Cache Timeout Strategies for on-Demand Routing in MANETs

Sanlin Xu Kim Blackmore Haley Jones
Department of Engineering, Australian National University, ACT 0200
{Sanlin.Xu, Kim.Blackmore, Haley.Jones}@anu.edu.au

Abstract

Varying the route caching scheme can significantly change network performance for on-demand routing protocols in mobile ad hoc networks (MANETs). Initial route caching schemes retain paths or links until they are shown to be broken. However, stale routing information can degrade network performance with latency and extra routing overhead. Therefore, more recent caching schemes delete links at some fixed time after they enter the cache. This paper proposes using either the expected path duration or the link residual time as the link cache timeout. These mobility metrics are theoretically calculated for an appropriate random mobility model. Simulation results in NS2 show that both of the proposed link caching schemes can improve network performance in the dynamic source routing protocol (DSR) by reducing dropped data packets, latency and routing overhead, with the link residual time scheme out-performing the path duration scheme.

I. Introduction

On-demand routing protocols such as the Temporally Ordered Routing Algorithm (TORA), Dynamic Source Routing (DSR) and the Ad Hoc On-Demand Distance Vector (AODV) protocol have proven effective in mobile ad hoc networks. Routing packets are only generated by those nodes that need to react to changes - nodes that are not currently involved in communication can move without causing any extra traffic in the network. Therefore, on-demand routing has relatively low overhead, which is desirable in low bandwidth wireless networks.

However, on-demand routing has the major drawback of introducing latency between route-request arrival and the determination of a valid route [1]. A route is discovered in reaction to a request to send a packet, so the packet cannot be sent before such a route has been found. Routes are found by searching the entire network to find the destination, which is the source of the latency. To avoid the need to repeat the discovery process once the route has been found, on-demand routing protocols utilize routing caches to store previously discovered routes. With mobile nodes leading to frequent topology changes, the cache may contain stale and, consequently, invalid routing information. Stale cache information can degrade network performance, causing increased numbers of dropped data packets, error replies and longer end-to-end delay.

In order to remove stale route information from the cache, various strategies have been introduced. It has been shown that a link cache has the potential to utilize route information more efficiently than a path cache scheme [2]. With link caching every node maintains a graph data structure representing its own view of the network topology, specifying known links. Links are deleted from cache T seconds after they are first discovered. Based on simulations across a range of timeout values, Hu and Johnson [3] showed that networks generally have good performance when the link timeout, T, equals 5 seconds (hence the scheme has come to be known as "static-5"). However, the simulations all involve nodes moving at an average speed of 10m/s, and it is not made clear how to choose the link timeout value in a more general setting.

We claim that the correct choice for the link timeout, in a non-adaptive (or static) scheme, is the expected path duration, a measure of the rate of change of the network topology. When nodes move according to a Random Walk mobility model (RWMM), this quantity can be calculated simply. We demonstrate that, while parameter settings similar to those used in [3] do, indeed, give good performance with $T = 5$s, performance is degraded for $T = 5$ when any of these parameters are varied. Simulations of DSR using the expected path duration as the link timeout value give significantly improved performance over a range of mobility scenarios.

We further consider an *adaptive* scheme and claim that the correct choice for the *individual* link timeout, in this case, is the expected link residual time. This is the expected time until a currently active link breaks. For a number of important synthetic mobility models, this is a function only of the current node separation. We provide a formula for the expected link residual time when the nodes move according to a Random Walk mobility model, as a function of the node separation when the path containing the link is entered into the cache. Simulations confirm that the resulting adaptive caching strategy outperforms the static caching strategies tested.

Other adaptive schemes include that of Qin and Kunz [4] who use the link expiration time (LET), which is the time until the node separation distance exceeds the transmission range, assuming the nodes continue to move with the current velocity. Jiang *et. al.* [5] use individual link timeout values, that are smaller than the LET, based on an estimate of the probability that the link breaks before the LET is reached due to changes in movement direction. Hu and Johnson [3] and Lou and Fang [2] base the individual link timeout on an estimate of the link stability derived from observations of the link duration in the past. These schemes assume a constant speed and direction over the link observation time, making comparisons difficult.

II. Route Caching Strategies

In on-demand routing schemes, every mobile node maintains a cache table, which is a representation of the topology graph of the network. The cache may store complete paths or a set of known links in the network. We focus on link, rather than path, caching strategies. The cache is obtained via route discovery or by overhearing route information from forwarding packets. The information in the cache may no longer be valid as continual node movement means links in the cache may have broken since the information was received. When a source node attempts to send a data packet to a destination node, it executes a graph search algorithm, such as the Dijkstra shortest path algorithm, to find a route to the destination using the links in the cache. The node then attempts to use the path, but the attempt will be unsuccessful if one of the links in the path no longer exists. In this case the link is removed from the cache. If no alternative path is available in the cache, the source node will initialize a route discovery process to search the entire network.

Automatically removing stale caching information can significantly reduce delay in the network, as nodes do not waste time trying to use a path that is no longer valid. Therefore, whenever a valid link is entered into the cache, it can be assigned a link timeout. Until the timeout is reached, the link is considered active. The link timeout is reset each time the link is used successfully by the node. When the link timeout expires, the corresponding link is deleted from link cache.

A. Uniform Timeout Using Path Duration

A simple technique for implementing cache timeouts is to set a uniform value for the timeout for all links entering the cache. Hu and Johnson [3] demonstrated some success with such a caching strategy, setting the link timeout equal to 5 seconds. However, this empirical result was obtained for specific scenarios, in which each mobile node has an average speed of 10m/s. Generally, if the average node speed is higher, links will come and go much more quickly in the network, so it is likely that a much smaller timeout value would be appropriate. On the other hand, if the node speeds are relatively lower, the network topology is more stable and link timeout should be longer.

Each time a path is used successfully, the timeout for each of the component links is reset. A uniform timeout strategy would then result in all links in this path being deleted at the same time (unless they are also components in other paths). Therefore, it is desirable to delete these links at the moment when the path breaks.

The *path duration* is the length of time from the moment the last of the links in the path becomes active until the first moment one of those links breaks. If node velocities change randomly, it is not possible to determine the path duration ahead of time. However it may be feasible to make a local estimate of the *average* path duration for the network (where the average is over the network and over time). Moreover, for synthetic mobility models, it may be possible to calculate the *expected* path duration.

Clearly the expected path duration is not larger than the expected link duration for each of the component links - as soon as one link breaks, the path is broken. In a link cache it would perhaps seem logical to use expected link duration for cache timeout. However, in a uniform timeout strategy, whole paths are deleted at once so, using expected link duration for cache timeout would significantly overestimate the stability of the network topology, introducing routing delays due to the attempt to access stale routes.

In Section III we address the problem of calculating the expected path duration when nodes move according to a particular synthetic mobility model. In a more general setting, the expected path duration could be replaced by the observed *average* path duration. This is good news, because average path duration is computable in a distributed wireless network environment even without global network knowledge [6].

B. Adaptive Timeout using Link Residual Time

Adaptive timeout strategies assign an individual timeout duration for each link. In this case, it is sensible to keep the link in the cache as long as it is active, so the timeout duration is implicitly an estimate of the *link residual time* — the length of time a currently active link is expected to remain active. For a particular active link, the link residual time is less than or equal to the link duration. However, the link residual time may be significantly larger than the *expected* link duration, which is taken over all links in the network. This will occur if the currently observed link is of greater duration than the “average” link. Therefore, there may be great variation in the timeout values assigned to links - significantly smaller than the expected link duration for many links but, at other times, much larger.

We consider situations where link existence is determined completely by the distance between two nodes. If node separation is large at the time the link is entered into the cache, nodes may soon move out of transmission range so the link timeout is small. However, if the two nodes are close together when the link is entered into the cache, the link timeout will be relatively large. Determining the exact values of the link timeout is not simple. If nodes moved with constant velocity, it would suffice to measure the current velocity and calculate the time until the node separation exceeds the transmission range. However, MANETs are designed to operate in a situation where nodes do constantly change velocity. In Section III we demonstrate that the expected link residual time can be calculated for nodes moving according to a Random Walk mobility pattern. Then, in Section IV, we demonstrate the efficacy of an adaptive caching scheme using the expected link residual time.

III. Timeout Calculation for Random Walk

We have proposed both a *uniform* caching strategy, using the expected path duration as the cache timeout, and an *adaptive* caching strategy, using the expected link residual time as the individual link timeout. Now we give approximate analytic expressions for the proposed timeout values when nodes move according to a Random Walk mobility model.

In a RWMM, each node's movement is independent and identically distributed. We divide the nodes' movements into epochs, in which each node moves with a velocity uniformly distributed in speed over $[0, 2\overline{v}]$, and direction over $[0, 2\pi)$. Both the speed and direction change in each epoch but are constant for the duration of an epoch, and are independent of each other.

The expected link residual time and the expected path duration are calculated in [7] and [8]. This is achieved by describing the separation distance between any pair of mobile nodes as a Markov chain process, with an absorbing state. From this it is possible to determine the *link persistence* — the probability that a currently active link will continue to exist until some specified time in the future. The path persistence is the product of the component link persistences. The expected path residual time and expected path duration are determined from the path persistence and, similarly, the expected link residual time and the expected link duration are determined from the link persistence. We show that the expected link residual time is determined only by the current node separation, and the expected path duration is determined by the number of hops in the path.

No closed form expression for the required quantities exists, and the Markov chain calculation is too cumbersome for implementation in MANET nodes. Therefore, we use the simple analytic approximations given in [8]. If the initial separation distance between a pair of nodes is l_0, the expected link residual time may be approximated as

$$E\{\mathcal{R}(l_0)\} \approx \frac{(r + 2\overline{v}/3)^2 - l_0^2}{2(\overline{v}^2 + \sigma_v^2)}, \tag{1}$$

where $\overline{v}$ is the mean speed per epoch for the link between a given pair of nodes, $\sigma_v^2 = v^2/3$ is the speed variance per epoch and r is the transmission range. Consideration of the node separation when the link first comes into existence leads to the conclusion that the expected link duration satisfies

$$E\{\mathcal{D}\} \approx \frac{\overline{v}(12r - \overline{v})}{9(\overline{v}^2 + \sigma_v^2)}. \tag{2}$$

Since all nodes behave identically, the expected path duration is approximately equal to the expected link duration divided by the number of hops in the path. Therefore the expected path duration for a path with h hops satisfies

$$E\{\mathcal{D}(h)\} \approx \frac{E\{\mathcal{D}\}}{h}. \tag{3}$$

In order to calculate the timeout values, the node needs to know $\bar{v}$ and r and, for the adaptive timeout strategy, the initial node separation distance l_0. In our simulations, rather than making estimations, we have simply used the values for $\bar{v}$ and r that were used to generate the node mobility, and have calculated l_0 from the mobility trace. In a practical application, $\bar{v}$ and r must be estimated by the node. The initial node separation could be estimated using the global positioning system (GPS), or by a time differential of arrival (TDOA) scheme, which relies on the different transmission rates of radio frequency and infrared signals [9].

IV. Simulation Results

We investigate the performances of the two proposed link caching schemes utilizing expected path duration and link residual time, respectively, via simulations. If the link timeout is assigned a large value, the number of control overhead packets can be significantly reduced but the end-to-end delay is, consequently increased, and vice versa. As previously mentioned, in [3], a scheme in which the link timeout is set to 5 seconds was proposed, the "static-5" scheme, trading off overhead packets with latency. In this section, we compare our two link caching schemes to this scheme.

A. Simulation Parameters

All the simulations are conducted using the Network Simulator (NS2). The Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs is adopted as the MAC layer protocol. Therefore, the two nodes which make up a link will assign the link timeout and delete the expired link from their respective caches practically simultaneously, avoiding extra overhead traffic.

TABLE I
NS2 SIMULATION PARAMETERS

Parameters	Values	Parameters	values
phyType	Phy/WirelessPhy	Mobility model	Random Walk
Routing protocol	Modified DSR	Number of nodes	50
Propagation	TwoRayGround	Node speed	2-22m/s
Radio transmission range	250m	Packets packed in a frame	64
MAC	DCF of 802.11	Maximum packets to sent	1200
ifq (Queue type)	Droptail priority	Number of sources	20
ifqlen (max packet in ifq)	50	Traffic connection	CBR
Transmission rate	2Mbps	Network area	1500m $\times$ 500m
Simulation time	900 seconds per trial		

The simulation model used in this paper is based upon that used in hu:2000, for comparison purposes. Thus, the MANET simulated in this paper consists of 50 mobile nodes (MNs), moving according to a RWMM, in a bounded area of size 1500m $\times$ 1500m. The Two-Ray Ground Reflection Approximation is used as the radio propagation model with a transmission range of 250m. The radio model is based on the Lucent Technologies WaveLAN 802.11 product, providing a 2Mbps transmission rate. The communication traffic simulated in all scenarios is generated by the NS2 traffic generator script. Similarly to [3], 20 constant bit rate (CBR) connections (each with a maximum of 1200 packets for transmission) are generated, with a packet rate of 4 packets per second. The size of each data packet is 64 bytes, and each node has at most 2 CBR connections at the same time. Note that we choose small packets because this leads to network performance being dominated by changes of network topology rather than other issues, such as network congestion.

The simulated time is set to 900 seconds per trial. For each node speed, 10 different mobility scenario files are generated, and the final network performance metric is the average of these trials. Note that the scenarios were generated in advance, and that the identical scenarios were used to evaluate each of the caching schemes.

B. Simulation Results

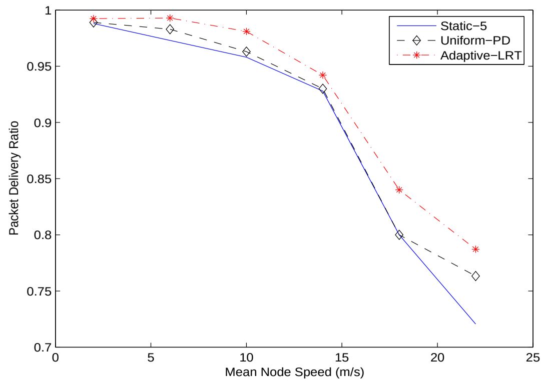
In Fig. 1 network performance, using various measures, with the uniform path duration link caching scheme and the adaptive link residual time link caching scheme are compared with that for the static-5 link caching scheme. The performance measures compared are packet delivery ratio, end-to-end delay, packet overhead and average path length. Note that the packet delivery ratio and the end-to-end delay are calculated with respect to transmitted and received *data* packets, whereas the routing overhead is calculated with respect to *routing control* packets.

We first consider the average path length, as shown in Fig. 1(d). It can be seen that the performances of the three caching schemes are practically indistinguishable. This is because the average path length is largely dependent upon node density, relating to the transmission range, the simulation area size and the number of simulated nodes. The caching scheme is, then, largely inconsequential.

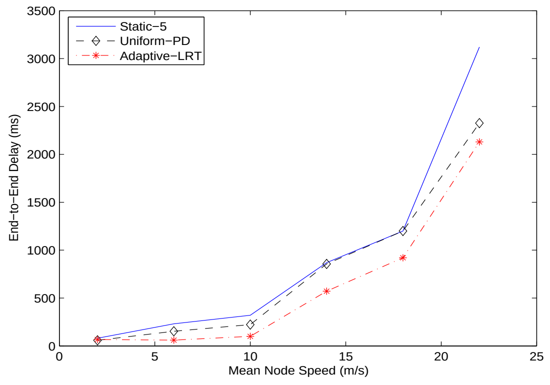
For each of the other performance measures: packet delivery ratio, end-to-end delay and overhead, as would be expected, the adaptive link residual time scheme performs the best overall, followed by the path duration scheme and then the static-5 scheme. This is because the static-5 scheme uses the same link cache timeout regardless of average node speed; the uniform path duration scheme, similarly to the static-5 scheme, utilizes the same value for every link in the network, but chooses a different timeout value for different average node speeds; and, finally, the link residual time scheme adapts to the properties of each link, taking into account average node speed but also including other parameters in its choice of cache timeout value. That is, the link residual time scheme “fits” the given network more closely.

It is interesting to note that there is very little difference in network performance among the caching schemes when the average node speed is low, despite large differences in the actual cache timeout values. For example, for an average speed of 2m/s, the uniform path duration caching scheme has a link timeout of 27.7 seconds, which is more than 5 times greater than that for the static-5 caching scheme. The negligible performance differences are because at low speeds the network has a highly stable topology so that all of the links are highly reliable. Thus, even in the static-5 caching strategy, the link timeout is extended frequently and rarely expires, due to constant incoming communication traffic. Consequently, route discovery is required infrequently resulting in lower routing overhead.

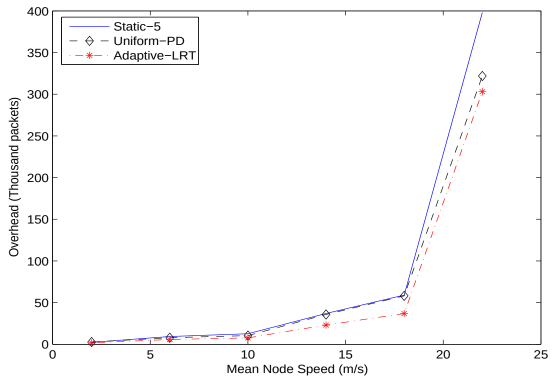
However, it can be seen that the newly proposed caching schemes gradually outperform the static-5 scheme as the average node speed increases, since they can adaptively track the changes of the network topology and the link re-



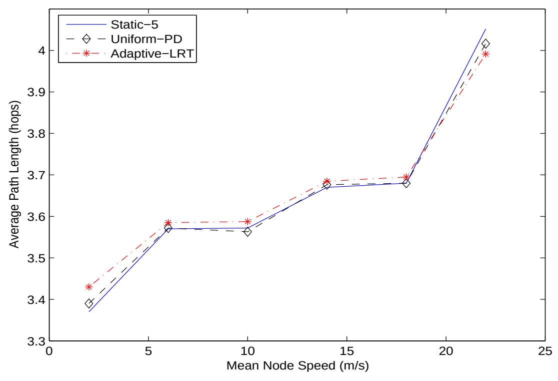
(a) Packet Delivery Ratio



(b) End-to-end Delay



(c) Overhead



(d) Average Path Length

Fig. 1. Performance Comparison between Static-5, Uniform-PD and Adaptive-LRT Link Caching Strategies using the Random Walk Mobility Model

liability. Compared with the static-5 strategy, the adaptive link residual scheme and uniform path duration scheme improve packet delivery ratio 7 percent and 4.3 percent, respectively, when the average node speed is 22m/s. The end-to-end delay is reduced by 990ms and 800ms, and the routing overhead is reduced by 95000 packets and 76000 packets, respectively.

As noted in the Introduction, comparison with existing adaptive schemes is difficult, generally, due to differing mobility models. Further, comparison with the results in [5] is impossible due to the reliance of their results on measured values to estimate the required parameter ϵ.

C. Implementation Issues

We note that the path duration caching scheme is easy to implement. The original DSR simply requires the addition of a 1-byte field into the routing packet, carrying the current clock time plus path duration. The implementation of the link residual time scheme requires the aid of node location information, which is piggy-backed in the packet header. As mentioned in Section II, one way to obtain the node location is to utilize GPS. Alternatively, the relative distance between a pair of nodes can be estimated by other means, such as TDOA.

D. Expected Link Duration and Path Duration

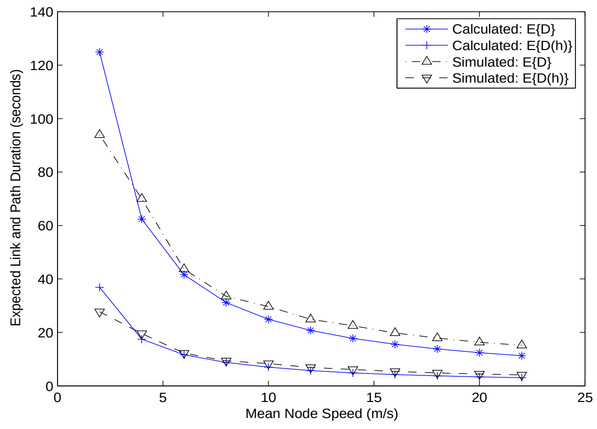


Fig. 2. Expected Link Duration ($E\{\mathcal{D}\}$) and Expected Path Duration ($E\{\mathcal{D}(h)\}$). 50 nodes are simulated in the area 1500m $\times$ 500m with 5000 seconds. The calculated values are from (2) and (3).

As illustrated in Fig. 2, the simulated values of $E\{\mathcal{D}\}$ and $E\{\mathcal{D}(h)\}$ are slightly greater than the calculation due to the effects of a bounded simulation area. The calculations of (2) and (3) are derived for an unbounded scenario. In a bounded simulation environment, MNs are "reflected" back into the simulation area if their movement would otherwise take them outside. Thus, nodes near the edges are more likely to remain within transmission range, artificially increasing the link and path durations compared to those for an unbounded simulation area.

Note that the calculated and simulated expected path duration values, shown in Fig. 2, are 4.8 and 6.1 seconds, respectively, when the average node speed is 10m/s. These values are close to the 5 seconds of the static-5 scheme, as would be expected, because this was the average node speed used in [3] in which the static-5 scheme was proposed.

V. Conclusions

This paper investigates two different mobility metrics in link caching schemes for on-demand routing protocols in wireless ad hoc networks. Expected path duration is used for uniform caching, and expected residual link time is used for adaptive caching. Simple analytic expressions for the proposed timeout values are given. As a result, only small modifications to existing caching schemes are necessary.

Simulation results show that the proposed schemes can enhance network performance for DSR routing over a range of node speeds. The uniform caching strategy is simple to implement, and does not require any extra hardware to calculate node separation. And, the adaptive caching scheme gives even further improvements in network performance. This shows that it can be advantageous to use the additional node location information when choosing timeout values for the link cache.

Routing delay is the major drawback of on-demand routing protocols in MANETs. By slight modification of the existing DSR routing protocol, the two proposed link cache schemes provide solutions to reduce latency and to increase packet delivery ratio.

This work gives the principles of setting link cache timeout in two caching schemes. In future work, we plan implement these methods with respect to other models, such as the Ad Hoc Mobility model and Random Waypoint model.

Acknowledgement

The authors would like to thank Dr. Shengming Jiang from SCUT and Yaoda Liu from Aalborg University for helpful discussions during the development of this work. We also would like to thank Dr. J. Boleng from Colorado School of Mines for provisioning of the NS-2 source code for link duration simulation.

References

[1] B. Liang and Z. Hass, “Optimizing Route-Cache Lifetime in Ad Hoc Networks,” in *Proceedings of IEEE INFOCOM*, vol. 3, Apr. 2003, pp. 1745–1752.

[2] W. Lou and Y. Fang, “Predictive Caching Strategy for On-Demand Routing Protocols in Wireless Ad-Hoc Networks,” *Kluwar Academic J. of Wireless Networks*, vol. 8, pp. 671–679, Aug. 2002.

[3] Y. Hu and D. Johnson, “Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks,” in *Proc. of MobiCom 2000*, Aug. 2000, pp. 231–242.

[4] L. Qin and T. Kunz, “Increasing Packet Delivery Ratio in DSR by Link Prediction,” in *Proc. 36th Hawaii Int. Conf. on Syst. Sciences*, Jan. 2002, pp. 300–309.

[5] S. Jiang, Y. Liu, Y. Jiang, and Q. Yin, “Provisioning of Adaptability to Variable Topologies for Routing Schemes in Manets,” *IEEE J. Select. Areas Commun.*, vol. 22, no. 7, pp. 1347 – 1356, Sept. 2004.

[6] J. Boleng, W. Navidi, and T. Camp, “Metrics to Enable Adaptive Protocols for Mobile Ad Hoc Networks,” in *Proc. of the Int. Conf. on Wireless Networks (ICWN)*, 2002, pp. 293–298.

[7] S. Xu, K. Blackmore, and H. Jones, “Mobility Metrics for Manets Requiring Persistent Links,” in *Proc. of Int. Workshop on Wireless Traffic Measurements and Modeling (WiTMeMo ’05) of MobiSys 2005*. Seattle, June 2005.

[8] ——, “An Analysis Framework for Mobility Metrics in Mobile Ad Hoc Networks,” in *Submitted to J. of Ad Hoc Networks*, 2005.

[9] N. Priyantha, A. Chakraborty, and H. Balakrishnan, “The Cricket Location-support System,” in *Proc. of ACM MOBICOM*. Boston, MA, Aug. 2000, pp. 32–43.