality of a link can be thus quantified using this estimated link lifetime. Link lifetime can also be calculated using node location and movement estimated from signal strength or GPS. For practical protocol designs, such quantifications are necessary since it is much easier to append a value into a route message than to append an entire probability function.

Correspondingly, route lifetime can also be expressed in both manners. Suppose a route is composed of n links. Using link lifetime probability, the route lifetime distribution $P_r(t)$ can be calculated as

$$P_r(t) = 1 - \prod_{i=1}^{n}(1 - P_i(t)) \tag{1}$$

$P_i(t)$ indicates the probability for link i to be broken at time t. On the contrary, $1 - P_i(t)$ indicates the likelihood for link i to be valid at t. The probability for all the n links to be valid at time t is $\prod_{i=1}^{n}(1 - P_i(t))$, and the probability for the route to be broken at t is one minus this value. On the other hand, using quantified link lifetime estimations, the route lifetime l_r is simply the minimum lifetime of the n links.

$$l_r = \min\{l_1, l_2, \ldots, l_n\} \tag{2}$$

D-LLR determines the route query forwarding delay based on the quantification of route lifetimes. To focus on the study of LLR, we assume that lifetime has been estimated in a quantified manner, and it can be directly appended to routing messages as an additional field.

3.2. G-LLR: global LLR algorithm

The global LLR algorithm discovers the LLR at different route lengths for a given pair of nodes in a given network. The basic idea of g-LLR is to add the link with the longest link lifetime, then adjust the route length between each pair of nodes. Once the route length between two nodes changes, the new route is the LLR at the new route length, and the last added link lifetime is the route lifetime of this LLR. This step continues until all the links have been added to the network. Eventually, we have the LLRs at all the different route lengths.

Suppose we are interested in investigating the LLRs between the source node **S** and the destination node **D**. The arc set A is sorted in descending order by the lifetime $c[i,j]$ of the link composed of nodes i and j. We denote an edge as e or a link between node i and j as $e[i,j]$ if node i and j are connected. $d[i,j]$ is the hop distance between nodes i and j. d_{prev} is the last route length recorded between the pair. The g-LLR algorithm is shown in Algorithm 1 (see [5] for more details).

This algorithm is very similar to the Floyd–Warshall algorithm for solving the all-pairs shortest path problem [21]. The difference is in the outlet loop, in which we choose an edge, while the Floyd–Warshall algorithm chooses a vertex. Therefore, our g-LLR algorithm has a complexity of O(n^4), which is more than that of the Floyd–Warshall's O(n^3). This is because the number of the edges in a dense network is of O(n^2). However, with the additional complexity, we are able to obtain one more degree of route lifetime information. That is, we are able to obtain the maximum route lifetime at each route length for any given pair of nodes rather than only the shortest-path route. Besides, the increase of complexity is limited, and the algorithm is still of polynomial complexity.

Algorithm 1. G-LLR algorithm.

```
Data: $A$, initial $c[i,j]$ for each link
Result: Record of the longest lifetime achievable for
        routes with different hop distances $\{d[\mathbf{S,D}], c[\mathbf{S,D}]\}$
begin
    $S := \emptyset; \overline{S} := A; d_{prev} = \infty;$
    for all node pairs $[i,j] \in N \times N$ do
        $d[i,j] := \infty; pred[i,j] := 0;$
    end
    for all nodes $i \in N$ do $d[i,i] := 0;$
    while $|S| \neq A$ do
        let $e[i,j] \in \overline{S}$ for which
        $c[i,j] = \max\{c(e), e \in \overline{S}\};$
        $S := S \bigcup \{[i,j]\}; \overline{S} := \overline{S} - \{[i,j]\};$
        $d[i,j] = d[j,i] = 1;$
        for each $[m,n] \in N \times N$ do
            if $d[m,n] > d[m,i] + d[i,j] + d[j,n]$ then
                $d[m,n] := d[m,i] + d[i,j] + d[j,n]$ and
                $pred[m,n] := i;$
            end
            if $d[m,n] > d[m,j] + d[j,i] + d[i,n]$ then
                $d[m,n] := d[m,j] + d[j,i] + d[j,n]$ and
                $pred[m,n] := j;$
            end
        end
        if $d[S,D] < d_{prev}$ then
            $d_{prev} = d[\mathbf{S,D}]$ and record $\{d[\mathbf{S,D}], c[\mathbf{S,D}]\}$
        end
    end
end
```

We will only briefly discuss the proof of this algorithm since the proof is essentially the same as that of the Floyd–Warshall algorithm. First, when a new edge is added to the network, the hop distance of the shortest path between any pair of nodes will either decrease or remain the same. Second, since the new edge/link has a less weight than the previous edge/link, it will not increase the route lifetime if the route length remains the same; and it will be the first and the maximum route lifetime if the route length decreases.

Simulation results show that the lifetime of LLRs increases linearly with the route length of LLRs for non-stop random moving patterns [5]. A similar trend is also discovered for the commonly used random waypoint mobility model with no pause time. Therefore, there is a certain tradeoff on whether to choose an LLR with short route lengths or to choose an LLR with long lifetime but longer route lengths. On one hand, an LLR with a short route length can deliver packets using fewer hops, thus reducing the packet delivery overhead. On the other hand, an LLR with a short route length also has a shorter route lifetime and breaks faster than longer LLRs, thus increasing the routing overhead from route discovery. Depending on traffic density and node mobility, LLRs with different route lengths should be chosen correspondingly. When traffic is heavy and node mobility is low, packet delivery overhead becomes the dominating factor and LLRs with short route lengths should be used. When traffic is light and node mobility is high, route discovery overhead becomes dominant and LLRs with long lifetimes should be chosen.

For our distributed LLR design, we only attempt to discover LLRs with short route lengths. This is for two reasons. First, it is difficult to obtain LLRs with long route lengths in a distributed manner due to lack of global knowledge. Second, LLRs with long route lengths may outperform LLRs with short route lengths only in network scenarios with very light traffic and very high node mobility. Therefore, LLRs with short route lengths are more suitable for the majority of practical applications. Finally, errors during link lifetime estimation will eventually be reflected by route lifetime errors. The longer a route is, the more likely the route lifetime is shorter than expected. Therefore, for our distributed LLR approach, we are only interested in LLRs with short route lengths.

3.3. D-LLR: distributed LLR protocol

D-LLR can be used as an extension to most ad hoc routing protocols with minor modifications. D-LLR achieves two LLRs in a best-effort query procedure by observing the current LLR lifetime and determining the request forwarding delay accordingly. The main procedure of d-LLR is similar to a typical on-demand routing protocol such as DSR: broadcast a route request from the source node and unicast a route reply message back from the destination node. The difference lies in the implementation details such as routing packet format, routing update rules and LLR-REQ forwarding delay rules.

3.3.1. General procedures

In d-LLR, LLR-REQ contains a primary route that expands while the LLR-REQ propagates throughout the network, similar to that in DSR. In addition, it contains an auxiliary route, which does not have any impact on LLR-REQ forwarding decisions. Also included in the LLR-REQ are the primary and auxiliary route lifetimes. These route lifetimes are calculated at each intermediate node by choosing the minimum from the composed estimated link lifetimes. Finally, an extra field that specifies propagation duration is included in the LLR-REQ. This propagation duration indicates the time since the LLR-REQ packet was transmitted by the source node, and it is used by intermediate nodes to calculate the local delay time for LLR-REQ forwarding.

The procedures of d-LLR are illustrated as follows. We focus on the differences with the procedures of DSR:

1. The source node broadcasts an LLR-REQ packet, which contains two routes: the primary LLR and the auxiliary LLR, with their respective lifetimes. The primary LLR is the LLR with the shortest route length, while the auxiliary LLR is the LLR that is one hop longer than the primary LLR. Initially, these routes only contain the source node and their lifetimes are set as 0.
2. When an intermediate node receives an LLR-REQ for the first time, it appends itself into the prim/aux routes in the packet and records the request locally. Then it adjusts the lifetimes by choosing the minimum of the previous route lifetime and its link lifetime with the previous node. Next, it schedules a local delay time for forwarding this modified LLR-REQ based on certain delay rules (see below). When the delay time is up, it forwards this LLR-REQ packet.
   If the intermediate node receives a duplicate LLR-REQ, it will update the prim/aux routes in its recorded LLR-REQ based on the LLR update rule (see below). Meanwhile, it reschedules the delay time if a better route is found and a shorter delay should be applied. The update rule and the delay rule will be explained in detail later. In brief, the delay rule requires routes with longer primary lifetime to have shorter delay, and the LLR update rule requires that the auxiliary route is longer than the primary route in both route length and route lifetime, and that the route length should be one-hop longer than the primary LLR.
3. The destination uses the same LLR update rules when receiving LLR-REQs from different paths. However, it simply waits enough time and then it will unicast an LLR-REP to the source node using the primary route with the auxiliary route attached, just as in the normal DSR route response procedure.

3.3.2. LLR update rules

A node compares the routes in its current record with those in the newly arrived LLR-REQ packets. There are four routes involved in making the decision: the primary and auxiliary LLR in the local node's record and those in the newly arrived LLR-REQ. In brief, the node picks the one with the longest lifetime from the shortest routes as the primary route, and it picks the one with the longest lifetime from the second shortest routes as the auxiliary route.

In more detail, there are several cases. If an LLR-REQ arrives with a shorter primary route than the recorded primary route, the new primary route will be recorded as the primary route. The auxiliary route will be chosen from the recorded primary route and the newly arrived auxiliary route. If an LLR-REQ arrives with a primary route of the same route length as the recorded primary route, the primary route will be chosen from these two routes, and the auxiliary route will be chosen between the recorded and newly arrived auxiliary routes. If an LLR-REQ arrives with a longer primary route, only the auxiliary route will be chosen between the recorded auxiliary route and the newly arrived primary route. More details can be found in Algorithm 2.

Algorithm 2. LLR update rules at intermediate nodes.

```
Data: $pT_r$, recorded primary lifetime;
      $aT_r$, recorded auxiliary lifetime;
      $pT_n$, newly arrived primary lifetime;
      $aT_n$, newly arrived auxiliary lifetime;
      $pL_r$, recorded primary route length;
      $aL_r$, recorded auxiliary route length;
      $pL_n$, newly arrived primary route length;
      $aL_n$, newly arrived auxiliary route length;
begin
    if $pL_n < pL_r$ then
        $aL_r = \min\{pL_r, aL_n\}$ and adjust $aT_r$ accordingly;
        $pL_r = pL_n$ and $pT_r = pT_n$;
    else if $pL_n = pL_r$ then
        $aL_r = \min\{aL_r, aL_n\}$ and adjust $aT_r$ accordingly;
        $pT_r = \max\{pT_n, pT_r\}$;
    else
        $aL_r = \min\{aL_r, pL_n\}$ and adjust $aT_r$ accordingly;
    end
end
```

3.3.3. Delay setup rules

To find both the primary and auxiliary routes in one discovery procedure, two key rules have to be observed when setting up the LLR-REQ forwarding delay. First, intermediate nodes with a longer primary route lifetime should forward earlier so that neighboring nodes will have a better chance of incorporating this route in their auxiliary route before they do their forwarding. Second, nodes at the same hop distance to the source node should forward at the same pace to reduce the chance of missing a primary LLR by forwarding the LLR-REQ too early. We illustrate how both rules help set up the primary and the auxiliary routes using Fig. 1 as an example.

In Fig. 1, the number on each link indicates the link lifetime. After node **a** sends out the LLR-REQ, the ideal scenario is that when node **c** forwards the LLR-REQ, the LLR-REQ packet should contain the quadplex

⟨prim route, prim lifetime, aux route, aux lifetime⟩ as ⟨[**a**, **b**, **c**], 3, [**a**, **d**, **e**, **c**], 4⟩. If the first rule is violated by forwarding earlier for shorter lifetime routes, then node **c** will broadcast the LLR-REQ with [**a**, **b**, **c**] before receiving the LLR-REQ from node **e**, thus missing the auxiliary route. If the second rule is violated by forwarding longer lifetime routes too fast, then node **c** may forward [**a**, **d**, **e**, **c**] before receiving the LLR-REQ from node **b**, thus missing the primary route.

The delay function also needs to avoid potential packet collisions from the MAC layer. Neighboring nodes are likely to determine their rebroadcast time based on the same primary lifetime value, especially when the primary route lifetime is determined by the earlier shortest link lifetime. In Fig. 2, nodes **b** and **c** receive the same primary route lifetime 3 from node **a** simultaneously. To avoid node **b** and **c** choosing the same delay time and colliding in their transmissions to node **d**, jittering should be introduced in the delay function.

In our design, a node chooses its overall delay t_d based on the following function:

$$t_d = f(l; D_1) + D_2 \times (h - 1) + \text{Uniform}(D_3) \qquad (3)$$

The first item follows the first delay rule, where l is the primary route lifetime and D_1 is the delay parameter for rule 1. Function f is a monotonic non-increasing function of link lifetime that determines the delay from 0 to D_1 based on the primary route lifetime. The second item indicates that nodes at the same hop distance h to the source node should broadcast approximately at the same time. D_2 is the second delay parameter, and it should be larger than D_1 so that nodes will not forward out of pace by the delay from D_1. The last item is the jittering to avoid collisions. D_3 is the third delay parameter, and it should be much smaller than D_1 so that it is unlikely to alter the rule that longer lifetime routes lead to shorter delay.

Notice that instead of local delay, t_d in Eq. (3) is the overall delay from when the source node starts

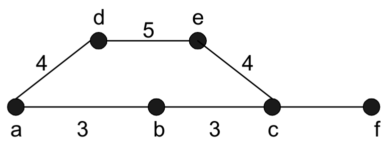

Fig. 1. An example showing how the delay rules can help set up both the primary and the auxiliary routes.

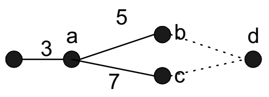

Fig. 2. An example showing a potential collision using the same primary lifetime. Nodes **b** and **c** may collide when using primary lifetime 3 to calculate their forwarding delay.

the LLR-REQ to when the current node forwards the packet. Therefore, when each node forwards the packet, it should attach the current propagation duration in the LLR-REQ packet. With this information, the next node is able to calculate the local delay time by subtracting the propagation duration from the overall delay t_d. We do not include the initial LLR-REQ time in the LLR-REQ because this would require global clock synchronization among all the nodes in the network, which is difficult to implement.

3.4. Other design considerations

There are some design specific issues remaining to be discussed. First, what should be the values for the delay parameters D_1, D_2 and D_3, and what should be the delay function *f*? By choosing a larger value of D_1, we are able to better differentiate routes with different lifetimes. However, since D_2 has to be larger than D_1, the overall delay for the route discovery will increase correspondingly. Although choosing a smaller D_1 may lead to smaller discovery delay, an even smaller D_3 has to be chosen, which may increase the chance of collisions and missing some important routes. Thus, proper parameters have to be chosen.

For the delay function *f*, a simple choice is to associate the delay linearly decreasing with the link lifetime. However, since we are able to determine the route lifetime distribution through either statistical results or analytical results [5], we could make *f* biased on the most possible occurring lifetime spans instead of spreading evenly.

We tested different sets of parameters by ranging D_1 from 0.01 to 0.1 s, D_2 from 0.1 to 1 s, and $D_3 = 0.01D_1$. We also tested two delay functions: a simple linear function as shown in the left plot of Fig. 3, and a biased two-piece linear function as shown in the right plot of Fig. 3. We noticed that lifetime performance is not much affected by various parameter choices. The lifetime difference is within 2 s. Considering the fact that link lifetime estimation is erroneous in nature, this minor lifetime performance difference can be easily dominated by the error. Therefore, by default, we use $D_1 = 0.05$s, $D_2 = 0.1$s, $D_3 = 0.0005$s, and a linear function with a cutoff time L_t at 1000 s throughout the entire paper.

A priority queue, similar to the one used in DSR, is utilized in our design. The priority queue is a necessity for LLR, especially when there is data traffic. Without the priority queue, delay from the data in the queue will add up to the local delay of the LLR-REQ, which invalidates the delay rule that a longer lifetime route leads to shorter delay.

4. Performance evaluation

In this section, we evaluate the performance of LLR through several groups of experiments. The first group of experiments focuses on evaluating the route lifetimes discovered using d-LLR with those discovered using g-LLR and those discovered using DSR. This informs us how close to optimal our distributed LLR scheme performs compared to a global algorithm, and how much lifetime improvement we can obtain. The second group of experiments compares the general routing performance of d-LLR with that of DSR using UDP as the transport layer protocol. Since UDP is a semi-transparent transport protocol, we are able to demonstrate the direct advantage of LLR over random routes. The third group of experiments uses a more widely used transport layer protocol, TCP, and reinvestigates the performance of LLR. The last group of experiments tests the robustness and effectiveness of d-LLR by introducing errors to link lifetime estimations.

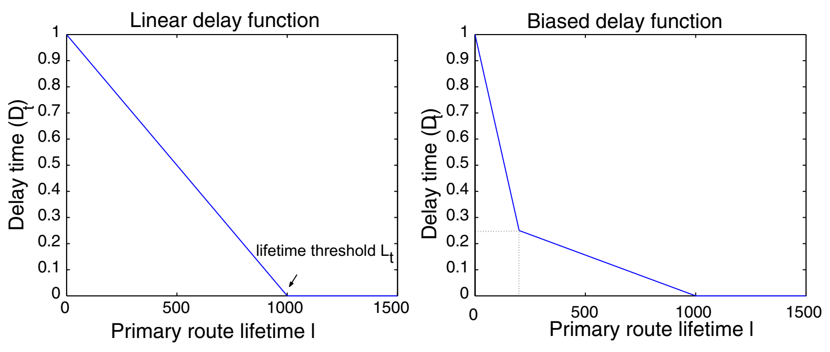

Fig. 3. The delay function $f(l;D_1)$. On the left is a simple linear function. On the right is a biased linear function to concentrate on the first 200 s.

4.1. Lifetime performance

First, we compare the route lifetimes found using g-LLR, d-LLR and DSR. Our simulations are done using NS-2 [22]. We look at a fully connected network by placing 100 nodes uniformly inside a network with a radius of 500 m. The transmission range of each node is 250 m. Nodes move with a speed uniformly distributed from [0, 10 m/s], according to the random waypoint model with no pause time. 802.11b is used as the MAC layer protocol and the wireless channel bandwidth is 2Mbps. We group the 100 nodes into 50 pairs and observe their initial route lifetime at time 0. We experiment using 10 different scenarios and average the results. Link lifetimes are calculated offline and input into each node based on node mobility pattern before the simulation.

We measure the route discovery success ratio and route lifetime for g-LLR, d-LLR and DSR. Route discovery success ratio (RDSR) indicates the likelihood that a route can be discovered when there exists one. Both d-LLR and DSR may fail to discover a route even if there exist routes between the observed pair of nodes. This is due to the potential collisions among flooded route request messages. Route lifetime is another metric we observe. For DSR, we examine the route lifetime of the returned random route. For LLR, the lifetimes to be examined are the primary lifetime (PL) and the auxiliary lifetime (AL).

The results are shown in Table 1. We found that for a total of 500 cases tested, DSR can successfully discover an initial route in one discovery attempt for only about 67% of the time. On the other hand, d-LLR discovers a route in one discovery attempt for 498 out of 500 cases, leading to a discovery ratio close to 100%. The offline lifetime results obtained using g-LLR show that routes exist between the observed pairs for all these cases. When we look into the two cases where d-LLR fails to find a route, however, we notice that in one case, the LLR ends at 0.44 s, and in the other case, the LLR ends at 0.3 s. When the LLR route is being returned by the destination, this route is already broken. That is why d-LLR fails to discover them in these two cases.

Table 1
Lifetime performance of DSR, g-LLR and d-LLR

	RDSR (%)	PL(s)	AL(s)
DSR	67	28.3	None
d-LLR	100	40.26	55.4
g-LLR	100	40.24	60.9

Collisions contribute to the high failure rate of the RDSR in DSR. As queries propagates in a random way, they not only collide with those from neighboring nodes, they also interfere with those from farther nodes. Sometimes, route queries do not reach the destination, while sometimes route replies are interfered in their way back and do not reach the source node. D-LLR is more capable of discovering routes than DSR in several ways. First, by using a relatively longer delay time and a jittering scheme, d-LLR is able to reduce collisions in the first place. Second, d-LLR ensures that nodes at the same hop distance rebroadcast at the same phase, thus propagating the query message in a more organized ring-out pattern. This further reduces the collisions, especially when the query has been broadcasted a few hops away from the center. Finally, destination nodes wait long enough and avoid potential collisions of route reply with the still on-going route query propagations.

Although the initial route discovery delay may become longer, fewer collisions are incurred, and thus this one-time route set-up delay is still within the scale of a second. Once a good route is discovered, this delay will be effectively compensated by the extended traffic flow using the route.

As for the auxiliary LLR, d-LLR can find a route for 351 out of 500 cases, resulting in a discovery ratio of about 75%. Meanwhile, g-LLR shows that for 437 out of 500 cases, there exists an auxiliary LLR that is one hop longer than the primary LLR, and for 33 cases, g-LLR discovers LLRs that are more than one hop longer than the primary LLR. For the remaining 30 cases, only the primary LLR exits.

The average route lifetime discovered by DSR is about 28.3 s. The average lifetime of the primary LLR discovered by d-LLR is 40.26 s, while the average lifetime from g-LLR is 40.24 s. The primary lifetime of d-LLR is slightly larger than that of g-LLR simply because d-LLR fails to discover the two LLRs with very small lifetime of 0.44 and 0.3 s. These two small lifetimes are not included in calculating the average lifetime, thus resulting in seemingly abnormal lifetimes for d-LLR. Therefore, d-LLR performs almost the same as g-LLR in terms of primary route lifetime. As for the auxiliary route, the average lifetime from g-LLR is 60.9 s, while the average auxiliary lifetime from d-LLR is 55.4 s, only

5 s less than that of g-LLR. Compared to DSR, d-LLR is able to discover a primary route that lasts 40% longer, and an auxiliary route that lasts 100% longer. Despite that the average route lifetime is different for different testing scenarios. Our protocol can discover the best shortest route available, thus reduce the impact from the variance of the route lifetime. The lifetime performance improvement, which occurs without sacrificing route length, directly leads to the routing performance improvement to be shown in the next section.

4.2. LLR routing performance with UDP

In this section, we use UDP as the transport layer protocol and investigate the routing performance of d-LLR and DSR. UDP is a semi-transparent best-effort transport protocol, and it feeds packets to the underlying routing protocol with little modification. A study using UDP provides direct insight into the advantage of LLR over DSR in terms of packet delivery ratio, packet delay and energy consumption.

We will experiment using d-LLR and DSR with various options to test the effect of each option. For d-LLR, we will investigate d-LLR with only the primary route used, denoted as LLR. We will also investigate a fast-switch LLR scheme with the auxiliary route reserved as a backup route. In this scheme, when the primary route is detected as broken, the auxiliary route will be used and a new route discovery procedure will be initiated. This auxiliary route continues to be used until a new route response is received. A new round starts where the new primary route will be applied, and the new auxiliary route will serve as the new backup route. We term this fast-switch scheme LLR-FS.

LLR attempts to discover the best route towards the destination. Therefore, intermediate nodes do not return route caches upon receiving a route request. This is because with high node mobility, route caches are likely to be stale and invalid. If a stale route cache is returned, a longer route discovery delay may occur. Similarly, only one route response needs to be returned by the destination node. This route response already contains the best-effort primary route and auxiliary route from the destination's point of view. As for DSR, we also remove the options of caching and multiple route replies. The removal of these options is in favor of DSR because these options incur long route discovery delay and make the connection between the source and destination unstable. We denote this DSR with no caching as DSR-NC.

Using the same scenarios as in the previous section, we feed the 50 pairs of nodes with traffic at a rate of one packet per second, and run each simulation for 300 s. The average performance of packet delivery ratio (PDR), packet delivery delay (PDD), and energy consumption per packet (ECPP) are shown in Table 2.

LLR achieves a packet delivery ratio of about 98%, while DSR-NC achieves about 95%. DSR-NC drops more packets due to more frequent and longer route recovery. LLR-FS has a packet delivery ratio of 98%, which is slightly lower than that of LLR. This is because LLR-FS may attempt to deliver a packet using an invalid auxiliary route due to erroneous link lifetime estimation. In the random waypoint model, a node may change its direction and thus either extend or decrease its link lifetime with its neighbors. However, earlier link lifetime estimation cannot predict this and have to estimate link lifetime based on the previous linear mobility pattern. Thus, both the primary and auxiliary route lifetime estimates may be erroneous. For LLR, when a primary route breaks, only one UDP packet is lost. Upon the next UDP packet, new routes will be discovered and the packet will be delivered successfully. For LLR-FS, however, the first lost packet only invalidates the primary route, while the auxiliary route still exists. The second packet will be lost if the auxiliary route is also invalid. Since a new route discovery is initiated simultaneously with the attempt of delivering the second packet, the third packet will definitely be delivered using new routes. Therefore, the packet delivery ratio of LLR-FS is slightly lower than LLR, at most one packet per route break.

However, if the auxiliary route of LLR-FS is valid, there will be no packet delivery delay from route discovery for the second packet and the rest of the buffered packets. From Table 2, the average packet delay is about 0.027 s for LLR, and it is an even lower 0.015 s for LLR-FS. With the fast-switch scheme, the delay is greatly reduced since there is

Table 2
UDP performance using DSR and LLR

	PDR (%)	PDD (s)	ECPP (J)
UDP + DSR-NC	95.3	1.226	0.15
UDP + LLR	98.3	0.027	0.10
UDP + LLR-FS	98.0	0.015	0.10

almost no delay from route discovery, and the only delay is from packet forwarding. LLR-FS sacrifices very little packet delivery ratio for a significant delay improvement and thus achieves potential fast recovery for upper transport layer protocols.

DSR, on the other hand, has an average delay as large as 1.23 s. This long delay is mostly due to the delay from route discovery. When a RREQ message arrives at the destination, it is very likely that the local area has high contention from flooded RREQ messages. An ARP process may also be required if a node does not recognize its neighbors. Therefore, the RREP message may not be returned to the source node in a timely manner, and the source node will back off its next route discovery attempt. Packets during the backoff will be buffered at the source node, and the delay for these packets eventually adds up if several route discovery attempts fail. For a similar reason, the energy consumption of DSR is about 50% more than that of LLR due to more frequent route breaks and more route discovery attempts.

The performance comparison of the route discovery cost can be easily deduced. Take Table 1 as an example, the lifetime of LLR is about 1.4 times of that of DSR. The route break frequency is inversely related with the lifetime, thus LLR breaks 0.7 times as often as DSR. Considering that DSR finds a route only 67% of the time, the searching cost of LLR is thus $0.7 \times 0.67 = 45\%$ that of DSR. For the gross cost of routing overhead including both route searching and data forwarding, data traffic rate needs to be considered. The readers are referred to [5] for more details.

4.3. LLR routing performance with TCP

The advantages of LLR can be further revealed when LLR is functioning with a more widely used transport layer protocol, TCP. Previous studies have shown that it is difficult to cooperate TCP with DSR without modifications to these protocols [4]. Packet flow may completely stop and become very hard to recover once a route breaks. The reason lies in both ends of TCP and DSR. For TCP, route breaks may be mistaken for network congestion, and thus TCP may perform an unnecessary slow start once it does not receive a TCP acknowledgement. For DSR, the excessive usage of cache may bring about very long route discovery delays. The combination of both drawbacks causes the TCP performance to be unacceptable for highly mobile scenarios.

Most solutions attempt to solve this problem by allowing TCP to communicate more with the routing protocol and explicitly differentiate network congestion from link failure [4,20,23]. However, the problem can only be partially solved. If a TCP acknowledgement is lost en-route, the source node cannot learn of the link failure, and thus still has to initiate a slow start. Considering the fact that TCP/IP has existed for decades and is a solid and mature protocol, we believe that a more practical approach should start from the routing protocols themselves.

We do not intend to test all combinations of TCP and DSR approaches in this section. Instead, we test two fundamental options. The first option is the same no caching option as in the previous section, which is to completely remove caches from DSR to ensure instant route discovery. The second option is to forbid TCP to backoff when an acknowledgement is not received due to temporary route break, termed as TCP-NB (No Backoff). This is essentially the same idea as explicitly differentiating link breaks with network congestion. For d-LLR, we test the same basic LLR scheme and the advanced LLR scheme with fast-switch LLR-FS.

The performance of these schemes at a packet rate of one packet per second is shown in Table 3. As we mentioned earlier, if the default TCP and DSR is used, TCP barely recovers from the unsynchronized backoff schemes with DSR. The performance under these circumstances is very poor as shown in the first row. The packet delivery ratio is only around 56%. Many packets are still buffered, waiting for a route to be discovered when the simulation ends.

In DSR-NC, we remove the caching technique and allow source nodes to discover a route that is guaranteed to be valid. With caching removed, TCP is better able to deliver packets, providing a

Table 3
TCP performance using DSR and LLR with a traffic rate of one packet per second

	PDR (%)	PDD (s)	ECPP (J)
TCP + DSR	56.7	0.035	0.307
TCP + DSR-NC	91.7	0.118	0.467
TCP-NB + DSR-NC	99.4	0.130	0.554
TCP + LLR	99.9	0.031	0.413
TCP + LLR-FS	99.9	0.020	0.426

92% packet delivery ratio. If we further remove the slow start scheme from TCP, the packet delivery ratio is further improved to 99% in the scheme TCP-NB + DSR-NC. TCP probes periodically when a route breaks, and once a route is discovered, TCP can start its transmission with minimal waiting time. Therefore, almost every packet can be delivered at the end of the simulation. This better packet delivery performance results in a higher energy consumption since more efforts are made to discover routes and push the traffic flow forward.

On the other hand, d-LLR achieves high packet delivery ratio without any need to modify TCP. Since new good routes can be discovered immediately and routes do not break very often, TCP barely needs to back off. As a result, the packet delay is as low as 0.031 s, mostly just from packet forwarding. If we apply the fast-switch technique to LLR, the traffic becomes more continuous, and the packet delivery ratio becomes even higher. Packet delivery delay is further reduced since TCP no longer needs to wait for a route to be discovered. LLR-FS switches to an auxiliary LLR when the primary LLR breaks, and when new packets arrive from TCP, new routes have already been discovered and the new primary LLR will be used for the rest of the transmissions until it breaks again. Similarly, energy consumption is less than DSR due to fewer route discovery attempts.

We also tested a higher packet rate of 10 packets per second. The results are shown in Table 4. When a larger packet rate is applied, the advantage of LLR becomes less obvious since more packets can be delivered once a route is discovered. These packets will smooth out and reduce the effect of high delay and energy consumption from route discovery. Nevertheless, we can still obtain better packet delivery, lower delay and better energy consumption using LLR. The advantage of continuous packet flows from LLR-FS over the basic LLR becomes more obvious since more packets have to be buffered in LLR when no fast switching is available.

Table 4
TCP performance using DSR and LLR with a traffic rate of 10 packets per second

	PDR (%)	PDD (s)	ECPP (J)
TCP + DSR	58.8	0.023	0.275
TCP + DSR-NC	88.7	0.070	0.375
TCP-NB + DSR-NC	98.8	0.065	0.414
TCP + LLR	97.5	0.019	0.382
TCP + LLR-FS	99.7	0.017	0.388

Although we may alter the node mobility patterns and the traffic rates and redo the test, they may not provide more information. The relative performance of LLR compared to DSR is actually determined by only one factor: the traffic load during one established route. Consider two scenarios, one is a scenario with a traffic load of an average of 1 packet per second and a node speed of 1 m/s; the other is with a traffic load of average 10 packets per second and a node speed of 10 m/s. The second scenario breaks 10 times more often than the first scenario. However, the same amount of traffic is sent during one route establishment. Thus, the second scenario is in fact an up-scale of the first scenario. Therefore, despite that there are so many mobility scenarios with different data traffic modes, they only differ in the average route lifetime and the data load within an established route time. Our simulations with a traffic rate of 1 packet per second and a traffic rate of 10 packets per second are thus sufficient to show the trend without loss of generalities.

4.4. LLR performance with inaccurate link lifetime estimation

In the previous studies, even though a link lifetime can be altered by nodes changing directions, we assume that link lifetime is accurately estimated if there are no such unpredictable changes. However, link lifetime estimation based on signal strength inevitably introduces errors from wireless channels, even if nodes do not change their directions. In order to discover how LLR is affected by these errors, we introduce link lifetime errors into our link lifetime estimates and reinvestigate the performance of LLR.

We assume that the path loss of the signal strength follows the shadowing model from [24].

$$PL(d)\ [\mathrm{dB}] = P_0(d_0) + 10n\log\left(\frac{d}{d_0}\right) + X_\sigma \qquad (4)$$

In this equation, the path loss PL is related to three components. The first component, P_0, is the received power at a reference distance d_0. The second component is related to the distance d between the transceivers and the path-loss exponent n. This component indicates the general trend of attenuation with distance. Typically, n is assumed to be 2 for the free space model and 4 for the two-ray ground model. The last component, X_σ, is a zero-mean

Gaussian distributed random variable (in dB) with standard deviation σ. X_σ describes the randomness from shadowing.

Fig. 4 shows an example of link lifetime estimation distribution using 7 signal samples. We choose $n = 2.7$ and $\sigma = 11.8$ from [25]. The X-axis is the estimated link lifetime referring to the normalized link lifetime. The Y-axis is the probability distribution of the estimated link lifetime. This figure can be explained as the link lifetime estimation error pattern, and it can be used to calculated the estimated link lifetime. For example, when the real link lifetime is 3, the link lifetime input to LLR could be 0.9 times the actual link lifetime, resulting in an estimated link lifetime of $0.9 \times 30 = 2.7$ s. Or, it could be 1.4 times the actual link lifetime, resulting in an estimated link lifetime of 4.2 s. However, according to the estimation distribution pattern, the likelihood of estimating the lifetime as 2.7 s should be about 7 times that of estimating the link lifetime as 4.2 s.

The performance of LLR with TCP using erroneous link lifetime estimation is shown in Table 5. As expected, the performance of LLR is degraded by the erroneous link lifetime estimation. The packet delivery ratio drops even lower than the DSR-NR scheme. The LLR-FS, however, is much more robust in maintaining a good performance. Despite the fact that the real lifetime of the primary LLR is likely to be lower than expected, the auxiliary route lifetime is still very likely to be larger than the primary route lifetime. Therefore, when a route breaks, the traffic flow still can be continuous.

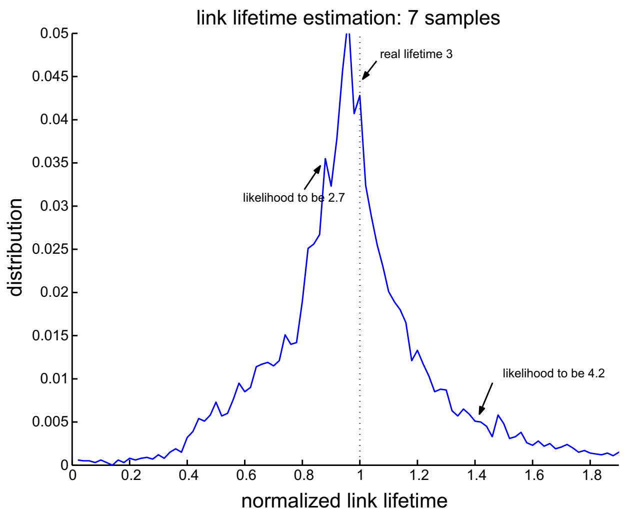

Fig. 4. Link lifetime estimation error pattern. The X-axis indicates the normalized link lifetime estimation. The Y-axis indicates the probability distribution of lifetime estimation. For example, when the real lifetime is 3, the likelihood of estimation as 2.7 is 7 times that of estimation as 4.2.

Table 5
TCP performance using LLR with inaccurate link lifetime estimation for different traffic rates

	PDR (%)	PDD (s)	ECPP (J)
TCP + LLR-1	98.7	0.043	0.431
TCP + LLR-FS-1	99.0	0.028	0.445
TCP + LLR-10	87.0	0.022	0.374
TCP + LLR-FS-10	99.2	0.021	0.388

LLR-FS actually provides an automatic tradeoff method between energy and link lifetime estimation accuracy. When lifetime estimation is accurate, LLR-FS is both energy-efficient and ensures continuous traffic flows. When lifetime estimation becomes inaccurate, LLR-FS still ensures continuous traffic flow, but it requires more frequent route searches. In other words, LLR-FS is self-adaptive to the accuracy of the current link lifetime estimation method. Therefore, it can cooperate with current lifetime estimation schemes to improve the overall ad hoc networking performance without sacrificing the integrity of the existing TCP protocols and networking stacks.

5. Conclusions

In this paper, we propose LLR, a long lifetime route discovery approach to improve the performance of ad hoc networks without modifying existing network stacks and protocols. A global LLR algorithm is proposed to study the statistical behavior of long lifetime routes, and a distributed LLR approach is proposed to implement LLR for practical design. Simulation results show that LLR is able to improve the performance of ad hoc networks with high mobility. LLR can take advantage of existing link lifetime estimation technologies. It automatically adapts to different estimation schemes by trading off energy consumption with link lifetime estimation accuracy. D-LLR can be applied to most ad hoc routing protocols as an extension, and it does not require any modification on TCP and existing network architecture.

References

[1] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: T. Imielinski, H. Korth (Eds.), Mobile Computing, vol. 353, Kluwer Academic Publishers, 1996, pp. 153–181.

[2] C. Perkins, E.M. Royer, Ad hoc on-demand distance vector routing, in: Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99), 1999, pp. 90–100.
[3] V.D. Park, M.S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, in: Proceedings of IEEE INFOCOM, 1997.
[4] G. Holland, N.H. Vaidya, Analysis of TCP performance over mobile ad hoc networks, in: Proceedings of Mobile Computing and Networking, 1999, pp. 219–230.
[5] Z. Cheng, W. Heinzelman, Exploring long lifetime routing in ad hoc networks, in: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2004), 2004.
[6] D. De Couto, D. Aguayo, B. Chambers, R. Morris, Performance of multihop wireless networks: Shortest path is not enough, in: Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I), 2002.
[7] J. Boleng, W. Navidi, T. Camp, Metrics to enable adaptive protocols for mobile ad hoc networks, in: Proceedings of the International Conference on Wireless Networking (ICWN 2002), 2002.
[8] M. Gerharz, C.D. Waal, M. Frank, P. Martini. Link stability in mobile wireless ad hoc networks, in: Proceedings of the 27th IEEE Local Computer Networks (LCN 2002), 2002.
[9] A.B. McDonald, T. Znati. A path availability model for wireless ad-hoc networks, in: Proceedings of the IEEE Wireless Communications and Networking Conference 1999 (WCNC'99), 1999.
[10] C.K. Toh, Associativity-based routing for ad hoc mobile networks, International Journal on Wireless Personal Communications 4 (2) (1997).
[11] R. Dube, C.D. Rais, K.Y. Wang, S.K. Tripathi, Signal stability-based adaptive routing(ssa) for ad hoc mobile networks, IEEE Personal Communications 4 (1) (1997) 36–45.
[12] S. Lee, M. Gerla, Split multipath routing with maximally disjoint paths in ad hoc networks (2001).
[13] L. Li, J. Halpem, Z.J. Hass, Gossip-based ad hoc routing, in: Proceedings of IEEE INFOCOM, 2002, pp. 1707–1716.
[14] N. Panchal, N.B. Abu-Ghazaleh, Active route cache optimization for on-demand ad hoc routing protocols, in: Proceedings of the International Conference on Networking (ICN'04), 2004.
[15] Y.-C. Hu, D.B. Johnson, Caching strategies in on-demand routing protocols for wireless networks, in: Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000), 2000.
[16] T. Dyer, R. Boppana, A comparison of TCP performance over three routing protocols for mobile ad hoc networks, 2001.
[17] Hala Elaarag, Improving TCP performance over mobile networks, ACM Computing Survey 34 (3) (2002) 357–374.
[18] G. Xylomenos, G. Polyzos, P. Mahonen, M. Saaranen, TCP performance issues over wireless links, IEEE Communications Magazine 39 (4) (2001) 52–58.
[19] Kartik Chandran, Sudarshan Raghunathan, S. Venkatesan, Ravi Prakash. A feedback based scheme for improving TCP performance in ad-hoc wireless networks, in: International Conference on Distributed Computing Systems, 1997, pp. 472–479.
[20] K. Nahm, A. Helmy, C. Kuo, TCP over multihop 802.11 networks: issues and performance enhancement, in: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2005.
[21] T.H. Cormenand, C.E. Leiserson, R.L. Rivest, Introduction to algorithms, first ed., MIT Press and McGraw-Hill, pp. 558–565.
[22] NS-2 documentation. <http://www.isi.edu/nsnam/ns/ns-documentation>.
[23] X. Yu, Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness, in: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, 2004, pp. 231–244.
[24] D.C. Cox, R. Murray, A.W. Norris, 800 mhz attenuation measured in and around suburban houses, AT&T Bell Laboratory Technical Journal (1984).
[25] Theodore S. Rappaport (Ed.), Wireless Communications Principles and Practice, Prentice Hall PTR, 1999.

Zhao Cheng is a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of Rochester. He received a B.S. degree in Radio Engineering from Southeast University, China in 2000 and M.S. 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 Heinzelman is an assistant professor in the Department of Electrical and Computer Engineering at the University of Rochester. She received a B.S. degree in Electrical Engineering from Cornell University in 1995 and M.S. and Ph.D. 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.