

# Blacklist-Aided Forwarding in Static Multihop Wireless Networks

Srihari Nelakuditi\*, Sanghwan Lee†, Yinzhe Yu†, Junling Wang\*, Zifei Zhong\*, Guor-Huar Lu†, Zhi-Li Zhang†

\* University of South Carolina, Columbia  
{srihari,wang257,zhongz}@cse.sc.edu

† University of Minnesota, Minneapolis  
{sanghwan,yyu,ghlu,zhzhong}@cs.umn.edu

**Abstract**—Static broadband wireless networks, due to their ease of deployment, are likely to proliferate in the near future. The major stumbling block, however, is that wireless links are prone to external interference, channel fading, inclement weather, etc. Therefore scalable and reliable routing despite frequent link quality fluctuations is needed for accelerating the growth of these networks. Most of the wireless routing schemes proposed in the literature are less suitable for these networks, as they are designed primarily for mobile ad hoc networks with dynamic and unpredictable topologies. In this paper, we propose a novel link-state-based *blacklist-aided forwarding* (BAF) approach, that takes advantage of the fact that the nodes and therefore their adjacencies are relatively static, for scalable packet delivery in static wireless networks. Under BAF, each packet carries a *blacklist*, a minimal set of degraded-quality links encountered along its path, and the next hop is determined based on both its destination and blacklist. BAF provides loop-free delivery of packets to reachable destinations regardless of the number of degraded links in the network. We evaluate the performance of BAF and show that it is not only reliable but also scalable.

## I. INTRODUCTION

There is a general trend towards wireless connectivity to eliminate the costs and delays associated with building and maintaining a wired infrastructure. Static multihop wireless networks are emerging as the technology of choice for connecting the communities and for providing broadband access to the Internet [1]–[3]. In such static multihop wireless networks, routers (“nodes”) are connected through wireless channels (“links”) instead of wired links, and are responsible for forwarding packets from/to various wireless end systems such as laptops, PDAs, etc. For these wireless networks to be a viable alternative, they should offer similar level of service availability and reliability as wired networks. Unfortunately, apart from the component failures that are common in wired networks, wireless links have additional sources of degradation such as external interference, channel fading, and inclement weather. Therefore, reliable and scalable routing despite such fluctuations is essential for overcoming a potential hurdle to widespread deployment of static wireless networks.

A key characteristic of the aforementioned networks is that the nodes and their *adjacencies* (potential neighbors) are relatively static, whereas the state of links can be quite dynamic.

In other words, the state of a link may vary frequently between good channel conditions and noisy channel conditions, while the set of potential neighbors of a node changes rarely. Most of the wireless routing schemes [4]–[6] proposed in the literature have been designed primarily for *mobile* ad hoc networks with dynamic and unpredictable topologies, and thus are not ideally suited for static networks. These schemes in general assume that individual wireless devices are responsible for discovering and maintaining routes to other peer devices, hence energy and mobility are major concerns. Though some schemes [7], [8] have been proposed recently for wireless mesh networks, their route discovery and packet forwarding mechanisms are similar to mobile ad hoc routing schemes. We believe schemes based on traditional link state routing are better suited for static multihop wireless networks provided they do not incur the expense of frequent link state updates.

In this paper, we propose and develop a novel link-state based routing approach – *blacklist-aided forwarding* (BAF) – for scalable packet delivery in static multihop mesh networks. The proposed approach aims to balance the trade-offs in reliability (high packet delivery rate), optimality (routing along the best quality paths) and scalability (routing overheads) by taking advantage of the unique characteristics of static multihop mesh networks. The central idea behind the BAF approach is to let link state packets disseminate a *reference topology* reflecting the adjacencies of nodes and their long-term qualities to the entire network, while regular data packets convey *negative*<sup>1</sup> information on links with *degraded* short-term quality (w.r.t. the reference topology) to the nodes in a neighborhood. Under BAF, each packet carries a *blacklist*<sup>2</sup>, a set of currently degraded links, and it is forwarded along a route based on both the destination and blacklist information. A packet’s blacklist is normally *empty* and the next hop is chosen along a path with *decreasing cost* (according to the reference topology) to the destination. But when such *greedy forwarding* is not possible at a node due to the degradation of an adjacent link, that link is *added* to the packet’s blacklist and forwarded to an alternate next hop. The blacklist of a packet is *reset* to empty when the next hop has lower cost to the destination than any of the nodes visited so far by the packet.

This work was supported in part by the National Science Foundation under the grants ITR-0085824, CNS-0435444, and CAREER award CNS-0448272. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

<sup>1</sup>If a link’s short-term quality is better than its long-term quality, forwarding over it is loop-free even without informing any other node of its current state.

<sup>2</sup>The notion of blacklist here is different from that in [9], [10]. We use blacklist as a way of propagating link state locally and packets may be forwarded over a blacklisted link as explained in Section VI.

The proposed blacklist-aided forwarding approach has several attractive features: (i) it guarantees loop-free delivery to reachable destinations regardless of the number of degraded links; (ii) when there are no degraded links, BAF behaves just like conventional shortest path forwarding; (iii) only the nodes in the vicinity of a degraded link need to be informed of the link being blacklisted, resulting in localized link state updates; (iv) BAF amounts to on-demand creation of a few alternate routing entries at nodes around vulnerable links, making it a scalable scheme for reliable delivery.

The remainder of the paper is organized as follows. We describe the problem setting in Section II and propose our solution, BAF, in Section III. We present the formal algorithms and implementation details of BAF approach in Section IV. The performance of the proposed approach is evaluated in Section V. In Section VI, the proposed approach is extended to handle more fine-grain link state changes than just good and bad. The related work is discussed in Section VII. Finally, we conclude the paper in Section VIII.

## II. PROBLEM SETTING

The problem scenario addressed in this paper can be described as follows. Consider a multihop wireless network where nodes (wireless routers) are deployed in fixed locations and they are responsible for routing and forwarding packets from/to wireless end systems within their radio range to other wireless end systems in the network (or gateways to the wired Internet). In such networks, the *adjacencies* of a node (the set of neighbors that could be next hops from this node to some destination) are relatively static, since nodes do not “join” or “leave” the network dynamically. On the other hand, the state of the wireless channel or link between two neighboring nodes may oscillate frequently, due to various causes. For simplicity, first we represent link state as either *good* or *disrupted*. A link is considered to be good if packets can be forwarded successfully over that link; otherwise it is regarded as disrupted. We motivate and develop the proposed approach based on this two-state model and later extend it in Section VI to handle fine-grain degradation in the quality of links. Aside from the *instant* state of good or disrupted, a *cost* (e.g. ETX [11] or ETT [8]) is associated with each link that represents its *long-term* throughput or channel capacity. The state (good or disrupted) of a link may fluctuate frequently, whereas its cost varies much slowly. For example, in the topology shown in Fig. 1 where each edge is labeled with its cost and disrupted links are indicated by *dashed* edges, the link A–C has a cost of 1 but it is currently disrupted. Given this scenario, our goal is to design a scalable scheme that always attempts to forward packets along low cost links, while routing around those that are currently disrupted, so as to attain high packet delivery rate and throughput.

Although traditional link state routing schemes can be applied to such static multihop networks, they suffer the problem of poor scalability, while without necessarily ensuring high packet delivery rate: any discrepancy between the previously updated state and the current state of the network could cause

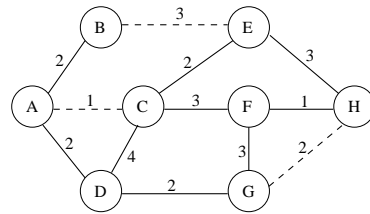


Fig. 1. Topology used for illustration

packet drops or forwarding loops. For example, in the topology of Fig. 1, suppose all the nodes know the cost of each link but only the nodes adjacent to a disrupted link are aware of its current state and other nodes are not informed. Assume A is the source and C is the destination for a packet. Knowing that the link to its next hop C is currently disrupted, A can either drop the packet or forward it along the alternate shortest path via D. If A forwards to D, since D is not aware of the disruption of A–C, it will forward back to A, along its usual shortest path for C, resulting in a loop. This is certainly undesirable particularly when all the nodes are reachable from each other without disrupted links. On the other hand, always maintaining accurate view of the network, to avoid looping or dropping of packets, is either expensive or infeasible when the timescale at which link state varies is relatively small compared to the time required for global dissemination of link state updates and route recomputations.

Several approaches for reducing the overhead of link state routing have been suggested for both ad hoc and wired networks [12]–[16]. Fisheye State Routing (FSR) [12] and Hazy Sighted Link State (HSLs) routing are such ad hoc routing schemes that update the nearby nodes at a higher frequency than the remote nodes. An algorithm proposed in [13] performs local restoration by informing only the routers in the neighborhood about link failure events instead of all routers. Failure insensitive routing approach [15] provides local rerouting for IP networks through interface-specific forwarding and failure inferencing. All these approaches may work fine for individual and independent link disruptions but cannot handle simultaneous disruptions of multiple links which is quite likely in wireless networks. Therefore, a new routing scheme is needed that ensures loop-free delivery to any reachable destination regardless of the number of disrupted links without requiring accurate state of the network. In this paper, we propose and evaluate such a scheme that is ideally suited for static multihop wireless networks.

## III. OUR APPROACH

In this section, we start with how greedy forwarding can be performed to locally route around disrupted links. We then present our blacklist-aided forwarding approach that overcomes the limitations of greedy forwarding and provides reliable delivery despite transient disruptions without frequent global link state updates.

### A. Greedy Forwarding

Consider the topology shown in Fig. 1 and suppose that all the nodes know the cost of each link through global updates, but only the nodes adjacent to a disrupted link are aware of its current state. Now assume that A is the source and H is the destination for a packet. The usual shortest path from A to H is via C. But since A–C link is currently disrupted, we need to find an alternate next hop. We want to choose a next hop such that the packet does not get caught in a forwarding loop. One way to guarantee loop-freedom is to forward the packet along the path with *decreasing cost* to the destination, i.e., at each hop ensure *forward progress* towards the destination. Only the neighbors of a node with forward progress to the destination are considered *feasible* nexthops. The process of forwarding a packet to a feasible nexthop with *maximum* forward progress is referred to as *greedy forwarding*. It is important to note that for greedy forwarding to be loop-free, costs of shortest paths need to be determined consistently at all nodes *based only on the global link state updates*, disregarding any local knowledge of disruptions.

For example, in the topology of Fig. 1, the cost to reach H from A, B, D are 5, 6, and 4 respectively. To reach H from A, node B is not a feasible next hop since its cost to H is 6 which is greater than the cost from A to H. The other neighbor of A, node D is a feasible next hop since its cost to H is 4 which is less than 5. So A forwards the packet to D which in turn forwards to G. Again at G, the adjacent link G–H associated with the usual next hop is disrupted. So G looks for a feasible next hop and finds that F is feasible with its cost to H being 1 which is less than G to H cost of 2. This way, a packet from A is delivered to H successfully with greedy forwarding even though some links along the path are currently disrupted and all nodes do not have the accurate view of the network.

The greedy forwarding described above does not guarantee delivery of a packet to its destination even if there exists a path. A packet is discarded when forward progress is not possible, i.e., it reaches a *deadend* node whose cost to the destination is smaller than any of the possible next hops. For example, in Fig. 1, suppose the source of a packet is A and its destination is C. Given that A–C is currently disrupted, A drops the packet since it cannot find a next hop to C with cost smaller than 1. Note that a packet encounters deadend only when there is an unannounced disrupted link adjacent to the deadend node. If all other nodes are made aware of that disrupted link, there would be forward progress and this node would not be a deadend. For example, if every node is aware that A–C is down, D becomes a feasible next hop for A to reach C. But triggering a global update upon every link state change causes significant overhead due to network-wide flooding of link state packets. It would be ideal to inform only those nodes in the neighborhood of the disrupted link that would be affected by the disruption. But it is not easy to determine the right scope for an update in the presence of multiple simultaneous or overlapping disruptions in the network. An alternate approach is to include such disrupted links that cause a deadend in the

packet itself so that the nodes along the packet’s flight utilize this information in forwarding the packet. That is precisely what is done under *blacklist aided forwarding*.

### B. Blacklist Aided Forwarding

Under blacklist aided forwarding (BAF), a packet can be thought of as being forwarded in two modes: *greedy* and *recovery*. A packet is normally forwarded in greedy mode to a next hop along the path with decreasing cost to the destination. When there is no discrepancy between the previously updated state and the current state of the network, greedy forwarding alone ensures delivery. However, in the presence of disrupted links a packet may arrive at a deadend node. When a packet hits a deadend in greedy mode, instead of discarding the packet, it is forwarded in recovery mode. In recovery mode, packets carry a *blacklist*, a set of disrupted links encountered along the path. A node while forwarding a packet chooses a next hop along a route that does not include blacklisted links. The forwarding of a packet is switched back to greedy mode when it arrives at a node with lower cost to the destination than the node at which it entered recovery mode. This approach guarantees loop-free reliable delivery even in the presence of many disrupted links.

BAF requires that each packet carries an additional field *blist*, the set of blacklisted links, apart from the *dest* field for the purpose of forwarding. There is actually *no explicit forwarding mode*. Instead, the next hop is determined based on both *dest* and *blist* fields of the packet. For ease of understanding, however, we can imagine that packet is forwarded in greedy mode when its *blist* is  $\emptyset$  and in recovery mode otherwise. Apart from the *dest* and *blist* fields, for convenience of explanation (though not necessary for forwarding as we will see in the next section), let each packet  $p$  have another field *cost* to keep track of the smallest cost to  $p.dest$  seen so far by  $p$ . This  $p.cost$  would be the same as the cost to  $p.dest$  from the last deadend node if the packet is in recovery mode. Otherwise it would be the same as the cost to the destination from the currently forwarding node.

Consider again the example scenario of Fig. 1, where a packet  $p$  is being forwarded from its source A to destination C. We have seen that under greedy forwarding, A would not be able to find a feasible next hop and therefore drops the packet. With blacklist-aided forwarding, instead of dropping the packet, A includes the link A→C in the packet’s blacklist since the usual nexthop to reach C is C itself, sets  $p.cost$  to 1 and forwards it to alternate next hop D. The node D would compute the next hop without the blacklisted link A→C and finds that the next hop is C itself. Since the cost to the destination C from the next hop C is 0 and therefore smaller than the current  $p.cost$  of 1, the  $p.blist$  is reset to  $\emptyset$  and  $p.cost$  is set to 0. The packet thus arrives at C along the alternate path A→D→C. The contents of the  $p.blist$  field while  $p$  is traversing the links A→D and D→C are A→C and  $\emptyset$  respectively. The corresponding values of the  $p.cost$  field are 1 and 0. This example, though quite trivial, demonstrates how a packet is forwarded reliably with the aid of a blacklist.

Now consider another scenario where B is the source and E is the destination for a packet  $p$ . Since B→E is currently disrupted, B would forward  $p$  to A. It will set  $p.cost$  to 3 and include B→E in  $p.blist$  since the cost from A to E is not less than 3. Then, A adds A→C to  $p.blist$  and forwards to D. The node D determines C as the next hop based on  $p.blist$ . Before forwarding, it resets  $p.blist$  to  $\emptyset$  since the cost of 2 from C to E is less than  $p.cost$  which is 3. Thus, the path taken by  $p$  would be B→A→D→C→E. The corresponding values of  $p.cost$  at each of these hops would be 3, 3, 2, and 0 respectively. Similarly,  $p.blist$  would be {B→E}, {A→C,B→E},  $\emptyset$  and  $\emptyset$  respectively. Thus the blacklist of a packet *grows when necessary, and shrinks if possible* during the flight to its destination so as to ensure loop-free delivery.

These examples demonstrate the reliability and scalability of BAF. It is easy to see that BAF delivers a packet to its destination if there exists a path without disrupted links. Moreover, under BAF, only a few nodes in the vicinity of a disrupted link need to be notified of the link being blacklisted, amounting to localized link state updates. In our example topology, packet to any destination is delivered by BAF without nodes G, F and H being informed of the disruption of A→C or B→E, and likewise nodes A, B, C, D and E of the disruption of G→H. Such an *on-demand propagation of state* makes BAF a scalable scheme for reliable delivery.

#### IV. ALGORITHMS AND IMPLEMENTATION

In this section, we first provide a formal description of the greedy forwarding algorithm and then build upon it to develop BAF algorithm. We prove that BAF guarantees loop-free delivery to any reachable destination regardless of the extent of disruptions. We also show that blacklist-aided forwarding can be performed by a simple table lookup based on both destination and blacklist fields of a packet. Finally, we extend blacklist-aided forwarding to include learning which offers a trade-off between continual exploration of shortest paths and early avoidance of disrupted links.

Before we present formal procedures, we introduce some notation used in this paper and listed in Table I. We denote by  $\tilde{\mathcal{E}}$ , the set of all edges in the network and by  $\tilde{c}_{i \rightarrow j}$ , the cost of an edge  $i \rightarrow j$  according to the most recent global update. It is assumed that the set of all edges  $\tilde{\mathcal{E}}$  and their costs  $\tilde{c}$  do not change often and any changes are updated globally. All edges in  $\tilde{\mathcal{E}}$  are considered to be in *good* state by default. Due to causes such as interference, an edge may temporarily be in the *disrupted* state. Among the set of all edges  $\tilde{\mathcal{E}}$ , the set of edges that are adjacent (local) to  $i$  and currently in the disrupted state are denoted by  $\tilde{\mathcal{B}}_i^l$ . Similarly the set of edges that are non-adjacent (remote) to  $i$  and *learned* by  $i$  to be in the disrupted state are denoted by  $\tilde{\mathcal{B}}_i^r$ . We use  $\mathcal{P}_{i \rightarrow d}(\mathcal{E})$  to refer to the shortest path from  $i$  to  $d$  with edges in  $\mathcal{E}$  and the corresponding cost is denoted by  $\mathcal{C}_{i \rightarrow d}(\mathcal{E})$ .

##### A. Greedy Forwarding

The procedure GF for selecting a next hop along the shortest path with forward progress from node  $i$  to destination  $d$  given

TABLE I  
NOTATION

$\tilde{\mathcal{V}}$	set of all nodes as per last global update
$\tilde{\mathcal{E}}$	set of all edges as per last global update
$\tilde{c}_{i \rightarrow j}$	cost of edge $i \rightarrow j$ as per last global update
$\tilde{\mathcal{B}}_i^l$	set of disrupted adjacent edges known to $i$
$\tilde{\mathcal{B}}_i^r$	set of disrupted non-adjacent edges known to $i$
$\mathcal{P}_{i \rightarrow d}(\mathcal{E})$	shortest path from $i$ to $d$ w.r.t. edge set $\mathcal{E}$
$\mathcal{C}_{i \rightarrow d}(\mathcal{E})$	cost of the shortest path from $i$ to $d$ w.r.t. $\mathcal{E}$
$p.dest$	destination address in packet $p$
$p.cost$	smallest cost to $p.dest$ seen so far by $p$
$p.blist$	set of blacklisted edges in $p$

the set of all edges  $\mathcal{E}$  and the set disrupted edges  $\mathcal{B}$  is shown in Alg 1. A neighbor  $j$  is considered a feasible next hop with forward progress if the cost of the shortest path from  $j$  to  $d$  is smaller than that from  $i$  to  $d$  (line 4). GF returns  $\emptyset$  when there is no such feasible next hop. If more than one feasible candidate exist, it picks the neighbor  $j^*$  via which  $i$  has the shortest path to  $d$  (lines 5-8). We need to point out that GF is a variant of classic greedy forwarding as it does not always choose a next hop with maximum forward progress. Instead, GF chooses a next hop such that it amounts to shortest path forwarding when there are no disrupted edges. Under greedy forwarding, a node  $i$  forwards a packet  $p$  to next hop  $k$ , where  $k = \text{GF}(i, p.dest, \tilde{\mathcal{E}}, \tilde{\mathcal{B}}_i^l)$ . Here it is assumed that node  $i$  is aware of only its adjacent disrupted edges  $\tilde{\mathcal{B}}_i^l$ . When there is no feasible next hop, i.e., if  $k = \emptyset$ , the packet is discarded. It is easy to show that since the forward progress is consistently ascertained w.r.t. the same set of edges  $\mathcal{E}$ , forwarding using GF is loop-free [17].

**Alg 1** : Greedy Forwarding :  $\text{GF}(i, d, \mathcal{E}, \mathcal{B})$

---

```

1:  $j^* \leftarrow \emptyset$ 
2:  $h^* \leftarrow \infty$ 
3: for all  $j \in \text{neighbors}(i, \mathcal{E} \setminus \mathcal{B})$  do
4:   if  $\mathcal{C}_{j \rightarrow d}(\mathcal{E}) < \mathcal{C}_{i \rightarrow d}(\mathcal{E})$  then
5:      $h \leftarrow \tilde{c}_{i \rightarrow j} + \mathcal{C}_{j \rightarrow d}(\mathcal{E})$ 
6:     if  $h < h^*$  then
7:        $j^* \leftarrow j$ 
8:        $h^* \leftarrow h$ 
9: return  $j^*$ 

```

---

##### B. Blacklist Aided Forwarding

Under BAF, the blist field of a packet is initialized to  $\emptyset$  at the source and it is updated along the path. The formal description of the BAF procedure for forwarding a packet  $p$  by node  $i$  is shown in Alg 2. Under BAF, we first look for a next hop with the smallest path cost and forward progress without the edges in the packet's blacklist (line 1). If no such next hop is found, at least one adjacent link of node  $i$  must be in disrupted state. Hence we need to update the packet's blacklist by adding the disrupted link(s) to the blacklist. The disrupted links adjacent to  $i$  that need to be blacklisted are identified as follows. First, we find the neighbor with the smallest path cost

using only the edges in  $\tilde{\mathcal{E}} \setminus p.\text{blist}$  (line 3). If the link to that neighbor is currently disrupted, then it is added to the blacklist (line 4-5). This process is repeated until either i) we find a next hop, the link from node  $i$  to which is not disrupted, or ii) there is no such next hop (lines 4-6). In the latter case ( $j = \emptyset$ ), the destination is *unreachable* and the packet is dropped. Otherwise, it is forwarded to  $j$ . Before forwarding, if the next hop  $j$  has smaller cost to the destination than any node visited so far by  $p$ , its blacklist  $p.\text{blist}$  is reset to  $\emptyset$  (lines 8-10). Thus, only during the recovery till forward progress can be made, a packet is forwarded with the aid of a non-empty blacklist.

---

**Alg 2** : Blacklist Aided Forwarding : BAF( $i, p$ )

---

```

1:  $j \leftarrow \text{GF}(i, p.\text{dest}, \tilde{\mathcal{E}} \setminus p.\text{blist}, \tilde{\mathcal{B}}_i^l)$ 
2: if  $j = \emptyset$  then
3:    $j \leftarrow \text{GF}(i, p.\text{dest}, \tilde{\mathcal{E}} \setminus p.\text{blist}, \emptyset)$ 
4:   while  $j \neq \emptyset$  &  $i \rightarrow j \in \tilde{\mathcal{B}}_i^l$  do
5:      $p.\text{blist} \leftarrow p.\text{blist} \cup \{i \rightarrow j\}$ 
6:      $j \leftarrow \text{GF}(i, p.\text{dest}, \tilde{\mathcal{E}} \setminus p.\text{blist}, \emptyset)$ 
7:   if  $j \neq \emptyset$  then
8:     if  $\mathcal{C}_{j \rightarrow p.\text{dest}}(\tilde{\mathcal{E}}) < p.\text{cost}$  then
9:        $p.\text{blist} \leftarrow \emptyset$ 
10:       $p.\text{cost} \leftarrow \mathcal{C}_{j \rightarrow d}(\tilde{\mathcal{E}})$ 
11: return  $j$ 

```

---

The rules for updating the blacklist of a packet  $p$ ,  $p.\text{blist}$ , at node  $i$  can be summarized as follows:

- link  $i \rightarrow j$  is *added* to  $p.\text{blist}$  if
  - $i \rightarrow j$  is currently in disrupted state ( $i \rightarrow j \in \tilde{\mathcal{B}}_i^l$ )
  - had  $i \rightarrow j$  been good,  $j$  would be the next hop ( $j \in \text{GF}(i, p.\text{dest}, \tilde{\mathcal{E}} \setminus p.\text{blist}, \emptyset)$ )
  - no feasible next hop exists without  $i \rightarrow j$  ( $\text{GF}(i, p.\text{dest}, \tilde{\mathcal{E}} \setminus p.\text{blist}, \{i \rightarrow j\}) = \emptyset$ )
- $p.\text{blist}$  is *reset* to  $\emptyset$  if
  - there exists a feasible next hop  $j$  ( $j \in \text{GF}(i, p.\text{dest}, \tilde{\mathcal{E}} \setminus p.\text{blist}, \tilde{\mathcal{B}}_i^l)$ )
  - cost to  $p.\text{dest}$  from  $j$  is smaller than any other node  $p$  visited so far ( $\mathcal{C}_{j \rightarrow p.\text{dest}}(\tilde{\mathcal{E}}) < p.\text{cost}$ )

In other words, blacklist-aided recovery is ended and greedy forwarding is resumed as soon as the packet arrives at a node with forward progress. Such a revision of a packet's blacklist makes it carry the minimal set of disrupted links to ensure that its forwarding is loop-free. A formal proof of loop-freedom can be found in [18].

The description of blacklist-aided forwarding so far focused on its functionality in terms of how a next hop is selected for a packet based on its blacklist and how the blacklist is updated along the path to destination. Consider the operations performed by a node  $i$  under BAF while forwarding a packet  $p$ . It has to select a next hop  $j$  after excluding the links in  $p.\text{blist}$  and also update the  $p.\text{blist}$ . The  $p.\text{blist}$  may remain *unchanged* or get *reset* to  $\emptyset$  or *grow* with the addition of adjacent links of  $i$  that are currently disrupted. In all these cases, the *forwarding operation amounts to mapping  $p.\text{dest}$  and  $p.\text{blist}$  to a next hop  $j$  and new  $p.\text{blist}$  based on the current local state at  $i$ ,  $\tilde{\mathcal{B}}_i^l$ , and the last updated global state*

$\tilde{\mathcal{E}}$ . Note that  $\langle p.\text{dest}, p.\text{blist} \rangle$  uniquely determines  $p.\text{cost}$ , i.e.,  $p.\text{cost} = \min\{\mathcal{C}_{u \rightarrow p.\text{dest}}(\tilde{\mathcal{E}}) \mid \forall u \rightarrow v \in p.\text{blist}\}$ . So, there is *no need for an explicit cost field* in the packet. This mapping from  $\langle p.\text{dest}, p.\text{blist} \rangle$  to  $\langle j, p.\text{blist} \rangle$  can be computed on-demand and remembered when node  $i$  first encounters this  $\langle p.\text{dest}, p.\text{blist} \rangle$  pair. Once the mapping is done, thereafter any packet with that  $\langle p.\text{dest}, p.\text{blist} \rangle$  combination can be forwarded simply by a table look up. This mapping has to be recomputed only when  $\tilde{\mathcal{B}}_i^l$  changes which is local and when  $\tilde{\mathcal{E}}$  changes which is rare.

### C. Blacklist Aided Forwarding with Learning

Under the BAF scheme described above, blacklist information carried in a packet is used to forward that packet only and does not influence the forwarding of any other packet. Each packet gets forwarded along the usual shortest path till it encounters a disrupted link and gets rerouted. While this approach delivers packets along the shortest paths when available, it could make packets traverse longer paths. For example, in Fig. 1 suppose A is the source and E is the destination for a packet. Since  $A \rightarrow C$  and  $B \rightarrow E$  are currently disrupted, the path taken by that packet would be  $A \rightarrow B \rightarrow A \rightarrow D \rightarrow C \rightarrow E$ . The corresponding values of  $\text{blist}$  field at each of these hops would be  $\{A \rightarrow C\}$ ,  $\{A \rightarrow C, B \rightarrow E\}$ ,  $\{A \rightarrow C, B \rightarrow E\}$ ,  $\emptyset$  and  $\emptyset$  respectively. As long as links  $A \rightarrow C$  and  $B \rightarrow E$  are disrupted, according to BAF procedure, every packet from A to E follows the same path  $A \rightarrow B \rightarrow A \rightarrow D \rightarrow C \rightarrow E$ . Instead, a node can *learn* from the blacklists of packets arriving at it and utilize this knowledge about *non-adjacent* disrupted links in forwarding other packets. In this example, once A learns about  $B \rightarrow E$  being disrupted from a packet's blacklist, then onwards packets to E can be forwarded along a shorter path  $A \rightarrow D \rightarrow C \rightarrow E$ . We refer to this approach as *blacklist-aided forwarding with learning* (BAFL).

---

**Alg 3** : Blacklist Aided Forwarding with Learning: BAFL( $i, p$ )

---

```

1: while  $(\mathcal{P}_{i \rightarrow p.\text{dest}}(\tilde{\mathcal{E}} \setminus p.\text{blist}) \wedge (\tilde{\mathcal{B}}_i^l \cup \tilde{\mathcal{B}}_i^r)) \neq \emptyset$  do
2:    $p.\text{blist} \leftarrow p.\text{blist} \cup (\mathcal{P}_{i \rightarrow p.\text{dest}}(\tilde{\mathcal{E}} \setminus p.\text{blist}) \wedge (\tilde{\mathcal{B}}_i^l \cup \tilde{\mathcal{B}}_i^r))$ 
3:    $j \leftarrow \text{next}(\mathcal{P}_{i \rightarrow p.\text{dest}}(\tilde{\mathcal{E}} \setminus p.\text{blist}))$ 
4:   if  $j \neq \emptyset$  then
5:     if  $p.\text{blist} \neq \emptyset$  then
6:        $k \leftarrow \text{lastdeadend}(p.\text{dest}, p.\text{blist}, \tilde{\mathcal{E}})$ 
7:       if  $\mathcal{C}_{j \rightarrow p.\text{dest}}(\tilde{\mathcal{E}}) < \mathcal{C}_{k \rightarrow p.\text{dest}}(\tilde{\mathcal{E}})$  then
8:          $p.\text{blist} \leftarrow \emptyset$ 
9:   return  $j$ 

```

---

A node under the BAFL scheme learns about non-adjacent disrupted links from the blacklists of packets arriving at that node. Let  $\tilde{\mathcal{B}}_i^r$  be the set of remote edges that  $i$  learnt to be disrupted. BAFL uses this information in addition to its local knowledge of adjacent disrupted links  $\tilde{\mathcal{B}}_i^l$  while determining the next hop and new blacklist of a packet. The formal description of BAFL procedure is given in Alg 3. It first computes the shortest path to  $p.\text{dest}$  and checks if any of the edges along the shortest path are blacklisted, i.e., in  $\tilde{\mathcal{B}}_i^l \cup \tilde{\mathcal{B}}_i^r$ . If so, those edges are added to  $p.\text{blist}$ . This process is repeated till a path without any blacklisted edges is found or no such

TABLE II  
PARAMETERS OF BAFL SCHEME

$\eta$	limit on the size of a blacklist
$\gamma$	max transmission tries before rerouting
$\tau^l$	refresh interval for adjacent links
$\tau^r$	refresh interval for non-adjacent links

path exists (lines 1-2). Here we abuse the notation and use  $\mathcal{P}_{i \rightarrow d}$  to refer to both the sequence and the set of edges along the shortest path. Once the next hop is found, the shortest path cost to  $p.dest$  from the last deadend node (which can be uniquely determined based on the  $p.blist$ ) is compared with that from the next hop. If the latter is smaller, then the packet's blacklist is reset (lines 6-8). Thus, BAFL ensures that blacklist information carried in a packet remains the same as before under BAF. For example, values of  $blist$  field at each of the hops of the path  $A \rightarrow D \rightarrow C \rightarrow E$  would be  $\{A \rightarrow C, B \rightarrow E\}$ ,  $\emptyset$  and  $\emptyset$  respectively. Essentially, under the same set of disruptions, both BAF and BAFL provide loop-free delivery while BAFL routes around disrupted links sooner than BAF.

**Alg 4** : Processing of a packet under BAFL:  $recv(i, p)$

---

```

1: if  $p.dest = i$  then
2:    $toupper(i, p)$ 
3:    $\tilde{\mathcal{B}}_i^r = \mathcal{B}_i^r \cup p.blist$ 
4:    $j \leftarrow \text{BAFL}(i, p)$ 
5:   while  $j \neq \emptyset$  and  $|p.blist| \leq \eta$  and  $\text{failed}(\text{send}(j, p, \gamma))$  do
6:      $\tilde{\mathcal{B}}_i^l = \mathcal{B}_i^l \cup \{i \rightarrow j\}$ 
7:      $j \leftarrow \text{BAFL}(i, p)$ 
8:    $\text{drop}(p)$ 

```

---

The actions taken by node  $i$  under BAFL upon reception of a packet are abstracted and shown in Alg 4 and its configurable parameters are listed in Table II. If the packet is destined for  $i$ , it is passed to the upper layer. Otherwise it needs to be forwarded. Before forwarding, the set of blacklisted remote edges  $\tilde{\mathcal{B}}_i^r$  is updated to include this packet's blacklist (line 3). Then the next hop is determined using the BAFL procedure. If there is no next hop or if the size of the resulting blacklist of the packet is greater than a preset limit  $\eta$  (BAFL  $\equiv$  GF when  $\eta = 0$ ), packet is discarded. Otherwise, attempt is made to send the packet to the chosen next hop  $j$ . If a certain number of attempts  $\gamma$  fail, that link  $i \rightarrow j$  is blacklisted and added to the set of blacklisted local edges  $\tilde{\mathcal{B}}_i^l$  (line 6). Another next hop is chosen and this process is repeated till either the packet is successfully forwarded or discarded.

The above description of packet processing shows how a node learns about new local and remote disrupted edges. Since an edge may be in disrupted state temporarily, a node needs to *unlearn* blacklisted edges periodically. For the purpose of unlearning the blacklist knowledge, a time stamp is associated with each edge in these sets. Let  $t_{i \rightarrow j}$  be the time an adjacent link  $i \rightarrow j$  is added to the blacklist  $\tilde{\mathcal{B}}_i^l$ . Then it is removed from  $\tilde{\mathcal{B}}_i^l$  at time  $t_{i \rightarrow j} + \tau^l$ , where  $\tau^l$  is a configurable parameter. Similarly a non-adjacent link  $u \rightarrow v$  is removed from  $\tilde{\mathcal{B}}_i^r$  if

TABLE III  
SUMMARY OF DIFFERENCES IN FORWARDING SCHEMES

Scheme	Forwarding Operation
SPF	$\langle i, p.dest, \tilde{\mathcal{E}} \rangle \mapsto j$
GF	$\langle i, p.dest, \tilde{\mathcal{E}}, \tilde{\mathcal{B}}_i^l \rangle \mapsto j$
BAF	$\langle i, p.dest, \tilde{\mathcal{E}}, \tilde{\mathcal{B}}_i^l, p.blist \rangle \mapsto \langle j, p.blist \rangle$
BAFL	$\langle i, p.dest, \tilde{\mathcal{E}}, \tilde{\mathcal{B}}_i^l, p.blist, \tilde{\mathcal{B}}_i^r \rangle \mapsto \langle j, p.blist \rangle$

not refreshed within a time interval  $\tau^r$ . When  $\tau^r$  is set to 0, BAFL behaves like BAF. Otherwise, these values are set such that  $\tau^r > \tau^l$ , i.e., local links are probed more frequently than remote links. It should be noted that learning and unlearning of blacklists can be done using data packets only.

#### D. Summary

The various forwarding schemes described so far differ in the information they use in determining the next hop for a packet. These differences are summarized in Table III. The conventional shortest path forwarding (SPF) scheme computes next hops based solely on the globally updated link state information  $\tilde{\mathcal{E}}$ . It does not use any knowledge it has about the changes in the state of its adjacent edges until they are globally notified lest it could cause forwarding loops. The GF scheme uses a node's awareness of the state of its adjacent edges  $\tilde{\mathcal{B}}_i^l$  and avoids forwarding loops by choosing only those next hops that ensure forward progress. GF would be same as SPF when there are no adjacent disrupted edges, i.e.,  $\tilde{\mathcal{B}}_i^l = \emptyset$ . Our BAF scheme makes the packet carry a blacklist when forward progress is not possible and uses this additional information  $p.blist$  to recover from deadends. Finally, BAFL scheme learns about the remote disrupted edges from packets blacklists and takes advantage of this acquired knowledge  $\tilde{\mathcal{B}}_i^r$  for early rerouting around disrupted edges.

## V. PERFORMANCE EVALUATION

We now evaluate the performance of blacklist-aided forwarding approach in terms of its reliability, optimality and scalability. We show that BAF delivers packets reliably along near-optimal paths with minimal overhead even when many links and nodes are disrupted.

We first discuss our evaluation methodology. We randomly generate a wireless network of 200 nodes in a  $3\text{km} \times 3\text{km}$  field such that no two nodes are too close (at least 70m apart). We assume that the wireless transmission range of a node is 300m. Each link between two nodes that are within the transmission range is assigned a cost (a measure of its long-term throughput) randomly chosen from 100 to 300. This base topology ( $\tilde{\mathcal{E}}$ ) is assumed to be known to all the nodes in the network (via global link state updates). To simulate the effect of disruptions, a certain fraction of the links in the base topology are randomly chosen and designated as disrupted. We then run our BAF and BAFL schemes on the resulting topology for forwarding packets between every node pair and collect various statistics. Each such experiment is run with 5

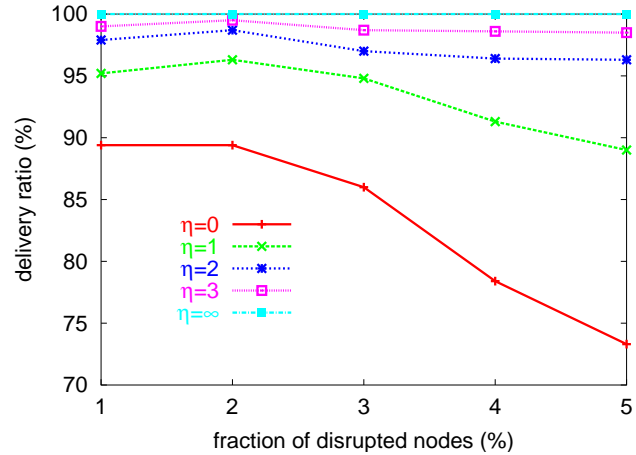
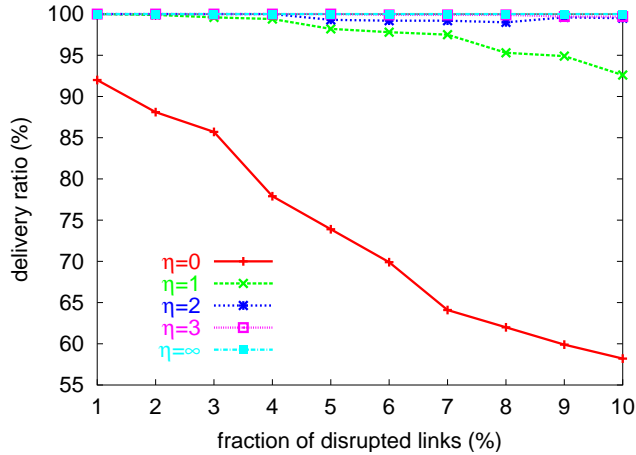


Fig. 2. Reliability of BAF: delivery ratio with (a) link disruptions; (b) node disruptions

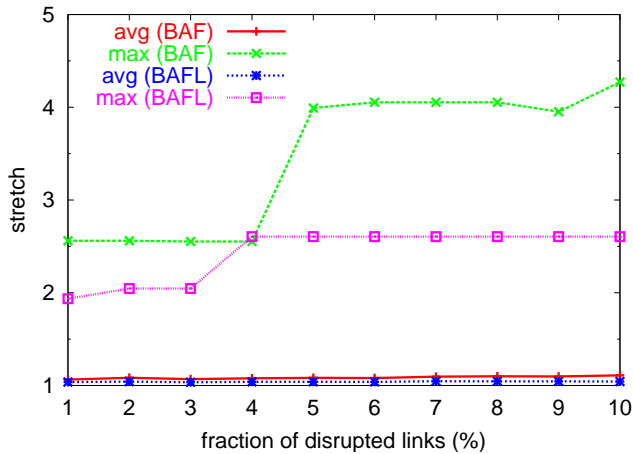


Fig. 3. Optimality of BAF: stretch

different seeds and we report the aggregate results. Note that in this setting, BAF and BAFL differ only in terms of optimality. Therefore we refer to both by BAF unless it is necessary to distinguish them.

#### A. Reliability

We measure the reliability of BAF by the percentage of node pairs between whom packets can be forwarded successfully using BAF among all reachable (having a path without disrupted links) node pairs. Fig. 2(a) shows the reliability of BAF with different  $\eta$  values when the fraction of disrupted links is varied from 1% to 10%. With  $\eta = 0$ , BAF would be same as GF. As the fraction of disrupted links increases, the delivery ratio under GF falls off steeply. This shows that greedy forwarding alone is incapable of handling disruptions. In contrast, the delivery ratio improves dramatically under BAF even when a packet's blacklist is limited to at most one link. BAF with  $\eta = 1$  has a delivery ratio of more than 92% even though 10% of the links are disrupted, which demonstrates the power of

blacklist-aided forwarding approach. As expected, when there is no constraint on the blacklist size ( $\eta = \infty$ ), we have perfect delivery. The strength of BAF is that similar reliability can be achieved even when blacklist size is limited to 3.

In some scenarios, a disruption could be such that a node can not communicate with any of its neighbors, i.e., all its links are disrupted. To see the effect of such correlated disruptions of links on BAF, we evaluated its reliability in the face of node disruptions. Roughly speaking, disruption of  $x$  fraction of nodes corresponds to disruption of  $2x$  fraction of links. Fig. 2(b) shows the delivery ratio of BAF when the fraction of disrupted nodes is varied from 1% to 5%. Once again there is a significant improvement between GF ( $\eta = 0$ ) and BAF ( $\eta > 1$ ). Compared to Fig. 2(a), in Fig. 2(b), the delivery ratio of BAF with node disruptions is relatively lower than that with individual link disruptions. This is not surprising since BAF does not distinguish between node and link disruptions and therefore would have to blacklist many adjacent links of a disrupted node when that node is along the best path to the destination. Consequently, a packet's blacklist may reach the limit and get dropped. However, with only  $\eta = 3$ , the delivery ratio is more than 98% even when 5% of the nodes are disrupted. As before, when the blacklist size is not constrained, the delivery ratio is 100%. These results demonstrate that BAF can deal with disruptions of many links and nodes without global link state updates.

#### B. Optimality

Under BAF, a packet takes the usual shortest path till it encounters a disrupted link and then gets rerouted along the alternate path. Consequently, in the presence of link disruptions, BAF may forward packets along longer paths compared to the optimal paths computed based on the global link state updates. BAFL improves upon BAF by having a node learn from the blacklists of packets arriving at it and utilize this knowledge about non-adjacent disrupted links in forwarding other packets. For example, in the topology of

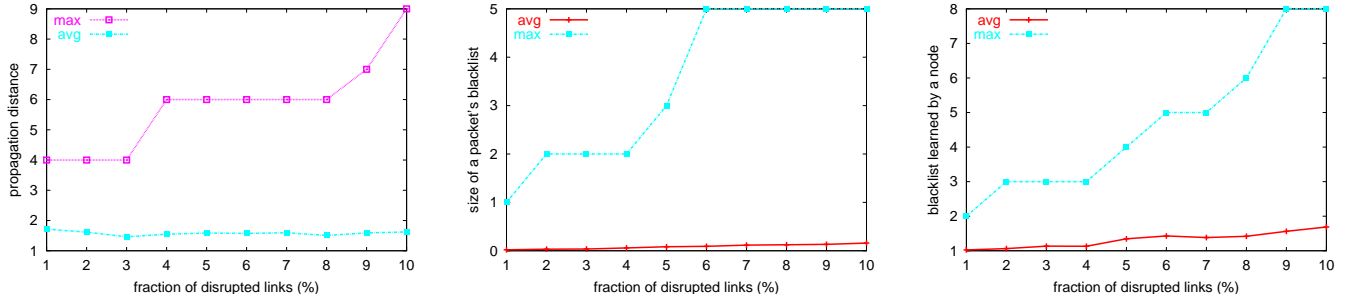


Fig. 4. Scalability of BAF: (a) blacklist propagation distance; (b) size of a packet’s blacklist; (c) size of a node’s blacklist.

Figure 1, packets from A to E are forwarded along the path  $A \rightarrow B \rightarrow A \rightarrow D \rightarrow C \rightarrow E$  while BAFL forwards them along the path  $A \rightarrow D \rightarrow C \rightarrow E$ . Still, forwarding paths under BAF may not be optimal. For example in the topology of Figure 1, due to the disruption of  $A-C$ , both BAF and BAFL forward packets from F to A along the path  $F \rightarrow C \rightarrow D \rightarrow A$ . Had node F been made explicitly notified of the disruption of  $A-C$ , packets would be forwarded along the shorter path  $F \rightarrow G \rightarrow D \rightarrow A$ . However, we show that the extent of this elongation due to BAF and BAFL is not significant.

Let *stretch* of a path between a pair of nodes be the ratio of the lengths of the path under BAF (BAFL) and the optimal shortest path. When the weights of all the links are not same, path length is said to be the sum of the weights of its links. Note that without any link disruptions, there is no difference between the BAF or BAFL paths and the optimal shortest paths and so the stretch is 1. The stretch for packets from F to A due to BAF or BAFL is  $\frac{9}{7}$ . On the other hand, the stretch for packets from A to E due to BAF is  $\frac{12}{8}$  while it is  $\frac{\infty}{\infty}$  (optimal) under BAFL. The average and the maximum stretch due to BAF and BAFL ( $\eta$  set to  $\infty$ ) for the pairs of nodes *affected* by link disruptions is shown in Figure 3. The average stretch due to BAF, across varying fraction of disrupted links, is less than 1.1 while the maximum is 4.3 when 10% links are disrupted. BAFL improves the average stretch slightly compared to BAF but brings down the maximum stretch significantly to around 2.5. It is worth pointing out that blacklist aided forwarding decouples optimality from reliability. It ensures reliable delivery even under severe conditions. Optimality can be controlled by adjusting amount of resources used for update traffic. In other words, BAF allows a trade-off between scalability and optimality of routing without impacting its reliability which is not the case with many other routing protocols.

### C. Scalability

We measure the scalability of BAF in terms of: i) how far the information about a disrupted link is propagated via a packet’s blacklist; ii) how large is the blacklist of a packet; and iii) how many total blacklisted links a node sees under BAF. First, we show in Fig. 4(a) the average and the maximum distance from a disrupted link to the farthest notified node measured in hops. Average blacklist propagation distance is

less than 2 hops regardless of the fraction of disrupted links while the maximum distance goes up to 9 with 10% link disruptions. This points out the limitation of schemes such as [12] based on locally scoped updates with a fixed scope. The chosen scope could be more than sufficient in some cases and less than necessary in other cases resulting in either unnecessary overhead or packet drops and forwarding loops. On the other hand, BAF localizes the blacklist propagation whenever possible and propagates the blacklist to distant nodes when necessary for ensuring loop-free packet delivery.

Next, in Fig. 4(b), we plot the average and the maximum sizes of a packet’s blacklist at each hop. The average is close to 0 and increases only slightly as the fraction of disrupted links increases, i.e., most of the packets do not carry non-empty blacklist. Even the maximum blacklist size is only 5. This shows that per packet overhead due to BAF is negligible. Finally, Fig. 4(c) gives the the number of non-adjacent disrupted links learned by a node through packet’s blacklists. We considered only the nodes that see any blacklist at all. The average number of blacklisted links seen by a node is close to 1 and increases slowly as the fraction of disrupted links increases. The fact that a node would see on the average only 1 and at the most 8 blacklisted links out of around 59 (10%) disrupted links while ensuring reliable delivery is a testimony of the effectiveness of on-demand state propagation approach of BAF. These results confirm that communication and computational overheads due to BAF are quite small and establish BAF as a scalable scheme for reliable delivery.

## VI. FINE-GRAIN BAF

We have so far shown that the BAF approach is suitable for static multihop wireless networks where the state of a link changes frequently between *good* and *bad* due to disruptions. In this section, we demonstrate that the BAF approach can effectively handle even fine-grain fluctuations in link quality. The problem scenario addressed in this section can be described as follows. A *cost* (e.g. ETX [11] or ETT [8]) is associated with each link that represents its quality or throughput. The *short-term* cost of a link may vary considerably while its average *long-term* cost is relatively stable. According to the BAF approach, the long-term cost of a link is conveyed globally, whereas its short-term cost, if worse than the long-

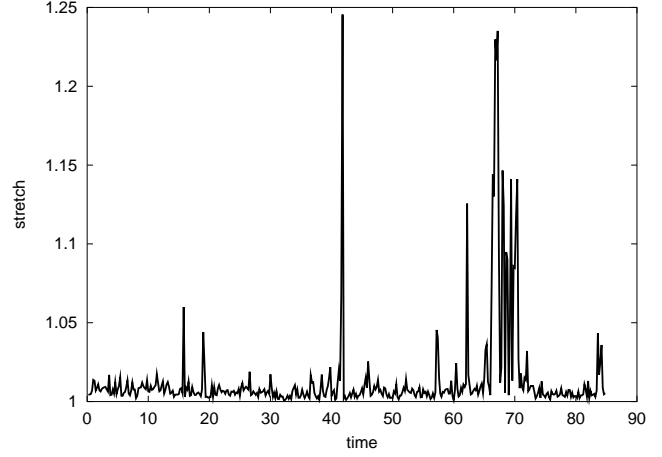
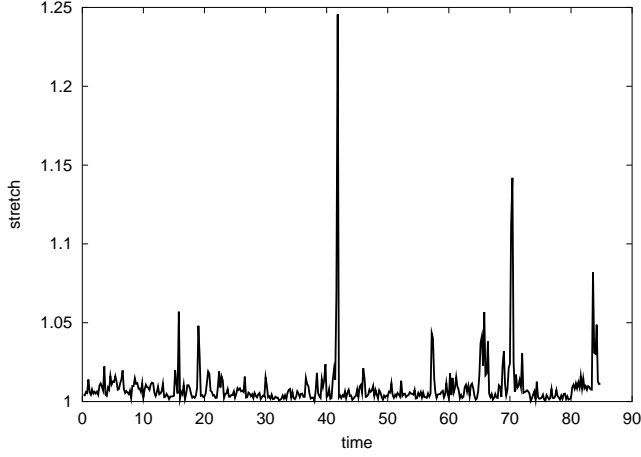


Fig. 5. Stretch with global update interval of: (a) 10 sec; (b) 30 sec;

term cost, is propagated only locally. This contrasts with our earlier formulation of the problem where it is assumed that the state of a link is either good or bad, i.e., when it is in good state, it has a finite cost representing its long-term throughput, whereas its cost is infinity when it is in bad state. Here, we propose an extension to BAFL referred to as *fine-grain blacklist-aided forwarding* (FBAF) that can deal with more fine grain variations in link quality. In the following, we discuss the implementation FBAF and evaluate its performance.

#### A. Implementation

The main difference between BAFL and FBAF is that along with a blacklisted link, its current short-term cost is also included under FBAF. We introduce some additional notation to help describe the operation of FBAF. As before,  $\tilde{\mathcal{E}}$  denotes the set of all edges, and  $\tilde{c}_e$ , the long-term cost of an edge  $e$  according to the most recent global update. Due to causes such as interference, the short-term cost of an edge may be worse than its long-term cost in which case it is considered *disrupted* and it may be blacklisted. The short-term cost of an edge  $e$  according to a node  $i$  is denoted by  $c_e^i$ . Among the set of all edges  $\tilde{\mathcal{E}}$ , the set of edges (both non-adjacent and adjacent) that are known to  $i$  to be in the disrupted state are denoted by  $\tilde{\mathcal{B}}_i$ . We use  $\mathcal{P}_{i \rightarrow d}(\mathcal{E}, c)$  to refer to the shortest path from  $i$  to  $d$  given the set of edges  $\mathcal{E}$  and their costs  $c$ . Similarly, the cost of the shortest path is denoted by  $C_{i \rightarrow d}(\mathcal{E}, c)$ .

Under FBAF, an edge  $e$  may be blacklisted, by a node  $i$ , if its current cost  $c_e^i$ , according to  $i$ , is worse than globally updated cost  $\tilde{c}$  resulting in the selection of an alternate next hop. The procedure for computation of a packet's blacklist and its next hop are shown in Alg. 5. We first compute the shortest path based on globally known set of costs  $\tilde{c}$  (lines 1-2). If it contains any of the blacklisted edges, they are included in the packet's blacklist (line 5). Also a new path is computed again after revising the cost of those blacklisted edges (line 4). This process is repeated till no new edges are added to the packet's

---

#### Alg 5 : Fine-grain Blacklist Aided Forwarding: FBAF( $i, p$ )

---

```

1:  $c' \leftarrow \tilde{c}$ 
2: while  $((\mathcal{P}_{i \rightarrow p, dest}(\tilde{\mathcal{E}}, c') \cap \tilde{\mathcal{B}}_i) \setminus p.blist) \neq \emptyset$  do
3:   for all  $b \in ((\mathcal{P}_{i \rightarrow p, dest}(\tilde{\mathcal{E}}, c') \cap \tilde{\mathcal{B}}_i) \setminus p.blist)$  do
4:      $c_b' \leftarrow c_b^i$ 
5:      $p.blist \leftarrow p.blist \cup b$ 
6:    $j \leftarrow \text{next}(\mathcal{P}_{i \rightarrow p, dest}(\tilde{\mathcal{E}}, c'))$ 
7:   if  $j \neq \emptyset$  then
8:     if  $p.blist \neq \emptyset$  then
9:        $k \leftarrow \text{lastdeadend}(p, dest, p.blist, \tilde{\mathcal{E}}, \tilde{c})$ 
10:      if  $C_{j \rightarrow p, dest}(\tilde{\mathcal{E}}, \tilde{c}) < C_{k \rightarrow p, dest}(\tilde{\mathcal{E}}, \tilde{c})$  then
11:         $p.blist \leftarrow \emptyset$ 
12: return  $j$ 

```

---

blacklist. The rest of the procedure is similar to BAFL.

---

#### Alg 6 : Processing of a packet under FBAF: $\text{recv}(i, p)$

---

```

1: for all  $b \in p.blist$  do
2:   if  $b.time > t_b^i$  then
3:      $t_b^i \leftarrow b.time$ 
4:      $c_b^i \leftarrow b.cost$ 
5:      $\tilde{\mathcal{B}}_i \leftarrow \tilde{\mathcal{B}}_i \cup b$ 
6:    $j \leftarrow \text{FBAF}(i, p)$ 
7:    $\text{send}(j, p)$ 

```

---

One of the details not mentioned above is that a timestamp  $t_e^i$  is associated with each link  $e$  at node  $i$ . This timestamp reflects the last time that link's state is measured and updated either globally or locally through a packet's blacklist. It is assumed that this value is monotonically increasing. A larger timestamp indicates more recent state of a link. Whenever a link  $b$  is included in the blacklist of a packet by a node  $i$ , both its timestamp  $t_b^i$  and its current cost  $c_b^i$  are also included. We use the notation  $b.time$  to refer to the timestamp of a blacklisted link and  $b.cost$ , its corresponding cost. The processing of a packet under FBAF is shown in Alg. 6. Based on the packet's blacklist, we update the blacklist cache at node  $i$ . Only if the timestamp of a link in the packet's blacklist is

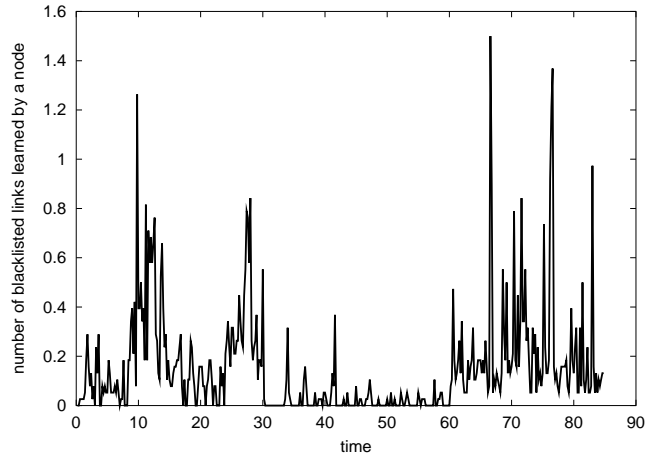
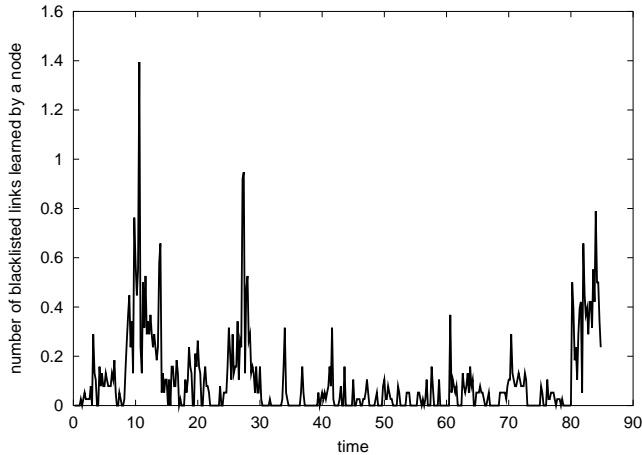


Fig. 6. No. of blacklisted links learned by a node with: (c) 10 sec; (d) 30 sec;

larger, the blacklist at node  $i$  is updated. The FBAF procedure is then used to compute the blacklist and next hop for the packet, and it is forwarded to that next hop.

### B. Evaluation

The performance of FBAF is evaluated using the link-level measurements data from MIT Roofnet project [1]. The Roofnet is a 38-node multi-hop wireless mesh network spread across approximately six square kilometers where each node consists of a PC with an 802.11b card connected to an omni-directional antenna mounted on the roof. The corresponding measurement trace records a delivery ratio for each of the 352 uni-directional links every 200 ms for 90 sec. A detailed description of the measurement setup and data collected can be found in [19]. It is observed that this network has many intermediate quality links and thus making it suitable for evaluating FBAF.

The FBAF formulation is based on the notion of short-term cost and long-term cost which are determined in our evaluation setting as follows. Based on the forward delivery ratio  $f$  (of link  $i \rightarrow j$ ) and backward delivery ratio  $b$  (of link  $j \rightarrow i$ ) given by the trace, the short-term cost or ETX<sup>3</sup> of link  $i-j$  is computed as  $\frac{1}{fb}$  [11]. The short-term cost of a link is determined every 200 ms based on the corresponding delivery ratios. The long-term cost of a link is computed as the average ETX since the last global update. The global update interval under FBAF is set to 10 sec or 30 sec.

To evaluate the optimality of FBAF, we measure the stretch under FBAF w.r.t the optimal routing. Fig. 5(a) and Fig. 5(b) show the average stretch among all the node pairs under FBAF 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 FBAF, we plot the number of blacklisted links learned by a node in Fig. 6(a) and Fig. 6(b). The average size of a blacklist maintained by

a node under FBAF in both cases is insignificant considering that there are 308 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 FBAF. Once again these results illustrate the efficacy of localized on-demand link state propagation effected by the BAF approach.

### VII. RELATED WORK

Many routing schemes are conceivable for multihop networks with static nodes and dynamic links. An ideal scheme delivers packets to destinations (reliable), along the shortest paths (optimal), with minimal overhead (scalable). In this section, we discuss various possible routing alternatives and argue that link state routing with blacklist-aided forwarding is the most suitable one for static multihop wireless networks.

There have been several proposals for making link state routing scale for ad hoc networks. They are categorized into *efficient dissemination* approaches and *limited dissemination* approaches [20]. OLSR [16] is an efficient dissemination based approach that propagates updates through the entire network but more efficiently than traditional flooding. Fisheye State Routing (FSR) [12] and Hazy Sighted Link State (HSLs) routing [20] are limited dissemination based schemes that update the nearby nodes at a higher frequency than the remote nodes that lie outside a certain scope which is static and therefore could be more than sufficient in some cases and less than necessary in other cases for ensuring loop-free packet delivery. Blacklist-aided forwarding can be categorized as a limited dissemination approach, but unlike FSR and HSLs it effectively notifies only the nodes in the vicinity of a disrupted link that need to be informed and delivers reliably despite multiple simultaneous disruptions.

It is interesting to contrast ideas of blacklist-aided forwarding with similar ideas in other schemes. Blacklist-aided forwarding works along the similar lines of position based forwarding [6] and can be thought to switch between greedy and recovery modes. But it does not have the same deficiencies

<sup>3</sup>Since ETX is defined only for bidirectional links, we discard unidirectional links such as  $i \rightarrow j$  with no corresponding  $j \rightarrow i$ . A total of 44 such unidirectional links exist and our experiments include the rest 308 links.

as position based forwarding in terms of optimality and reliability. Blacklist aided forwarding is similar in spirit but opposite in effect to loose source routing. While a packet under loose source routing carries a list of nodes that *must be traversed*, under blacklist aided forwarding it contains a list of links that *must not be traversed*. Also, a packet's loose source route is determined at its source while its blacklist is updated during the flight to its destination. It is suggested that DSR [9] can benefit from caching "negative" information about links that are currently providing highly "variable" service. Also, DSR allows backtracking but prevents a packet from being salvaged more than once. Our approach lets each packet explicitly carry a blacklist and allows both backtracking and salvaging multiple times during a packet's flight.

Link Quality Source Routing (LQSR) is scheme proposed recently [21] specifically for static multihop wireless networks. LQSR is based on DSR but uses a link cache instead of route cache and is essentially a link state routing protocol. Under LQSR, each packet carries the source route and intermediate node updates the source route with the current metric for outgoing link. The receiver has to send either a gratuitous reply back to the source or piggyback it on data packet in which case it effectively carries two source routes. In addition, LQSR uses a proactive background mechanism to maintain the metrics of all links. This is done by piggybacking the link info on route requests and sending a dummy route request message. In contrast under BAF, forwarding is done hop by hop and each packet carries only the blacklist in addition to the destination address.

Blacklist-aided forwarding can be categorized as on-demand table-driven link-state routing scheme. The advantages and disadvantages of table-driven, pro-active, link-state routing for mobile ad-hoc networks are well studied [4]. They avoid the route discovery latency at the expense of route maintenance overhead. It is said that proactive routing protocols suffer the disadvantage of repairing a broken route even though no applications are using them. That would not be the case with blacklist-aided forwarding. When a failed link were not to be traversed by a packet, that failed link would not be blacklisted and doesn't cause any overhead under blacklist-aided forwarding.

## VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we focused on the design of a scalable scheme for reliable delivery in static multihop wireless networks with frequent disruptions. We proposed a link-state-based blacklist-aided-forwarding approach that provides loop-free delivery despite disruptions through on-demand propagation of information on disruptions via blacklists carried in data packets while taking advantage of static adjacencies of nodes. BAF guarantees delivery of packets to all reachable destinations irrespective of the extent of disruptions. We have evaluated BAF and demonstrated that it is not only reliable but also near-optimal and highly-scalable. We have also shown that BAF approach can be extended to effectively handle link quality variations at a finer scale than just good and bad.

The main limitation of this paper however is that we have only presented preliminary evaluation results of BAF to demonstrate its features. We have not compared its performance against similar schemes such as LQSR [21] and HSLs [20]. We are currently simulating these schemes using NS2 [22] and performing a thorough evaluation of BAF and other schemes that are targeted for static multihop wireless networks. We will also be implementing BAF and conducting real-world experiments to further bolster the case of BAF.

## REFERENCES

- [1] MIT Roofnet, "<http://www.pdos.lcs.mit.edu/roofnet/>."
- [2] Mesh Networks, "Mesh networks technology overview," <http://www.meshnetworks.com>.
- [3] Nokia Communications, "Nokia RoofTop Wireless Routing," White Paper, Apr. 2002.
- [4] E. Royer and C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications Magazine*, vol. 6, no. 3, pp. 46–55, Apr. 1999.
- [5] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 1, pp. 1–22, Jan. 2004.
- [6] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network Magazine*, vol. 15, no. 6, pp. 30–39, Nov. 2001.
- [7] Richard Draves, Jitendra Padhye, and Brian Zill, "Comparison of routing metrics for static multi-hop wireless networks," in *Proc. ACM Sigcomm*, 2004.
- [8] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proc. ACM Mobicom*, 2004.
- [9] D. Johnson, D. Maltz, and J. Broch, "DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks," in *Adhoc Networking*, Charles E. Perkins, Ed. 2001, pp. 139–172, Addison-Wesley.
- [10] Omprakash Gnawali, Mark Yarvis, John Heidemann, and Ramesh Govindan, "Interaction of retransmission, blacklisting, and routing metrics for reliability in sensor network routing," in *Proceedings of the First IEEE Conference on Sensor and Adhoc Communication and Networks*, Santa Clara, California, USA, October 2004, pp. 34–43, IEEE.
- [11] Douglas De Couto, Daniel Aguayo, John Bicket, and Robert Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. ACM Mobicom*, 2003.
- [12] M. Gerla, X. Hong, and G. Pei, "Fish-eye state routing protocol for ad hoc networks," IETF Internet Draft, June 2002, draft-ietf-manet-fsr-03.txt.
- [13] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, "Local restoration algorithms for link-state routing protocols," in *ICCCN*, 1999.
- [14] Anindya Basu, Alvin Lin, and Sharad Ramanathan, "Routing using potentials: A dynamic traffic-aware routing algorithm," in *Proc. ACM Sigcomm*, Aug. 2003.
- [15] Sanghwan Lee, Yinzhe Yu, Srihari Nelakuditi, Zhi-Li Zhang, and Chen-Nee Chuah, "Proactive vs Reactive Approaches to Failure Resilient Routing," in *Proc. IEEE Infocom*, Hong Kong, 2004.
- [16] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Oct. 2003.
- [17] Sundar Iyer, Supratik Bhattacharyya, Nina Taft, and Christophe Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proc. IEEE Infocom*, Mar. 2003.
- [18] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Junling Wang, Z. Zhong, G.-H. Lu, and Zhi-Li Zhang, "Blacklist-Aided Forwarding in Static Multihop Wireless Networks," Tech. Rep. TR-2005-002, University of South Carolina, 2005.
- [19] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *Proc. ACM Sigcomm*, 2004.
- [20] C. Santivanez, R. Ramanathan, and I. Stavrakakis, "Making link-state routing scale for ad hoc networks," in *Proc. ACM Mobicom*, 2001.
- [21] Richard Draves, Jitendra Padhye, and Brian Zill, "Comparison of routing metrics for static multi-hop wireless networks," Tech. Rep. MSR-TR-2004-18, Microsoft Research, 2004.
- [22] NS2, "<http://www.isi.edu/nsnam/ns/>."