LINK CACHE EXTENSIONS FOR PREDICTIVE ROUTING AND REPAIR IN AD HOC WIRELESS NETWORKS

Adrian J. Cahill[1]*, Phillip L. De Leon*[2]*, and Cormac J. Sreenan*[1]

[1]University College Cork
Department of Computer Science
Cork, Ireland
{cahill, cjs}@cs.ucc.ie

[2]New Mexico State University
Klipsch School of Electrical Engineering
Las Cruces, NM 88003
pdeleon@nmsu.edu

KEYWORDS

routing, prediction, cache, wireless, ad hoc, mobile

ABSTRACT

In ad hoc networks that employ source-based on-demand routing, network nodes can use information contained in packet headers to populate a link cache. Given sufficient route requests and network traffic, a link cache will provide a good view of the spatial network topology in the past. This view can be successfully applied to making future routing decisions provided that links are allowed to expire through a timeout mechanism, so that the cache stays fresh. In this paper, we further assume that each node has the ability to determine its location, velocity, and bearing. We propose a link timeout mechanism that is based on mobility prediction, and introduce link cache extensions that incorporate temporal information regarding the future state of the network topology. By utilizing spatial and temporal information, better and more efficient routing decisions can be made. Furthermore, local repairs can be preemptively initiated in anticipation of a link break.

1. INTRODUCTION

Wireless ad hoc networks are distinguished from conventional packet networks by the lack of an established infrastructure of routers. Instead, ad hoc networks rely on network nodes to cooperate in forwarding packets through the network, thus enabling communication between nodes that are outside each others' limited wireless transmission range. In addition, the majority of applications envisaged for such networks require that nodes be mobile. Hence the nodes in a mobile ad hoc network are required to not only perform a routing function, but to do so for a network topology that changes dynamically.

Routing protocols for ad hoc networks can generally be classified as being either *periodic* or *on-demand*. Periodic protocols operate in the same way as well-established protocols such as Distance-Vector routing and Link-State routing, in that they rely on regular exchanges of routing information between network nodes, with the aim of ensuring that each node has a valid route to every possible destination. On-demand protocols do not maintain complete routing information, but instead require a node to attempt to discover a route when that node actually has a packet to send to the destination. There are several factors to be considered in selecting a protocol, including the per-node memory and computation requirements, aggregate power consumption for wireless communication, and the impact of delays due to route discovery in on-demand protocols.

On-demand routing protocols for ad hoc networks use caching in order to avoid having to perform expensive route discovery operations each time a packet is to be transmitted. In source-based routing protocols, the complete route is contained in each packet header. Since each node in an ad hoc network also serves as a router, spatial information regarding the network can be extracted from source-routed headers and used to populate a link cache. Given enough diverse route requests and network traffic, a link cache will provide a good view of the network topology in the past. This view can be successfully applied to making future routing decisions provided that links are allowed to expire through a timeout mechanism, so that the cache stays fresh. Prior work has studied the choice of link timeout mechanisms for source-based on-demand routing, comparing a static and adaptive choice of timeout for the Dynamic Source Routing (DSR) protocol [1]. The static approach assigned the same timeout value for every link cache entry. The adaptive approach assigned a timeout based on the stability of the link endpoints, calculated based on the elapsed time since the link was last used and the last time the link was observed to break.

In this paper, we propose extending the link cache timeout in the DSR protocol with one based on a link expiration time that utilizes a simple model of mobility prediction adopted from [2]. This serves to improve network performance in two ways: 1) link cache management that avoids purging cache entries too early or too late, 2) better route selection by taking advantage of both spatial and temporal data in the link cache. With this extension, we identify two features that are especially relevant to the provision of quality of service in ad hoc networks. Firstly, the ability to select a suitable route after taking into account predicted route lifetimes, hop counts and/or delay. Secondly, the use of preemptive localized route repair to attempt to extend the lifetime of a route that is in active use for a flow of packets, without having to resort to a source-initiated route discovery.

Sections 2 and 3 review the background and related work on ad hoc routing and predictive routing. Section 4 explains the simple mobility model we are currently using for our protocol. Section 5 presents our predictive on-demand routing protocol enhancements, followed in Section 6 by details of predictive route repair. Section 7 concludes the paper.

2. BACKGROUND ON AD HOC ROUTING

There are several routing protocols which have been used for wireless ad hoc networks: Dynamic Source Routing (DSR) [3], Destination Sequenced Distance Vector (DSDV) [4], and ad hoc On-

Demand Distance Vector (AODV) [5] are some of the more common protocols.

In DSDV, each node maintains a routing table which holds information regarding how to reach all other nodes in the network. This information includes the number of hops to reach each a node, the next hop on the route to that node, and a sequence number to distinguish between stale routes and valid routes. This information is updated in one of two ways. A "full dump" is broadcast from each node periodically which contains all entries within the sending node's routing table. This information propagates through the network with each node updating its routing table accordingly. The other method of table updates occurs when an entry within a node's routing table is changed. In this instance only information which has been changed since the last "full dump" is broadcast. Routes generated using DSDV are generally the most optimal which can be obtained, as each node has a complete view of the network.

AODV routing is similar to DSDV, in that they both use routing tables which contain the next hop, number of hops and sequence numbers for nodes of the network. Route discovery is also done in a similar fashion to that of DSDV. AODV on the other hand does not keep an entry for all nodes in the network. Instead each node only stores information regarding routes which that node is currently involved in, or has been involved in. When a node using AODV desires to send a packet, it looks up its routing table, if it has a valid route to the destination, then it sends the packet, otherwise the node must initiate a Path Discovery process, whereby the node broadcasts a Route Request (RREQ) packet to its neighbours, who in turn forward the packet to their neighbours until eventually the packet reaches either the destination or a node which has a route to the destination. A Route Reply (RREP) packet is unicast back along the desired route with each node making an entry in its routing table. In the event of a link failure on the route (in which case the upstream nodes will propagate a link failure notification message back to the source) or the source itself moves, the source node can re-initiate the Path Discovery protocol if necessary.

DSR routing does not use any routing tables, but instead contains a cache of complete routes. When a node using DSR wishes to send a packet to a destination, it looks up its cache to find a route to that node, if there is one present then it will use this route, otherwise a Route Discovery process is begun. In the Route Discovery process the sender broadcasts a Route Request packet to all nodes, looking for a route to the destination. Each node forwards the packet, and adds on its own identifier until the packet reaches the destination or a node which has a valid route to the destination, at which time a Route Reply message is sent back to the source. This Route Reply contains the list of hops on the route from source to destination. This route is stored in the route cache of the sending node. If at any time a link breaks then a Route Error message propagates back to the source node which truncates all routes which contain the broken link.

3. RELATED WORK

A key issue for improving the performance of on-demand protocols such as DSR is the use of caching to avoid unnecessary route discovery operations that are costly in terms of additional delay and power/bandwidth consumption. To this end we propose extending cache entries with information derived from a simple model of mobility prediction, thereby avoiding purging cache entries too early or too late, and allowing better route selection and maintenance.

In [1], the authors analyze and evaluate a large number of caching algorithms which utilize various route cache design choices such as cache structure, capacity, and timeout for the DSR protocol. The authors found that "by utilizing a cache data structure based on a graph representation of individual links rather than complete paths through the network, the routing protocol was much better able to make use of the potential information available to it." In regard to timeouts they concluded that the performance of adaptive caches is comparable to that of well-tuned static caches, and that when using no cache timeout the performance is worse than with timeouts. This suggests the need to remove links at a steady rate so as to keep the cache fresh (reducing route errors) but also requires that valid links should not be unnecessarily removed, as this would lead to an increase in route discoveries to recover lost links. This result leads us to question whether links can be removed from the cache based on mobility predictions of the nodes.

In [2], the authors exploit the regularity of mobility patterns as opposed to assuming a purely random pattern, in order to anticipate network topology changes and perform rerouting prior to route breaks. Spatial locality prediction has been investigated in conventional cellular networks where mobility models employing parameter estimates for such things as speed and bearing are used to improve network capacity and performance. In the wireless ad hoc network, the idea is to "piggyback" GPS position information on data packets which can then in turn be used to estimate a Link Expiration Time (LET) between two nodes. The LET is calculated from motion parameters (speed and bearing) of the nodes for a given transmission range. By predicting expiration times for each link in a route, Route Expiration Times (RETs) can be established during the route discovery process and used in route selection. In the proposed Distance Vector with Mobility Prediction (DV-MP) protocol, routes are selected from a node's route table based on stability (longest RETs). The benefit of prediction is a reduced routing update interval and thus fewer route table broadcasts and exchanges. The tradeoff is that the route with the largest RET may not have the fewest hops or shortest delay. Of additional interest in the DV-MP protocol are mechanisms to switch between an expiring route and a more stable route in the middle of a flow. The authors also sketch how a RET can be used by a destination node in an on-demand protocol to predict when a new route needs to be established for a given flow of packets. They do not address the use of caching in such a protocol.

In [6], the authors propose enhancement of the route cache by predicting the spacial locality of network nodes in order to select a route from the cache based on its likelihood of existence. The approach, however, relies on a user's "behaviour model whereby each user moves among a number of geographic locations (e.g. a residence, a workplace, a playground, etc.) spending some time (with a known probability) in each location before moving on." It is unclear, however, how the probability parameters in the user's behaviour model can be accurately established without relying on an extensive mobility history. Furthermore, there would have to be significant cooperation of all nodes in the network (each node would require its own model with appropriate parameters) and in all likelihood, significant overhead in sharing and updating this information or at least announcing the expected spatial behaviour to other nodes.

In [7], the authors propose an on-demand algorithm that aims to preemptively initiate route (re)discovery by anticipating that the current route is about to break. A route is considered likely to break when the received power on a link becomes close to the minimum detectable power. When this occurs, a warning message is sent to the source indicating the likelihood of disconnection, at which point it can perform route discovery again. The protocol does not con-

sider the use of prediction for other purposes such as cache maintenance, route selection or repair.

4. MOBILITY PREDICTION

With the DSR protocol, spatial information regarding the network is extracted from a route request packet by each node along a route. Given that enough diverse route requests and network traffic will pass through a node over time, the link cache will provide a reasonably good view of the network topology in the *past*. This view can often be successfully applied to making future routing decisions provided that links are allowed to expire through a timeout mechanism (either static or adaptive) so that the link cache is reasonably fresh. With an additional assumption that each node has the ability to determine its location either through GPS or some other positioning system, we can extend the link cache to include temporal or time-domain information regarding the future status of a link even if the network topology is dynamic. In this way the link cache will attempt to provide an accurate view of the network topology in the *future* which can be potentially far more useful in a highly mobile environment or one in which either route requests or traffic through a node is sparse.

We adopt the mobility model from [2], which assumes a free space propagation model where the received signal strength is a function only of the distance to the transmitter (assuming a fixed radiated power from each node). Thus a link from node i to node j can be maintained as long as the distance between the nodes is less than the transmission range, r or equivalently the signal strength is above some threshold. Next, we assume that all nodes are capable of determining a position either through GPS or some other other positioning system and that these positions are time-stamped so that a velocity and bearing can be computed (typical functions in an off-the-shelf GPS receiver). Thus associated with node i will be a coordinate (x_i, y_i), velocity v_i, and bearing θ_i. The expiration time for the link between nodes i and j or equivalently the time it takes (assuming constant velocities and bearings) for the distance between nodes i and j to exceed r is given by

$$t_{ij} = \frac{-(v_x d_x + v_y d_y) + \sqrt{(v_x^2 + v_y^2)\, r^2 - (v_x d_y - v_y d_x)^2}}{v_x^2 + v_y^2} \quad (1)$$

where

$$v_x = v_j \cos\theta_j - v_i \cos\theta_i \quad (2)$$

is node j's velocity in the x-direction relative to node i,

$$v_y = v_j \sin\theta_j - v_i \sin\theta_i \quad (3)$$

is node j's velocity in the y-direction relative to node i,

$$d_x = x_j - x_i \quad (4)$$

is node j's x-coordinate relative to node i, and

$$d_y = y_j - y_i \quad (5)$$

is node j's y-coordinate relative to node i. Compensation of unknown inaccuracies in position, velocity, and bearing can be accomplished through a reduction of Eq. (1). We will ignore this issue for now.

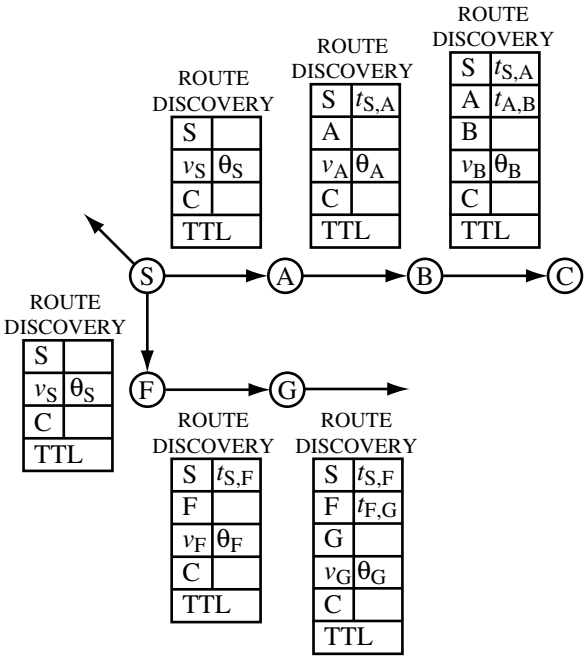

Figure 1: Route discovery process through flooding from Node S to Node C.

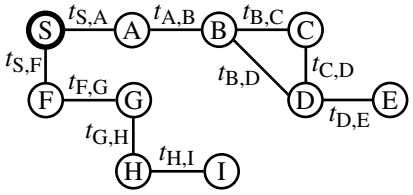

Figure 2: Sample link cache for Node S with link expiration times.

5. PREDICTIVE ROUTING

5.1. Route Discovery

When a source node does not have a route to a destination node in its link cache, it initiates a route discovery by packet flooding as depicted in Fig. 1. The discovery packet will initially contain the source's identification, position, velocity, and bearing as well as the destination identification and time-to-live (TTL). (We will also use TTL to scope route repair as will be discussed in Section 6.) Upon receiving the discovery packet a node will calculate the LET using Eq. (1) to the previous node which forwarded the discovery packet. The LET is recorded in the header as well as the node's identification, position, velocity, and bearing. In addition, the node loads all routing information and LET data contained in the discovery packet into its cache thereby extracting valuable network topology data from the flood. Over time, a node's link cache might resemble Fig. 2. We note that in addition to updating a node's link cache through route discovery, LET data may also be updated during route selection and data transport as described below.

The discovery process continues until the TTL expires. Assuming one or more route discovery packets make it to the destination, the destination node selects a route (more on the selection process

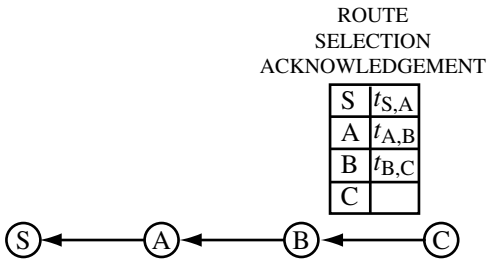

Figure 3: Route selection acknowledgment from Node C to Node S.

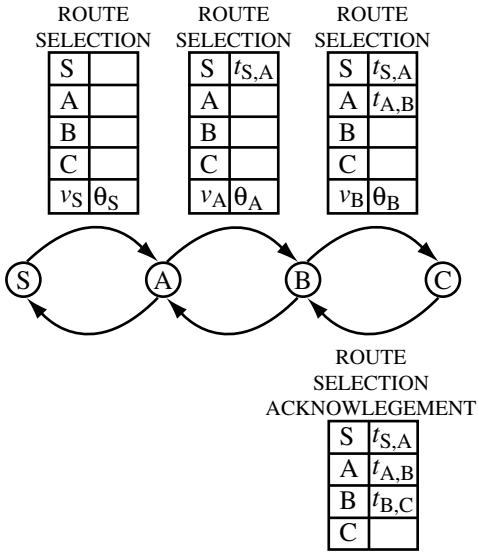

Figure 4: Route selection from Node S to Node C made from Node S's cache data.

later) and forwards a route selection packet back to the source to commence data delivery as in Fig. 3. All nodes in the return route load routing information and LET data contained in route selection packet into their cache.

5.2. Route Selection and Data Transport

When a source node has a route in its link cache to a destination node, it initiates a route selection process to the destination as depicted in Fig. 4. As the route selection packet hops from node i to node j, t_{ij} is calculated as in Eq. (1) and inserted into the packet header while v_j, θ_j replaces v_i, θ_i, respectively, in the header. A route selection acknowledgment is then returned from the destination node. It is important that in the route selection process, participating nodes update LETs for the route since otherwise we would rely entirely on route discoveries (which we want to minimize) as a means for updating link cache data. We note that with sufficient route selection acknowledgments flowing through the network, an extremely useful spatial and temporal view of network topology can be formed with only a few additional bits in the packet header and no periodic table exchanges. As is highly desired, the extent of this network view scales with route participation: the more a node is involved in routing, the more evolved both in space and time its link

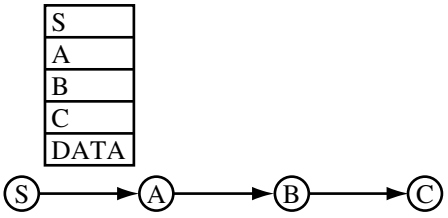

Figure 5: Routing a data packet from Node S to Node C.

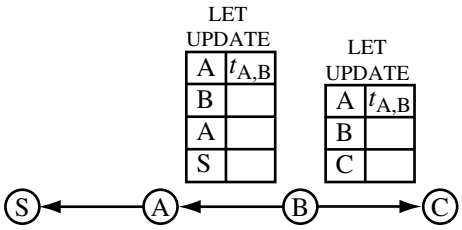

Figure 6: Updating LET data, $t_{A,B}$ to all nodes in a current route.

cache will be. Of course in DSR, new nodes will eventually be included in the link cache as routes expire and route discoveries are forced.

With a link cache built up as in Fig. 2 there may be several routes to a destination from which a source can choose from. The first priority in the selection process will be to determine which routes have a sufficient route expiration time (RET) (equal to the minimum of the LETs from source to destination) assuming file size or stream duration is known. From these routes, the second priority will then be to choose the route with the fewest hops. Thus with the proposed extensions to the link cache, we now have the ability to *jointly* select a route based both on RETs and hop count.

An important issue for ad hoc routing protocols is that of determining when a route is no longer available. In the DSR protocol, complete routing (hop) information is stored in the packet header as in Fig. 5. As the data packet is forwarded to the next (upstream) node, a link-layer acknowledgment is generated. If a node does not receive a link-layer acknowledgment, it assumes the next node can no longer participate in the route and will look in its cache for an alternate route to any of the upstream nodes. If a suitable upstream route exists, the route is repaired, and a route repair notification packet is sent to the source (as described below). If no upstream route exists, the node notifies the source of the broken link and the source may re-examine its link cache or re-initiate route discovery. Depending on the application, end-to-end acknowledgments may also be generated to verify packet delivery. During data transport, if a LET for one of the links in the route changes (due to nodes participating in a separate route discovery or selection process), a LET update is sent to all nodes in the current route as in Fig. 6. Finally, we note that each node in the route will be required to maintain an estimated round trip time (RTT) for purposes of predictive route repair to be described below.

6. PREDICTIVE, LOCALIZED ROUTE REPAIR

A route is initially selected based on RET and possibly hop count. However, due to unexpected changes in a node's trajectory, a link may be predicted to break before the original RET. Particularly

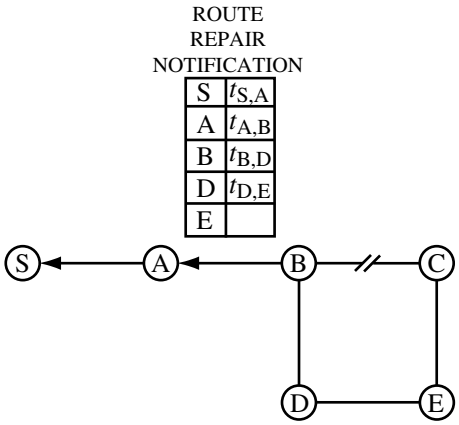

Figure 7: In the original route S-A-B-C-E, the link between Nodes B and C has broken. An alternate route to Node E, however, exists in Node B's cache. Node B sends a route repair notification packet to Node S.

when transporting packet flows such as streaming media, it is desirable to extend the lifetime of the current route if possible by preemptively beginning a local route repair, i.e. upstream from the first node in the anticipated break. Here, two cases exist: 1) a different route to the destination exists in the node's cache and 2) a route does not exist in the cache and route discovery to all upstream nodes is locally initiated. In the first case, a route repair notification packet (Fig. 7) is sent to the source so that future packets may traverse the new route. If the new RET is not sufficient, the source can initiate a new route discovery to the destination node. In the second case, route discovery to a reasonable number (implemented with a small TTL) of upstream nodes is initiated by the first node in the break, i.e. localized route repair. Upon possible receipt of one or more route selection acknowledgments from the upstream nodes, route repair is made, and a route repair notification packet is sent to the source with the new routing information. In both cases, participating nodes update their caches with link data contained in the route discovery and route repair notification packets.

In order to avoid disruption of a flow, route repair must be preemptively initiated and thus a starting time for the repair must be computed. As a worst case bound, we assume the time it takes to perform a route repair, R is no greater than the time it takes to perform a route discovery from source to destination. Thus in order for the source to have new routing information before a link expires, the first node in the break must initiate route repair within $(\mathrm{RTT} \times \lfloor \mathrm{LET}/\mathrm{RTT} \rfloor - R)$ seconds (we have assumed that the node has an estimate of the RTT). To see this another way, we assume that during a flow, the source sends packets with a period of RTT, thus $\mathrm{RTT} \times \lfloor \mathrm{LET}/\mathrm{RTT} \rfloor$ represents the last time the source will send data before the link expires. Backing this figure up by R seconds will enable a new route to be in place before the source is expected to send another packet.

7. CONCLUSION

In this paper, we propose extending the link cache timeout in the DSR protocol with one based on a link expiration time that utilizes a simple model of mobility prediction. This serves to improve network performance in two ways: 1) link cache management that avoids purging cache entries too early or too late, 2) better route selection by taking advantage of both spatial and temporal data in the link cache. With this extension, we identify two features that are especially relevant to the provision of quality of service in ad hoc networks. Firstly, the ability to select a suitable route after taking into account predicted route lifetimes, hop counts and/or delay. Secondly, the use of preemptive localized route repair to attempt to extend the lifetime of a route that is in active use for a flow of packets, without having to resort to a source-initiated route discovery. Future work will focus on a performance comparison of the proposed approach with other existing published proposals, with the aim of quantifying the degree of performance improvement that can be expected.

8. REFERENCES

[1] Y. Hu and D. Johnson, "Caching strategies in on-demand routing protocols for wireless ad hoc networks," in *Proc. 6th Int. Conference on Mobile Computing and Networks (MobiCom)*, pp. 231–242, 2000.

[2] W. Su, S. Lee, and M. Gerla, "Mobility prediction in wireless networks," in *Proc. IEEE Military Communications Conference (MilCom)*, pp. 191–195, 2000.

[3] D. Johnson and D. Maltz, "Dynamic source routing in ad-hoc wireless networks," in *Mobile Computing* (T. Imielinski and H. Korth, eds.), pp. 153–181, Kluwer Academic Publishers, 1996.

[4] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *Computer Communications Review*, pp. 234–244, October 1994.

[5] C. Perkins and E. Royer, "Ad-Hoc on-demand distance vector routing," in *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, 1999.

[6] E. Elmallah, H. Hassanein, and H. AboElFotoh, "Supporting QoS routing in mobile ad hoc networks using probabilist locality and load balancing," in *Proc. Globecomm*, pp. 2901–2908, 2001.

[7] T. Goff, N. Abu-Ghazaleh, D. Phatak, and R. Kahvecioglu, "Preemptive routing in ad hoc networks," in *Proc. 7th Int. Conference on Mobile Computing and Networks (MobiCom)*, pp. 43–52, 2001.