Localized On-Demand Link State Routing for Fixed Wireless Networks

J. Wang
Dept. of Comp. Sci. & Engg.
University of South Carolina
Columbia, SC, 29208
wang257@cse.sc.edu

S. Lee
Dept. of Comp. Sci. & Engg.
University of Minnesota
Minneapolis, MN 55455
sanghwan@cs.umn.edu

Z. Zhong   S. Nelakuditi
Dept. of Comp. Sci. & Engg.
University of South Carolina
Columbia, SC, 29208
{zhongz,srihari}@cse.sc.edu

1. INTRODUCTION AND MOTIVATION

Fixed multihop wireless networks are gaining in popularity, due to their ease of deployment, for connecting rural communities and for providing broadband access to the Internet [3, 5]. In these networks, the locations of nodes are fixed, and therefore the set of potential neighbors (*adjacencies*) of a node that are within its transmission range is also static. On the other hand, the quality of a wireless channel between adjacent nodes varies frequently due to various factors such as external interference, channel fading, and inclement weather. This work focuses on reliable and scalable routing in such networks where node *adjacencies* are relatively *static* whereas link *qualities* are quite *dynamic*.

Most of the wireless routing schemes have been designed primarily for mobile ad hoc networks with unpredictable topologies, and hence their route discovery and maintenance mechanisms are not ideal for fixed networks. Instead, link-state-based hop-by-hop routing schemes are better suited for these networks, provided they do not require frequent flooding of link state updates. To make link state routing scale for ad hoc networks, *limited dissemination* based schemes have been proposed [4]. Fisheye state routing (FSR) [1] and hazy sighted link state (HSLS) [4] routing schemes update the nearby nodes at a higher frequency than the remote nodes that lie outside a certain scope. The drawback is that the chosen scope can be more than sufficient in some cases and less than necessary in other cases resulting in needless updates or forwarding loops. Our objective is to design a *limited dissemination based routing scheme that ensures loop-free forwarding while notifying only those nodes that need to be informed of a link state change.* We propose such an approach – *localized on-demand link state* (LOLS) – for scalable routing in fixed multihop wireless networks [2].

2. LOLS APPROACH

The central idea behind the LOLS approach is to disseminate a *base topology* reflecting the *long-term* state of each link to all the nodes in the network, and propagate the *short-term* state of *discrepant* links with *negative deviations* w.r.t. the base topology to only the nodes in the neighborhood. Under LOLS, each link is assigned a long-term cost based on its *usual* quality and a short-term cost based on its *current* quality. The set of all links with their associated long-term costs is said to form the base topology. A link is considered to be *discrepant* if its short-term cost is *worse* than the long-term cost[1]. While the *global* base topology updates are performed *infrequently*, the *localized* discrepant link updates are triggered *on-demand*, i.e., a discrepant link's state is propagated only when needed and as far as necessary to enable loop-free forwarding.

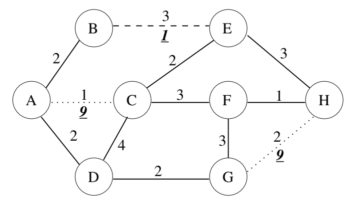

Figure 1: Topology used for illustration of BAF

We now describe *blacklist-aided forwarding* (BAF), a scheme based on the LOLS approach. Under BAF, each packet carries a *blacklist*[2], a set of discrepant links (and their current costs) encountered during its flight. A node *learns* of the non-adjacent discrepant links through the blacklists of packets forwarded by it, and *unlearns* them if not refreshed within a certain time interval. A packet's blacklist is initialized to $\emptyset$ at the source and is revised along the path. A node while forwarding a packet computes the shortest path to its destination according to the short-term costs of links in packet's blacklist and long-term costs of other links in the base topology. If the shortest path contains a discrepant link, that link (and its short-term cost) is *added* to the packet's blacklist. This process is repeated till no new discrepant link is added to the packet's blacklist. The packet is then forwarded to the corresponding next hop. Before forwarding, the packet's blacklist is *reset* to $\emptyset$ if, w.r.t. base topology, the next hop has *lower* cost to the destination

SIGCOMM'05, August 21-26, 2005, Philadelphia, Pennsylvania, USA.

[1]If a link's short-term cost is better than its long-term cost, its use will not cause a loop and an update is not essential.
[2]The notion of blacklist here is different from schemes like DSR. BAF uses blacklist to propagate the state of discrepant links and it may forward packets over blacklisted links.

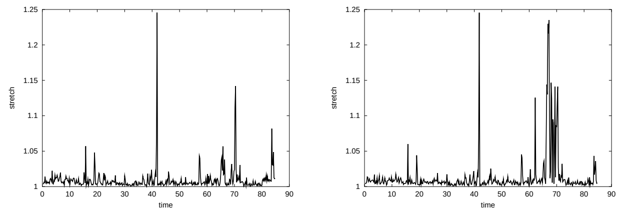

Figure 2: Stretch due to BAF when global link state update interval is: (a) 10 sec; (b) 30 sec

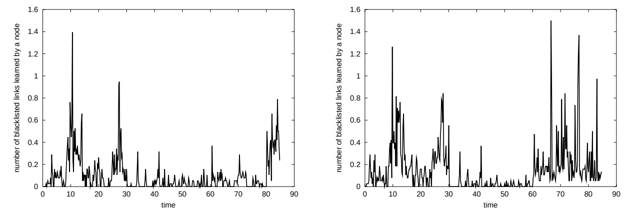

Figure 3: No. of blacklisted links learned by a node under BAF when interval is: (c) 10 sec; (d) 30 sec

than the lowest cost node[3] visited so far by the packet. Effectively, a packet is forwarded with the aid of a non-empty blacklist only when it does not make *forward progress* (w.r.t. base topology) towards its destination.

The operation of BAF is illustrated using a simple example below. Consider the topology shown in Figure 1, where each link is labeled with its long-term cost. Suppose the current short-term costs (shown underlined) of two dotted links (A−C and G−H) are greater than their long-term costs, whereas dashed link (B−E) has smaller short-term cost. Further, assume that all the nodes know the long-term cost of each link but only the adjacent nodes are aware of the short-term costs. Computation of paths based on the current cost of links such as B−E yield better next hops and therefore do not cause loops even without any updates. On the other hand, selection of next hops based on the current state of discrepant links such as A−C and G−H without informing other nodes can result in forwarding loops. For example, let A be the source of a packet and C be its destination. If A computes paths based on the current state of A−C, it will determine that D is the best next hop for C and forward the packet to D. Since D is not aware of the current state of A−C, it will forward it back to A, along its usual shortest path to C, resulting in a forwarding loop.

Under BAF, node A, while computing the next hop for destination C, includes A→C (and its current cost 9) in the packet's blacklist, and forwards it to D. The node D, based on the current cost of blacklisted link A→C, determines that the next hop is C itself. Since the cost to the destination C from the next hop C is 0, which is smaller than the smallest cost so far which is 1, the packet's blacklist is reset to ∅. The packet thus arrives at C along the path A→D→C, and its blacklist while traversing the edges A→D and D→C is A→C and ∅ respectively. On the other hand, packets from A to H are forwarded along A→D→G→F→H, keeping the blacklist unchanged from ∅. In this example topology, packets between any pair of nodes are delivered by BAF without nodes E, F, G, and H being informed of the current state of link A−C, and likewise A, B, C, D and E of G−H.

The description of BAF so far might have given the impression that it is overly complex. On the contrary, BAF is quite simple and scalable to implement. The forwarding operation at a node under BAF amounts to mapping a packet's destination and blacklist to a next hop and a new blacklist based on the discrepant links' state learned by that node and the last updated base topology. This mapping needs to be recomputed only when discrepant links' state changes which is incremental or the base topology changes which is infrequent. In the following, we show that the number of discrepant links learned by a node under BAF is very small and therefore per-packet overhead is not significant.

[3] Such a node with the least cost in the base topology can be identified easily as it can only be either the forwarding node or one of the adjacent head nodes of the blacklisted links.

3. PRELIMINARY EVALUATION

We evaluate BAF using the link-level measurements from MIT Roofnet [3], a 38-node multi-hop wireless mesh network with 352 uni-directional links. The measurement trace records a delivery ratio for each link every 200 ms for 90 sec. The short-term cost of a link is the ETX determined every 200 ms based on its forward and backward delivery ratios. The long-term cost of a link is computed as the average ETX since the last global update. To evaluate the optimality of BAF, we measure the stretch under BAF w.r.t the optimal routing. Fig. 2(a) and Fig. 2(b) show the average stretch among all the node pairs under BAF with global update interval of 10 sec and 30 sec. For both scenarios, the average stretch is quite close to 1 and always less than 1.25. To demonstrate the scalability of BAF, we plot the average number of blacklisted links learned by a node in Fig. 3(a) and Fig. 3(b). The average size of a blacklist maintained by a node under BAF in both cases is insignificant considering that there are 352 links in the network. It is worth noting that increasing the global update interval from 10 sec to 30 sec has little impact on the overall performance of BAF.

4. CONCLUSION

The above results, though preliminary, illustrate the efficacy of localized on-demand link state propagation effected by BAF. We are currently conducting simulations to perform a thorough evaluation of BAF and compare its performance against schemes such as LQSR [5] that are devised specifically for fixed multihop wireless networks.

5. REFERENCES

[1] M. Gerla, X. Hong, and G. Pei. Fisheye state routing protocol for ad hoc networks. IETF Internet Draft, June 2002. draft-ietf-manet-fsr-03.txt.

[2] Localized On-demand Link State Routing. http://www.cse.sc.edu/~srihari/research/lols.html.

[3] MIT Roofnet. http://www.pdos.lcs.mit.edu/roofnet/.

[4] C. Santivanez, R. Ramanathan, and I. Stavrakakis. Making link-state routing scale for ad hoc networks. In *Proc. ACM MobiHoc*, 2001.

[5] Self-Organizing Neighborhood Wireless Mesh Networks. http://research.microsoft.com/mesh/.