Active Packets Improve Dynamic Source Routing for Ad-hoc Networks

Yu He
Department of Computer Science
University of Southern California
yuhe@usc.edu

Cauligi S. Raghavendra
Department of Computer Science
University of Southern California
raghu@usc.edu

Steven Berson
Information Sciences Institute
University of Southern California
berson@isi.edu

Bob Braden
Information Sciences Institute
University of Southern California
braden@isi.edu

***Abstract*—*DSR (Dynamic Source Routing) is an on-demand routing method for ad-hoc wireless networks that floods route requests when the route is needed. Route caches in DSR are used to reduce flooding of route requests. But with the increase in network size and node mobility, cached routes quickly become stale or inefficient. The consequence is a huge number of routing packets in the network that consume significant bandwidth and computing resources. This paper presents a deployable active network approach to this route cache problem. In our method, an active packet roams around the network, and collects network topology information. By checking the content of this active packet when it passes through, network nodes are able to positively update their route caches. Thus cache miss rates decrease and the route discovery flooding is reduced. Simulation results show that our method reduces the miss rates by up to 60% and routing packet numbers by up to 47%.*

I. Introduction

DSR [4] is a widely used on-demand ad-hoc network routing strategy that uses route caches. DSR floods route requests to find a route that is needed. Since the cost of flooding is high, DSR maintains route caches to store routes that have been found via flooding or through promiscuous overhearing. DSR route caches have two disadvantages. First, DSR takes no measures to avoid flooding for a route to a "new" destination, which has not been overheard yet. DSR has to flood route requests to get the route and save it in route caches. Second, entries in route caches quickly become invalid or inefficient when node mobility is high. When an inefficient route is used, data packets suffer unnecessary delays; when an invalid route is followed, route failures will trigger flooding, creating additional latency for data packets.

Although table-driven routing algorithms ([1], [2], and [3]) could overcome the new-destination problem, they entail too much overhead. In these methods, each node needs to either exchange its view of the whole network topology with its neighbors or flood its link states to the whole network. The involved routing overhead is significant. There are proposals for tackling the stale-cache problem ([7], [8], and [9]). In [7] a link-based cache structure is proposed that encodes more topology information than a traditional path-based cache structure. In [8] the route failure notification mechanism is extended so that all nodes having the related broken links are informed to remove them. The paper also introduces negative caches to avoid using broken paths. In [9], authors investigate ways to validate and shorten cached routes. These proposed schemes are relatively complex. For example, in [7], maintaining the link-based cache is quite difficult in a high mobility network. In [9], it is unclear how to make decisions such as what subset of paths should be validated, and when the validation should happen.

In this paper, we propose a solution to both of the above problems of route caches in DSR using active network techniques ([5] and [6]). In our approach, an active helper is dynamically loaded upon arrival of an active packet, which is originated from an arbitrarily-chosen node. Driven by the active helper, the active packet travels through the network, collects topology information, and has nodes update their route caches. Route entries then contain new routes reflecting the most recent topology changes. Thus both problems are taken care of and flooding is reduced. This paper shows through simulations that improvements from this approach are significant. Besides, the active helper is deliberately designed to be an independent module and does not change any existing aspect of the DSR module. Thus, its deployment in current ad-hoc networks should be relatively easy.

This paper is organized as follows. In Section II, two measurements are given to quantify DSR performance. Then the active approach is described in Section III to improve DSR. Section IV presents simulation results. Finally a conclusion is given in Section V.

II. Route Miss Rate and Routing Packet Number

We define that a route cache miss happens not only when the needed route is not present in caches, but also when a previous hit has turned out to be wrong. Both cases trigger the route request flooding. Thus the *miss rate* quantity gives the degree of the flooding. The *routing packet number* counts all routing-related packets injected by all nodes in an ad-hoc network in order to send a certain number of data packets. This quantity reflects the overhead of the routing protocol.

We use ns-2 [11] simulations to measure these quantities. We simulated different sizes and mobility of ad-hoc networks to see how miss rates and routing packet numbers change in different scenarios. In each network, the moving patterns of nodes follow the Random Waypoint Motion Model [10]. In this model, each node chooses a destination with a uniformly random distribution over the network area, moves there at a speed uniformly between 0 and the maximum allowable velocity, stays there for a specified pause time, and then repeats the behavior. During each simulated session, nodes are uniformly randomly chosen to be the senders or receivers, and carry out CBR communications for a uniformly randomly chosen time. The number of CBR connections in each session is proportional to the number of nodes.

We simulated networks with 3 different sizes and 5 different pause times. The first size is 20 nodes within a 670 m x 670 m area, the second is 50 nodes within a 1500 m x 700 m area, and the third is 100 nodes within 2000 m x 1000 m area. The tested pause times are 300 ms, 100 ms, 50 ms, 20 ms and 0 ms. For the 20 node network, the maximum connection number that could be built during the simulation session is 40, that number is 100 for the 50 node network, and 200 for the 100 node network. The maximum moving speed is 20m/s. The CBR rate is 4 packets per second. The simulation time is 500 s. The results are shown in Fig. 1.

From Fig. 1, in each network, the miss rate and the routing packet number generally increases with increasing in mobility (pause times changed from 300 ms to 0 ms). DSR performs well in small-size networks. However, when the network size gets large (about 100 nodes within 2000 m x 1000 m area), the average route path becomes longer (the nominal transmission range is 250m). Since a longer path is more likely to get invalid than a shorter one, the miss rates go up significantly, and so does flooding. As a result, the routing packet numbers also increases. Note that the increase in routing packet numbers is more apparent than the increase in miss rates. For example, the miss rate goes up from 14% to 21% when the network size changes from 50 to 100 with a fixed pause time of 20 ms, but the routing packet number (per data packet) increases dramatically from 9 to 30. The reason is flooding goes much further in a large network than in a small network. Thus the same miss rate entails much higher overhead in a larger network.

III. Active Network Approach

Our approach to reducing miss rates and routing packet numbers is to keep route caches updated in an active way. Active networks ([5] and [6]), because of their programmability and dynamic-module-loading property, provide a good solution. There is an active helper module that operates on a predefined active packet. This module is independent of the existing DSR module. It is loaded only when the active packet arrives or originates at a node. Then this helper works in the background without interrupting normal operations of the DSR module. This independent design makes it easy to deploy our mechanism.

An active packet is periodically generated by a randomly-chosen node. The basic idea is to drive the active packet to visit each node twice. The first visit is to obtain topology information of the network; the second visit is to update route caches against the obtained information. There is a marker field in the active packet header to indicate if the packet is in the first or the second visit. The payload of the active packet is a connection matrix for the network topology. Upon being dynamically loaded, the active helper checks the header of the active packet. If the packet is on the first visit, the helper checks the nodes neighbor information and updates the connection matrix of the packet before sending it out. If the active packet is in its second visit, the helper makes use of its payload to validate and update the nodes route cache positively. The active helper is also responsible for changing the marker field when the packet finishes its first visit and freeing the packet when the second visit is completed too. Periodically, the active helper and the active packet are triggered and route caches keep being updated. The following sections describe the active helper with more details.

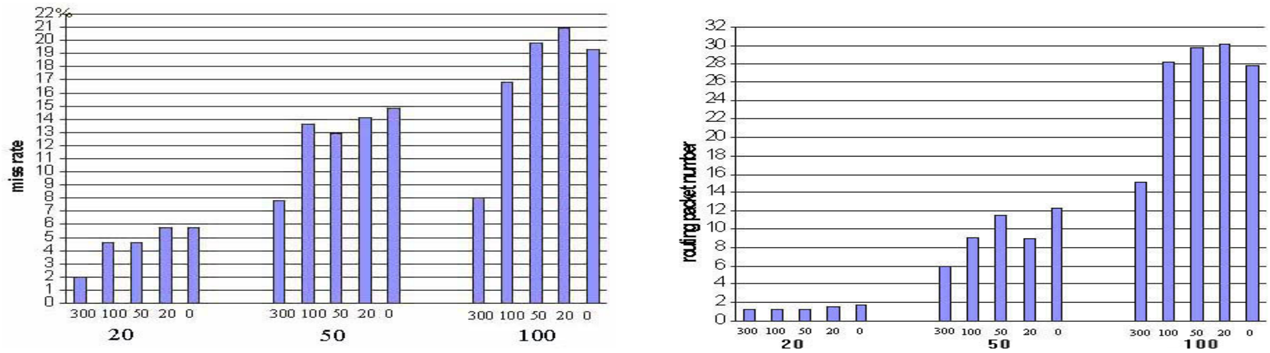

Fig. 1. Route miss rates (percent) and routing packet numbers (per data packet) for networks with 20, 50, and 100 nodes

A. Visiting nodes

It is assumed that each node knows its neighbors. This assumption is reasonable because a mobile node can issue beacons to detect neighbors within a nominal time. The algorithm also assumes that routes are bi-directional, which is the case for the typical wireless MAC protocol 802.11 [12].

Upon being created, an active packet begins to visit nodes in a depth-first order. During its first visit, the connection matrix of the active packet keeps being added to the topology information until the first visit is completed. Due to the depth-first property of the visit, the next node to be visited is generally a neighbor of the node that is currently being visited, except when backtracking happens. In this case, the next node is a neighbor of some node previously visited. A mechanism is needed to have the active packet go back to the previous node so that the visit can continue. Our method is to let the active helper compute a route on the fly from the current node to the previous one according to the information in the connection matrix. This way, less state information is kept for backtracking.

One question is what will happen if the network topology changes in the process of visiting. There are two possibilities; one is links between nodes the active packet has visited still exist, but there are also some new links appearing (including the situation when new nodes come into the network). In this case, the connection matrix, although not a complete reflection of links, is still valid. Thus the paths between visited nodes will still be valid, although may not be optimal. The other possibility is that some links are lost. In this case, the information about these links in the connection matrix is incorrect and paths containing these links are not valid. But the information in the connection matrix about other links should still be valid and useful. To address the above problems, we take measures mentioned in Section III.D to make the visiting of nodes fast compared to node movements.

If there are partitions preexisting in the network before the active packet visit happens, nodes in the same partition as the initiating node get visited, while nodes in other partitions are generally not visited. This is desirable since those visited nodes still get benefits while others remain untouched and don't load the active helper at all.

B. Updating Route Caches

During the second visit of the active packet, the active helper validates and updates the route cache of a node according to the connection matrix in the packet. In the validation phase, each routing entry is checked against the connection matrix and those that disagree are removed. This way, the flooding coming from route failures is reduced. In the updating phase, the active helper adds routes learned from the active packet. There are two ways to decide which routes should be added into the cache. The first approach is *conservative*; the node only adds those routes with destinations identical to the ones that old route entries had before the cache was checked. The idea is to update only preexisting routes. This strategy consumes little space in the route cache. The second approach is *proactive*; besides updating preexisting routes, nodes also add new routes that are valid. This strategy needs more cache space, but provides routes for future use, thus reducing new route request flooding. We implemented both updating methods.

C. Timers

There are two timers used in our approach. The initial timer is used to initiate the visits periodically; each time the initial timer expires, an active packet is generated and is sent out. The other is called maintenance timer. When the maintenance timer expires, route entries added by previous active packets are cleared. This timer has an interval that is a little smaller than the initial timer. It is useful when network partitions happen between visits. The nodes in the disconnected partitions automatically clear the entries left by old active packets, avoiding negative impacts of these possible stale routes.

There is a tradeoff in choosing the interval of the initial timer. The shorter the interval, the more frequent the updates, and the more recent the route cache entries. Thus the route flooding should be less frequent. However, a shorter interval also means that there are more active packets roaming around the network that consume network bandwidth. The value of the initial timer should be chosen to keep both miss rates and routing overhead low.

D. Visiting Efficiency

We took some measures to improve the visiting efficiency in our implementations. First, there is a function called TAP in DSR to allow nodes to promiscuously hear passed packets for the purpose of finding useful routes. In our implementation, we don't allow TAP on the active packets. Thus the TAP processing is saved for active packets and they move more quickly up along the network stack space. Furthermore, we put all out-going active packets in the head of the priority queue on the network interface. The priority order in the queue from high to low becomes active packets, DSR packets and then data packets. The active packets get the highest priority which saves their waiting time.

There are other possible measures. For example, the active packet can be restricted to visit each node only once, instead of twice. The active helper then updates route caches along the way of the visit by using partial topology information in the connection matrix. Another possibility is to make active packets location-aware and limit how far they can go. We are in the process of studying these measures.

IV. Simulation Results

To compare our protocol with the classical DSR routing, we call DSR with our active helper ACT. The same set of simulations as described in Section II was run under both DSR and ACT, for the conservative and proactive ACT. First we tested how much gain there is for the conservative method to update route caches. Fig. 2 shows results of running DSR and ACT on three scenarios, representing different network sizes and pause times.

From Fig. 2, conservative ACT indeed improves upon DSR. This is because updated route caches reflect more recent information for preexisting routes. The flooding triggered by stale routes is thus relieved. The miss rates get a slight improvement. As mentioned before, a slight decrease of miss rates can lead to significant savings for the flooding in a large network. The DSR packets get a good portion of savings (shown as the gray parts in the right half of the figure). But because the ACT packets should also be counted as routing packets (shown as the dark parts), the total savings of routing packets are not high.

On the other hand, if route caches are updated with new routes that may be used in future, the flooding triggered by new requests will also be reduced. For this purpose, we ran the proactive ACT for 3 different size networks, each with 5 different pause times. The results are shown in Fig. 3 through Fig. 5.

There are several points noticeable from the above figures. First, the proactive ACT performs much better than the conservative one. For example, for the 100 node network with the pause time of 0 ms, the conservative ACT improves the miss rate only by 1.7% and the routing packet number only by 1% (Fig. 2), while these two values for the proactive ACT are 33% and 47% respectively (Fig. 5). This is because the larger portion of DSR flooding is saved by also caching future routes. For example, in the scenario of the 100-node network with pause time 100 ms, the proactive ACT drops the DSR packet number from 28.2 to 8.1, saving more than 71% DSR routing packets. Although the active packets add to the number of total routing packets, the saving is still a lot.

For small and medium networks, the reduced miss rates are more impressive, while for larger networks, the savings for routing overhead are more impressive. The reason is that it is more difficult to have the cache entries remain valid with the increase in network size. One link change in a large network leads to a lot more route changes. The same value of a reduction of the flooding rate has more apparent effects in a large network than a small network. In simulations, we get least amount of improvement in routing packet number for the 20-node network. For many scenarios of this network, the packet number even goes up slightly although the miss rate improvements are good. The largest improvement for the miss rate is in the scenario of 50-node network with pause time 0 ms, which is 60%. The largest improvement for the routing packet number comes from the scenario of 100-node network with pause time 0 ms, which is 47%.

V. Conclusion

In this paper, we presented an active network approach for improving route failures and overhead with DSR. This method uses an active packet that periodically visits all nodes it can reach to get network topology information and then uses this information to validate and update the cached routes. With active networking, we not only adjust

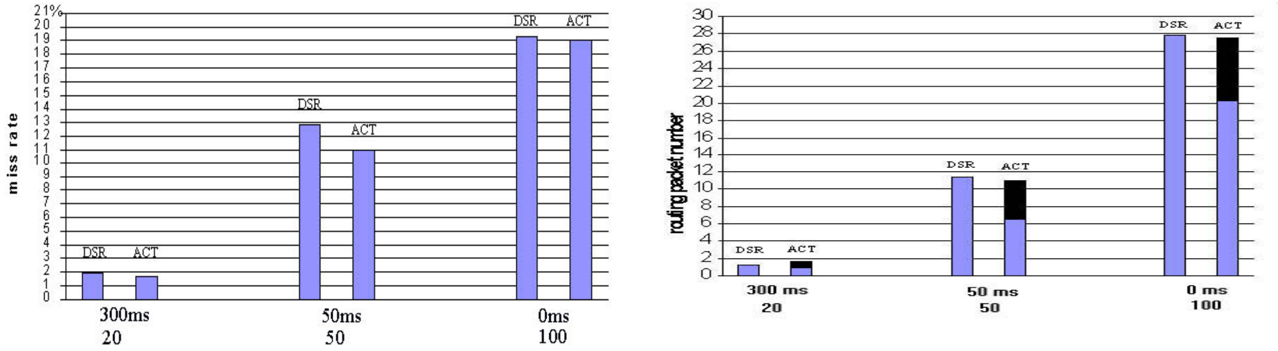

Fig. 2. Miss rates and routing packet numbers (per data packet) from the conservative ACT

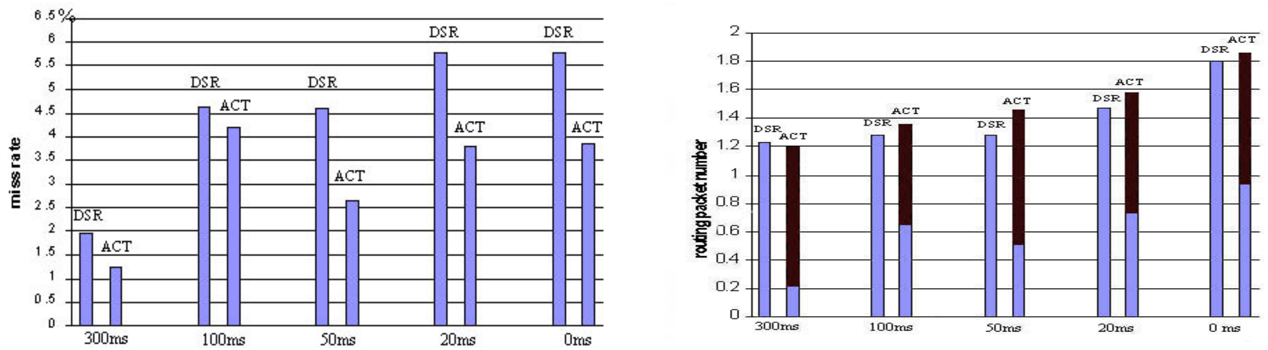

Fig. 3. Miss rates and route packet numbers (per data packet) from the proactive ACT for the 20-node network

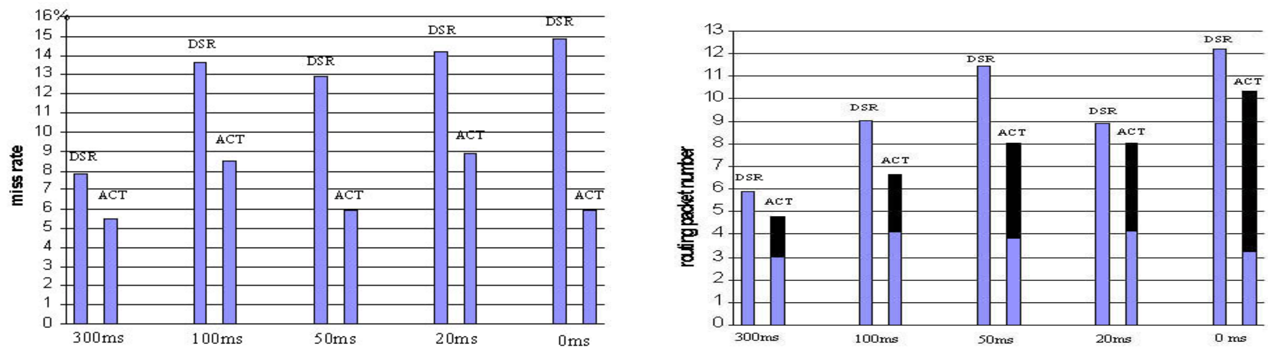

Fig. 4. Miss rates and route packet numbers (per data packet) from the proactive ACT for the 50-node network

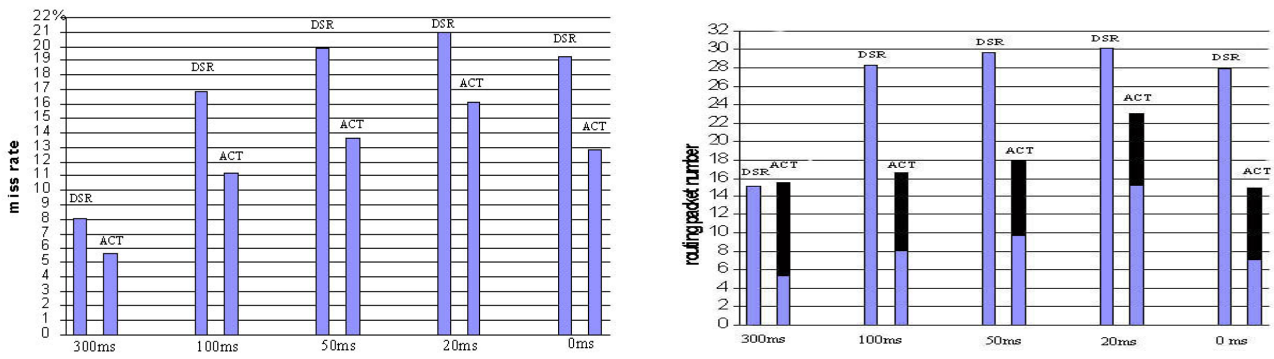

Fig. 5. Miss rates and route packet numbers (per data packet) from the proactive ACT for the 100-node network

existing routes, but also cache future routes based on the topology information. Thus both route request flooding for the stale routes and new routes are reduced. The reduction in the flooding rate also significantly reduces the routing overhead.

REFERENCES

[1] C-C Chiang, M. Gerla, and L. Zhang, Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel, *Proceedings of IEEE SICON 97*, pp. 197-211, Apr. 1997.

[2] S. Murthy and J. J. Garcia-Luna-Aceves, An Efficient Routing Protocol for Wireless Networks, *ACM Mobile Networks and Applications Journal*, pp. 183-197, Oct. 1996.

[3] C. E. Perkins and P. Bhagwat, Highly Dynamic Destination Sequenced Distance-vector Routing (DSDV) for Mobile Computers, *ACM Computer Communications Review*, vol. 24, no. 4, pp. 234-244, Oct. 1994.

[4] D. Johnson, D. Maltz, Y-C. Hu, and J. Jetcheva, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, *Internet Draft*, at *http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-05.txt*, Mar. 2001.

[5] Active Network Working Group, Architectural Framework for Active Networks, at *http://www.cc.gatech.edu/projects/canes/papers/arch-1-0.ps.gz*, July 1999.

[6] D. L. Tenenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. Minden, A Survey of Active Network Research, *IEEE Communication Magazine*, pp. 80-86, January 1997.

[7] Y.-C. Hu and D. Johnson, Caching Strategies in On-demand Routing Protocols for Wireless Ad Hoc Networks, *Proceedings of the International Conference on Mobile Computing and Networks (MobiCom00)*, Aug. 2000.

[8] M. Marina and S. Das, Performance of Route Caching Strategies in Dynamic Source Routing, *Proceedings of the 2nd Wireless Networking and Mobile Computing Workshop*, Apr. 2001.

[9] M. I. Panchal, and N. B. Abu-Ghazaleh, Active Route Cache Optimization for Ad Hoc Networks, at *http://opal.cs.binghamton.edu/ nael/research/papers/infocom02-draft.pdf*

[10] D. A. Maltz, J. Broch, J. Jetcheva, and D. B. Johnson, The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks, *IEEE Journal on Selected Areas in Communications, special issue on mobile and wireless networks*, Aug. 1999.

[11] USC/Information Sciences Institute, Network Simulator, at *http://www.isi.edu/nsnam/ns/*

[12] MAC 802.11, at *http://standards.ieee.org/getieee802/802.11.html*