SHARE: Scalable Hybrid Adaptive Routing for dynamic multi-hop Environments

Victoria Manfredi
Department of Mathematics and Computer Science
Wesleyan University, Middletown, CT 06459
vumanfredi@wesleyan.edu

Ram Ramanathan, Will Tetteh, Regina Hain, and Dorene Ryder
Raytheon BBN Technologies
10 Moulton Street, Cambridge, MA 02138
{ram.ramanathan,will.tettah,regina.hain,dorene.ryder}@raytheon.com

***Abstract*—No longer are wireless networks isolated islands: now devices in one network may communicate with devices in other networks. But as the density of devices increases and the range of wireless transmissions decreases, there is a critical need for *scalable* multi-hop wireless networking. In this paper, we propose SHARE, a novel approach to scalable multi-hop routing for wireless networks. SHARE is a *hybrid* of gradient routing, locally scoped link-state routing, multiple-path braid forwarding, and flooding. SHARE adapts the amount of control traffic and the amount of data traffic according to data traffic patterns and network dynamics to maximize network scalability and robustness while minimizing control overhead. We evaluate a real-world implementation of SHARE in ns-3 using Direct Code Execution (DCE), and compare SHARE against a DCE version of the optimized link state routing protocol (OLSR) for a static grid and for the Gauss-Markov mobility model. We show that SHARE is able to deliver up to 31% more data packets than OLSR across a range of network sizes and speeds, while reducing overhead by a factor of $7\times$.**

I. Introduction

The number and density of wireless networked devices (e.g. projections of 20-50 billion by year 2020 [6]) are rapidly increasing, while an increasing appetite for higher data rates is pushing wireless technologies (e.g. 5G) towards higher frequency bands (e.g. mmWave) [8], [17] that have shorter communication ranges. Shorter ranges are also characteristic of Internet of Things (IoT) devices due to their low-power requirements. In parallel, wireless networks are evolving from isolated homogeneous groups of mostly stationary devices to heterogeneous groups of overlapping and interacting stationary and mobile devices [12], including WiFi-cellular device networks [9], vehicular sensor networks [1], [5], wearable health devices synchronizing with cell phones [24], and microclouds and cloudlets for sharing computational power [31], [33].

This confluence of increasing device density, device communication across different networks, and shorter transmission ranges motivates two important features in wireless network design: *multi-hop* routing and *scalability*. Indeed, the IETF working group Routing Over Low power and Lossy networks (ROLL) calls out multi-hop scalability requirements of several 1000's of nodes depending on the deployment setting, while visions for 5G (e.g. [23]) call out scalability as a key requirement. Similarly, military networks have long sought but have yet to achieve scalable multi-hop wireless networking for brigade-sized (1000's of nodes) networks [27].

In this paper, we propose a new approach to scalable multi-hop routing for wireless networks. Our work is applicable to any mobile or dynamic network environment where a packet may be relayed over multiple wireless hops to reach its destination(s). Such networks encompass the well-known *ad hoc* networks, sensor networks, and mesh networks as well as emerging wireless networks of drones, smartphones, vehicles, and even balloons in the stratosphere [4]. Our protocol, **Scalable Hybrid Adaptive Routing for dynamic multi-hop Environments (SHARE)**, leverages two key ideas: (1) data traffic patterns should determine the amount and type of routing state collected, and (2) redundant packet transmissions are important for accommodating network dynamics. Using these ideas, SHARE adapts both the amount of control traffic and the amount of data traffic according to the traffic and network dynamics to maximize network scalability and robustness while minimizing overhead.

When the topology is relatively stable, SHARE forwards locally destined data traffic using proactive local link state routing and forwards non-locally destined data traffic using reactive gradient routing. To accommodate instabilities in the network topology, both the link state and the gradient state provide alternative routes for data traffic, allowing SHARE to deliver data traffic even when the network topology is dynamic. To provide additional redundancy for forwarding of data packets in dynamic networks, or when there is some potentially out-of-date state, SHARE uses scoped flooding of data packets over a *braid* around the shortest path on a per-flow basis [18], [14], [34], [20]. How wide to scope the braid is a configurable variable. In the absence of any link state or gradient state, such as in a highly dynamic network, SHARE seamlessly defaults to flooding of data packets, ensuring that traffic is delivered even when no routes exist. This flood, however, is not unconstrained: data packets are only flooded until they reach a node that contains gradient state or link state, or they reach the destination.

We implement SHARE in ns-3 using Direct Code Execution

This material is based upon work supported by DARPA and SPAWARSYSCEN Atlantic under Contract No. N65236-14-C-2816. Any opinions, findings and conclusions or recommendations expessed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or SPAWARSYSCEN Atlantic. Distribution Statement A: Approved for Public Release, Distribution Unlimited, Case 27316.

(DCE), and evaluate SHARE against a DCE version of OLSR for a stationary grid network and for a network with Gauss-Markov mobility. Our use of DCE means that our evaluation is based on real-world protocol implementations, not models: DCE code can be directly run in real hardware, unlike a non-DCE implementation (the ns-3 default). We study the delivery ratio and control overhead of SHARE and OLSR across a range of network sizes, node speeds (for the Gauss-Markov mobility model), and number of destinations. In the range of network sizes and speeds studied, SHARE delivers up to 31% more data packets than OLSR in a static grid, and up to 28% more data packets than OLSR in a mobile network. SHARE reduces control overhead by a factor of 7× over OLSR.

The rest of this paper is organized as follows. In Section II, we overview related work. Section III gives an overview of SHARE while Sections IV and V describes SHARE features in more depth. Section VI presents our ns-3 results and Section VII describes our conclusions.

II. Related Work

Proactive link state routing protocols (e.g. OLSR [13]) attempt to keep accurate global state at the cost of high control overhead, particularly when nodes are mobile, and risk unusable routing state at nodes. Protocols such as Hazy-sighted Link State (HSLS) [29] limit control traffic by scoping the number of hops over which control travels, or, as in Weak State Routing [7], support quantifiable routing state uncertainty. However, such protocols do so purely topologically, and do not handle situations where only a few nodes are sources or destinations, and are therefore fundamentally limited in their abillity to scale. Reactive (on-demand) routing protocols such as AODV [25] build routes only as required. However, if traffic patterns are non-uniform and some portions of the network have many flows, this incurs excessive control traffic.

Previous work has shown that there are regimes in which proactive routing does much better than reactive routing, giving rise to work on hybrid proactive-reactive routing protocols such as Zone Routing [11] which uses zones to determine whether to use proactive or reactive routing: destinations within the same zone use proactive routing while destinations outside the zone of the source use reactive routing. In comparison, SHARE's key contribution is a hybrid protocol unifying several ideas: proactive link state routing for local traffic, reactive gradient routing for non-local traffic, flooding when no gradient or link state is found, replication ("braiding") of data packets for robustness against topology changes, and a novel probabilistic truncation of gradient control flooding to reduce overhead.

Unlike gradient routing work in wireless mesh and sensor networks [15], [14], [34], [16], SHARE supports node mobility by providing multiple fallback routing mechanisms to accommodate topology changes, finally resorting to network-wide flooding if no route is found, but switching back to unicast forwarding as soon as state is found. In ad hoc networks, prior work on gradient routing [26] uses sender triggered gradient creation, whereas SHARE triggers gradient establishement by the destination, when it receives packets destined to it.

Some work in sensor networks on gradients [14], [34] leverages the alternative paths established by the gradient to forward data traffic over multiple paths braided together rather than a single path. While routing protocols for wireless mobile ad hoc networks typically provide only one route to a destination, resulting in poor delivery ratio in an intermittent network, some work has explored providing alternative paths or forwarding over a braid of multiple paths [18], [20]. Like these works, SHARE's gradient approach provides many alternative routes if a link breaks. But by differentiating local link-state control from network-wide gradient control, SHARE additionally incurs lower overhead and allows higher fidelity local state and lower fidelity global state. By additionally using braids and allowing flooding to be used when no routing state exists or packet time to live (TTL) has been exceeded, SHARE is less sensitive to faulty or non-existent state and more resilient to link breakages and mobility.

To scope flooding of gradient control packets, SHARE uses density-adaptive truncated flooding, a novel probabilistic approach based on node density to decide whether to retransmit a gradient control message, and which leverages the local link state that SHARE maintains. This is similar in goal to the Multi-Point Relays (MPRs) used by OLSR, but does not require coordination among nodes. Other approaches to probabilistically reduce floods include rebroadcasting with fixed probability [30] as well as additionally using counters to keep track of the number of times a packet has been seen [21]. In comparison to these schemes, SHARE nodes probabilistically choose to retransmit based on the probability that 1-hop neighbors will receive the packet given the local link state knowledge of node connectivity. SHARE is most similar to hybrid flooding [28] which probabilistically retransmits based on network density. SHARE extends this idea by explicitly taking the topology into consideration, as SHARE has access to the local topology from its local link state.

III. SHARE Overview

The goal of SHARE is to maximize network scalability and robustness to topology changes while minimizing control overhead. To do so, SHARE adapts the amount of control overhead and the data forwarding strategy to the traffic characteristics and network dynamics. We can thus categorize SHARE behaviour as a function of topology stability and traffic locality, with additional SHARE features coming into play as the topology becomes less stable.

A. High stability

Local routes (i.e., to destination nodes within two hops of the source node) can typically be setup more efficiently than non-local routes. In comparison, non-local routes require more control overhead to setup, but we expect that there will be significantly less non-local traffic than local traffic [27] and frequently such non-local destinations serve as sinks (e.g.,

as in sensor networks). Consequently, there are benefits to separating how local and non-local data traffic is handled.

SHARE proactively uses local control traffic to maintain high fidelity 2-hop link state information for routing *locally destined data traffic*, in accordance with the properties of real-world network traffic which typically falls off with hop-distance in a power-law fashion [27], [19].

SHARE reactively uses on-demand gradients to forward *non-locally destined data traffic* (more than 2 hops), and to provide alternative routes for data traffic. By having a gradient established for non-local destinations, all nodes will know of some path, and the alternative paths in the gradient provide robustness against topology changes and path breaks that are more likely as the distance between source and destination increase. Such gradients form destination-rooted sink trees. SHARE constrains the amount of gradient control traffic by (1) adapting the rate at which nodes send gradient updates about themselves depending on whether they are receiving traffic destined to them (thereby adapting the fidelity of the gradient state) and (2) truncating the scope of the network-wide floods of gradient messages using probabilistic, density-adaptive truncated flooding.

B. Medium stability

Both the link state and the gradient state provide alternative routes for data traffic, allowing SHARE to deliver data traffic even when the network topology is dynamic.

To provide additional redundancy for forwarding of data packets in dynamic networks, or when there is some potentially out-of-date state, SHARE uses scoped flooding of data packets over a *braid* around the shortest path on a per-flow basis [18], [14], [34], [20] and takes advantage of the alternative routes found in the link and gradient state. How wide to scope the braid is a configurable variable. In dynamic environments, braiding increases delivery probabilities by providing robustness to broken links via redundant data packet transmissions.

C. Low stability

In the absence of any link state or gradient state being found during the forwarding process, such as in a highly dynamic network, SHARE seamlessly defaults to network-wide flooding of data packets. Data packets are only flooded until they reach a node that contains gradient state or link state, or they reach the destination. This ensures that data packets are delivered even when no link or gradient state exists, while also scoping the flood of data packets to only what is needed.

IV. Route Establishment

We first describe the types of control packets that SHARE uses, and then describe SHARE's hybrid link-state and gradient routing approach.

A. SHARE Control Packets

To establish the link state and the gradient state, SHARE uses two kinds of control packets, *Gradient Establishment Messages (GEMs)* and *Heartbeats*, within the single unified message format shown in Table I which is used for all SHARE packets, both control and data. The additional packet header used on GEM and Heartbeat packets is shown in Table II.

TABLE I
SHARE packet header.

Field	Purpose
uint16_t type	Packet type: data or control
uint16_t src	Packet source
uint16_t dest	Packet destination
uint16_t len	Packet length
uint16_t crc	Cyclic redundancy check
uint32_t seq	Packet sequence number
uint16_t ttl	Packet time-to-live

TABLE II
Additional header fields for GEMs and Heartbeats.

Field	Purpose
uint32_t gemId	GEM ID
uint32_t cost	GEM cost
uint32_t numNbrs (Heartbeats only)	Number of neighbors
uint32_t nbrs[] (Heartbeats only)	List of neighbor addresses

TABLE III
Additional header fields for data packets.

Field	Purpose
uint64 hops1	Bitarray for node IDs from 0:63
uint64 hops2	Bitarray for node IDs from 64:127
uint64 hops3	Bitarray for node IDs from 128:191
uint8_t braid	Flag indicating whether to braid
uint8_t count	Braid duplication rate (k)
uint16_t flow	Packet flow number

GEMs. These control packets are global in scope and are used to establish the gradient state. They are retransmitted by all receiving nodes. GEM IDs are used to prevent retransmissions of the same GEM by a node. The frequency with which a node sources these GEMs depends on whether the node is receiving data destined to itself.

Heartbeats. These control packets are local in scope and are used to establish the link state. They are transmitted only one hop and are not re-tranmitted when received. The Heartbeats contain neighbor lists which are used to compute the 2-hop local neighborhood. A node sources these packets according to the heartbeat interval.

B. Proactive Local Link State Routing

SHARE proactively establishes 2-hop link state to route data packets with local destinations. By scoping the control to two hops, SHARE ensures that high-density regions are not overwhelmed with control traffic.

To establish the 2-hop link state, nodes must first discover all nodes within their 2-hop neighborhood. To do this SHARE uses the Heartbeat control packets. As in Table II, Heartbeats contain the complete neighbor list for a node, and are transmitted exactly once. The desired frequency for advertising Heartbeats is a configurable parameter. Nodes must maintain

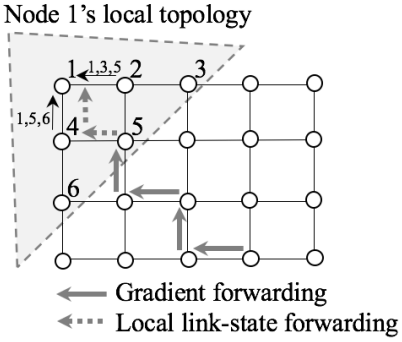


Fig. 1. Each node advertises a list of its neighbors periodically. This list is not retransmitted. Thus, each node knows its complete 2-hop topology, including nodes, links, and costs. GEMs are only triggered at a destination if the source is outside of the 2-hop neighborhood. When network diameter is ≤ 2, no GEMs are needed, giving $\Theta(1)$ overhead.

a neighbor table and actively track this frequency in order to expunge expired neighbor list information. Our current implementation of the neighbor lists found in a Heartbeat lists uses actual node IDs. It is possible to instead optimize the neighbor list with a more compact representation of node IDs.

Upon receiving neighbor lists from all neighbors, a node reconstructs the 2-hop neighborhood graph and runs Dijkstra's algorithm on the graph to determine the shortest-path between any node pair in the neighborhood. For every destination in the neighborhood, a unicast routing table is populated with the appropriate next-hop.

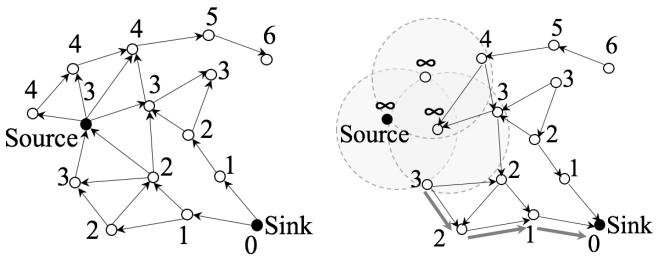
C. Reactive Gradient Routing

SHARE uses gradients to route data traffic with non-local destinations. Evidence shows network traffic falling off with hop-distance in a power-law fashion [19], [27], indicating that most data traffic would likely be routed using the local link state and only a small portion of the data traffic would be routed using the non-local gradient state. Consequently, we expect that gradient establishment and the concomitant flooding of GEMs should occur infrequently. SHARE's gradient-based approach also reduces dependence on state correctness, allows the use of asymmetric links (unlike AODV and DSR), and provides ready-made backup routes.

A gradient is state at a node indicating the relative cost to a destination. SHARE uses hop-count as the gradient cost measure, although other meaures could be used, such as energy in the case of sensor networks. To establish a gradient, a node broadcasts GEMs with an initial hop-count of 0 and a GEM ID which indicates when a GEM was generated. On receiving a GEM, a node records and increments the hop-count. A node re-broadcasts a GEM at most once, and only if the GEM contains the most recent GEM ID and lowest hop-count seen so far from the GEM source. Nodes do, however, record all GEMs received in their gradient tables. Once a GEM has been successfully disseminated into the network, each node constructs a gradient table to the destination.

TABLE IV
FOR TOPOLOGY IN FIGURE 1, AN EXAMPLE GRADIENT TABLE AT NODE 2 WITH NODE 1 AS THE DESTINATION.

Destination	Next Hop	Cost	GEM ID	Time
Node 1	Node 1	0	1	10332
	Node 3	2	1	10784
	Node 5	2	1	10986



(a) Gradient establisment (b) Gradient seeking

Fig. 2. Building and using SHARE gradients.

Table IV shows an example gradient table from the viewpoint of Node 2 in Figure 1 for Node 2 knows how to get to Node 1 via all neighbors, including Node 3 whose next hop is Node 2.

SHARE uses an adaptive rate at which to send GEMs, rather than a fixed rate. There are two frequencies at which GEMs are sent: a low background frequency at which all nodes send GEMs, and a higher, "normal" frequency that is triggered as long as a node receives data packets destined for itself and the source of the data packet is not in the destination's 2-hop link state. A node reduces its GEM rate when it has received no data traffic destined to itself for a threshold period of time.

When a node has data packets to send, those packets are iteratively forwarded to nodes with lower gradient costs, like water flowing downhill. A node may or may not have a gradient to a destination; for example, if a destination was sending GEMs too infrequently or GEMs were lost due to mobility or jamming. In this case, the data packet is flooded until it finds gradient or link state, or reaches the destination. The destination then invokes a higher rate of sending GEMs. Figure 2(a) illustrates the process of establishing a gradient from a sink node to a source node, while Figure 2(b) illustrates the process of another source node seeking out the gradient to the original sink node.

To limit GEM overhead, SHARE probabilistically truncates network-wide GEM floods based on node density. We do not use Multi-Point Relays (MPRs) as in OLSR as they are too rigid and risky in sparse networks, and instead present a novel distributed *density adaptive truncated flooding* algorithm. A node decides whether to re-transmit a GEM using the probability that all of its neighbors will receive at least C GEMs (where C can be thought of as a measure of redundancy or number of alternate paths desired). This probability is computed directly from the local link state information, as described below. By probabilistically truncating GEM floods, SHARE intelligently reduces the effective degree of the network thereby reducing

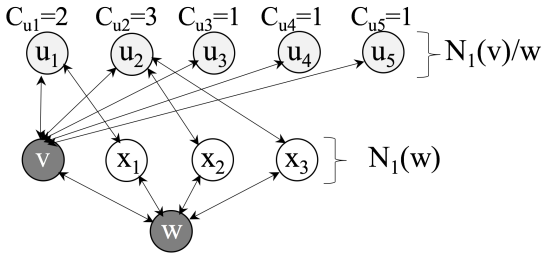


Fig. 3. Scoping a GEM flood. Node v retransmits a GEM based on the number of w's 1-hop neighbors that cover each of v's 1-hop neighbors, excluding w.

the bandwidth incurred by flooded messages.

From the link state information, each node knows its 1-hop and 2-hop neighborhoods. Therefore, if a node receives a broadcast packet, it also knows which of its neighbors should have received the packet. Suppose a node receives a GEM: to determine whether to retransmit the GEM, the node computes the minimum probability that its 1-hop neighbors will receive the GEM if it does not retransmit. Define the 1-hop neighborhood as the following set.

$$N_1(x) = \{y| \text{ there is a path of length 1 between x and y}\}$$

Let C be the desired number of nodes "covering" a 1-hop neighbor. Let v be the "considered" node, that is, the node for which to compute the re-transmission probability. Let w be the node from which $v \in N_1(w)$ received the packet. From the link state, v knows the set $N_1(w)$. For each 1-hop neighbor $u \in N_1(v) \backslash w$, let C_u be the set of 1-hop neighbors of w (other than v) that cover u, that is, $C_u = \{x | x \in N_1(w) \setminus v \text{ and } u \in N_1(x)\}$. Let C_{min} be the smallest such set over all u. Our retransmission must cater to the node that is least covered by nodes with the packet from w. Then the probability that node v retransmits, $p_R(v)$ is computed as follows.

$$p_R(v) = \begin{cases} 1 & \text{if } C \geq |C_{min}| \\ C/|C_{min}| & \text{otherwise} \end{cases}$$

We observe that the case $C \geq |C_{min}|$ might occur because there are insufficient nodes to cover the 1-hop neighbor, or because the link-state tables have not yet converged or are out of date. Regardless, the correct behavior is still to retransmit the GEM with $p_R(v) = 1$.

C controls the extent to which a packet is flooded. Scoping GEM floods has the effect of controlling the number of alternate paths of which a node is aware. However, the probabilistic nature of the retransmissions ensures that from one GEM flood to the next, different nodes will probabilistically respond, ensuring in expectation that over multiple floods that all available alternate paths will be found.

Thus, not all nodes re-transmit a received GEM. The set of forwarders is controlled so that every node gets the GEM from sufficient (e.g. 3 to 4) neighbors so that they have some alternate paths, but not from every neighbor.

V. Adaptive Data Redundancy

Wireless multi-hop networks are inherently dynamic and heterogeneous. Node density and link breakages in a network change temporally and spatially. Traffic traveling through a high mobility region of a network is more likely to get dropped than traffic traveling through a low mobility region. Traffic travelling many hops to a destination is more likely to get dropped than traffic travelling just a few hops. Traffic travelling through a highly congested region of a network is more likely to get dropped than traffic traveling through a a lightly loaded region. To accommodate these disparate reliability needs, SHARE provides multiple layers of redundancy, and the ability to adapt the redundancy to data traffic needs and network conditions.

We first overview how SHARE forwards data traffic, and then describe how additional redundancy is added.

A. SHARE Data Forwarding

The process that a node goes through to decide how to forward a data packet is as follows. These forwarding steps are repeated at each downstream node until the data packet reaches its destination.

1) *Check link state table.* A node first checks whether the destination of the packet is present in its local link state table. If the destination is present, the node uses the specified next hop to forward the packet.
2) *Check gradient table.* If the node does not find the destination in its link state table, the node checks whether the destination is present in its gradient table. If the destination is present, the node uses the specified next hop to forward the packet.
3) *Flood.* If the destination is not present in either the link state table or gradient table, or the packet TTL has reached zero, the node floods the packet, setting the next hop to be the broadcast address.

B. Braiding

The first mechanism that SHARE uses to provide redundancy to data traffic is a form of scoped flooding called braiding [18], [14], [34], [20]. Braiding provides controlled replication of data packets over a "'braid" around the shortest path. A braid is a subgraph of the network graph that includes the shortest path between the source and destination and all nodes and links within k-hops of the shortest path. The width of the braid can be adjusted based on the importance of the data being transmitted: for a maximally important packet, the braid defaults to a network-wide flood.

To braid a data packet using SHARE, a node sets four fields in the packet header (see Tables I and III).

1) *Braid flag*. 0 indicates the packet should not be braided. 1 indicates the packet should be braided by both the source and downstream nodes. 2 indicates that only the source should braid the packet and non-source nodes should just forward the packet on.
2) *Braid branching factor, k.* k is the number of next hop nodes that should forward the data packet, and can range

from 1 (no braiding) to all possible next hops (the packet is simply flooded).

3) *Braid next hops*. This is a list of up to k addresses indicating the nodes that should form the next hops in the braid. To choose the next hops, a node first sorts all available next hops found in its link state and gradient state based on hop-count to the packet destination. The node then selects the k next hops with the lowest hop-count to the destination. The use of link state and gradient state makes braiding straightforward: nodes are already aware of alternative paths, braiding simply provides a mechanism for choosing how to use those paths. There is, however, an interplay between braiding and the density adaptive truncated flooding of GEMs braiding: if the truncated flooding is too restrictive, resulting in few or sub-optimal alternative paths, braiding will perform poorly having limited choices for next hops.
4) *Packet time-to-live (TTL)*. This constrains the flood of braid packets, by setting the initial packet TTL to the maximum desired path length, TTL_{Max}. Then a downstream node only forwards on the packet if its shortest path length to the destination is less than the packet's current TTL. By varying how TTL_{Max} is set, the scope of the flood forming the braid can be varied. Braiding will generate at most $k{\cdot}TTL_{Max}$ packets, since the packet transmissions can be viewed as forming a binary tree of height TTL_{Max}. Because nodes do not retransmit packets it has already seen, and because of the TTL throttling so that only paths that can reach the destination within TTL_{Max} are used, we expect that fewer than $k \cdot TTL_{Max}$ packets will be generated.

When a packet is braided, it is sent as a broadcast packet, not a unicast packet: consequently, 802.11 does not retransmit braided data packets, only non-braided data packets. While SHARE's braiding mechanism is currently sender-centric, braiding could also be made receiver-centric, allowing receivers to decide whether to forward, similar to SHARE's density adaptive truncated flooding approach.

C. Flooding

The second mechanism that SHARE uses to provide redundancy to data traffic is the ability to flood data packets. SHARE's ability to flood data packets serves multiple purposes. First, flooding enables a simple mechanism for network wide broadcast, enabling nodes to easily broadcast data to all other nodes in the network, when desired. Second, flooding, allows new nodes to easily join the network and seamlessly start forwarding data. Third, flooding provides redundancy for important transmissions, when desired. Finally, flooding provides a fallback mechanism when no route to the destination is known.

The main use of flooding in SHARE currently is as a fallback mechanism. SHARE triggers a flood for a data packet when the destination of a data packet is not found in the gradient or link state at a node. Flooding may evolve to follow a directed path using either local link state or gradient state if a route to the destination is found during the flood. Thus, packets are only flooded until they reach a node that contains gradient state or link state, or they reach the destination. Once a flooded packet reaches a node with state it is forwarded on using unicast forwarding. Once the packet reaches its destination, GEMs are triggered to be sent at a higher rate, and gradient will be established if connectivity exists. While nodes will drop duplicate packets, it is possible multiple copies of the same flooded packet all find gradient or link state but at different nodes. In this case, the destination may still receive duplicate packets.

The need for flooding should occur only rarely since gradient state is used as soon as it is found, and gradients are set up when no gradient state is found. The reach of network-wide broadcasting is scoped using a TTL value that determines the number of hops a network-wide broadcast packet may travel before being dropped. A TTL value of -1 is reserved and indicates the maximum number of hops possible. Because nodes check for duplicates before rebroadcasting packets, the flood of data packets will eventually terminate.

VI. SHARE Evaluation

SHARE was implemented in C++ and evaluated in ns-3.22 [3] using DCE [32]. The DCE framework allows the same network-layer code that would run on a hardware platform to be run over a channel and radio model in ns-3, and therefore provides very high fidelity compared to simply using a model of a protocol. Using DCE, however, does limit the size of the networks we are able to simulate on a single machine to about 70 nodes, due to the computational costs and memory requirements of performing such high fidelity simulations. Our simulation sizes are also limited due to the amount of traffic in the network: every node in the network communicates with at least one other node, resulting in correspondingly more flows as the network size increases.

We compare SHARE against OLSR [13]. Since DCE implemented code can be run on real hardware, such code is higher fidelity than non-DCE code and the differences can impact simulation performance. Thus, to eliminate discrepencies due to DCE, we use the Naval Resarch Laboratory's (NRL) version of OLSR implemented in ns-3 DCE [2]. We note that we found no implementations of AODV [25] or other multi-hop wireless routing protocols in DCE against which we could compare. Table V summarizes our simulation parameters for SHARE.

A. Traffic

We evaluate SHARE and OLSR performance on two traffic patterns: (1) all-to-one traffic and (2) all-to-some traffic, rather than random source-destination traffic, as these are the representative cases for target applications for SHARE, such as IoT and 5G. Following previous work (e.g. [22]), we set the packet size to 64 bytes. Our simulations are 180 seconds in length, with traffic starting at 30 seconds and ending at 150 seconds. The same traffic description file is used for both SHARE and OLSR, so that the exact same sequence of packets, sources and destinations is generated for both runs.

TABLE V
SHARE SIMULATION PARAMETERS.

Feature	Variable	Value
Gradient	Background GEM rate	1 / 40 s
	Non-background GEM rate	1 / s
	Truncated flooding C value	2
Link State	Heartbeat GEM interval	1 s
	Shortest path compute interval	500 ms
	Neighbor timeout period	5 s
Braiding	Value of braid branching factor k	2
	Initial TTL for braided packets	20

B. Mobility Models

We compare the performance of SHARE and OLSR on two mobility models, a stationary grid and the Gauss Markov mobility model. For both we use 802.11 with RangePropagation model, setting the transmission range to $R = 630$m. Using RangePropagation ensures that for the grid, the topology really conforms to a grid.

For the *grid*, nodes are stationary and arranged in an $\sqrt{N} \times \sqrt{N}$ grid, where N is the number of nodes. The dimensions of each grid square is $R \times R$.

For the *Gauss-Markov mobility model*, we generate mobility traces using BonnMotion [10] to ensure that both SHARE and OLSR simulation operated on the same network topology. We ignored the first 3600 seconds worth of movement generated by BonnMotion. Nodes are distributed within an area of size $M \times M$ where $M = \sqrt{NR^2\pi/D}$ where N is the number of nodes and $D = 10$ is the node density.

C. Results

Figures 4(a) and 4(d), show that SHARE has a higher delivery ratio than OLSR across a range of network sizes, delivering up to 31% and 23% more traffic for static and mobile networks respectively. In general, the difference increases as size or speed increases, except in the case of very small networks where OLSR exhibits surprisingly poor performance. The sharp drop for both OLSR and SHARE at 60 nodes in Figure 4(d) is likely due to the destination being in a disadvantaged position for the particular random network.

Figure 4(e) examines the impact of node speed on performance for Gauss-Markov mobility. Performance falls off gently with speed for both OLSR and SHARE, but SHARE delivers up to 28% more packets than OLSR (at 19 m/s).

Figure 4(b) measures the penalty of gradient control messages, which are proportional to the number of destinations. In this experiment, the same aggregate traffic was distributed to multiple destinations (x-axis). As the number of destinations increases, the SHARE delivery ratio holds steady except for large numbers of destinations. When nearly all nodes are also destinations, OLSR is likely to be comparable to SHARE.

Finally, Figures 4(c) and 4(f) compare the control overhead of SHARE with that of OLSR as a function of network size and node speed. As the network size increases, OLSR overhead increases rapidly, validating the theoretical communication complexity $\Theta(n^2)$. In contrast, SHARE overhead grows much more slowly than OLSR overhead as the network size increases. At 64 nodes, OLSR consumes 7x the overhead consumed by SHARE. Clearly SHARE is much more scalable than OLSR. The control overhead of both protocols appears largely insensitive to speed, which is unsurprising since triggering control is not tied to link up/down rate in either case.

In summary, we have shown using high-fidelity simulations that SHARE is a more scalable and robust protocol under a range of network sizes, node speeds and traffic patterns.

VII. CONCLUSIONS

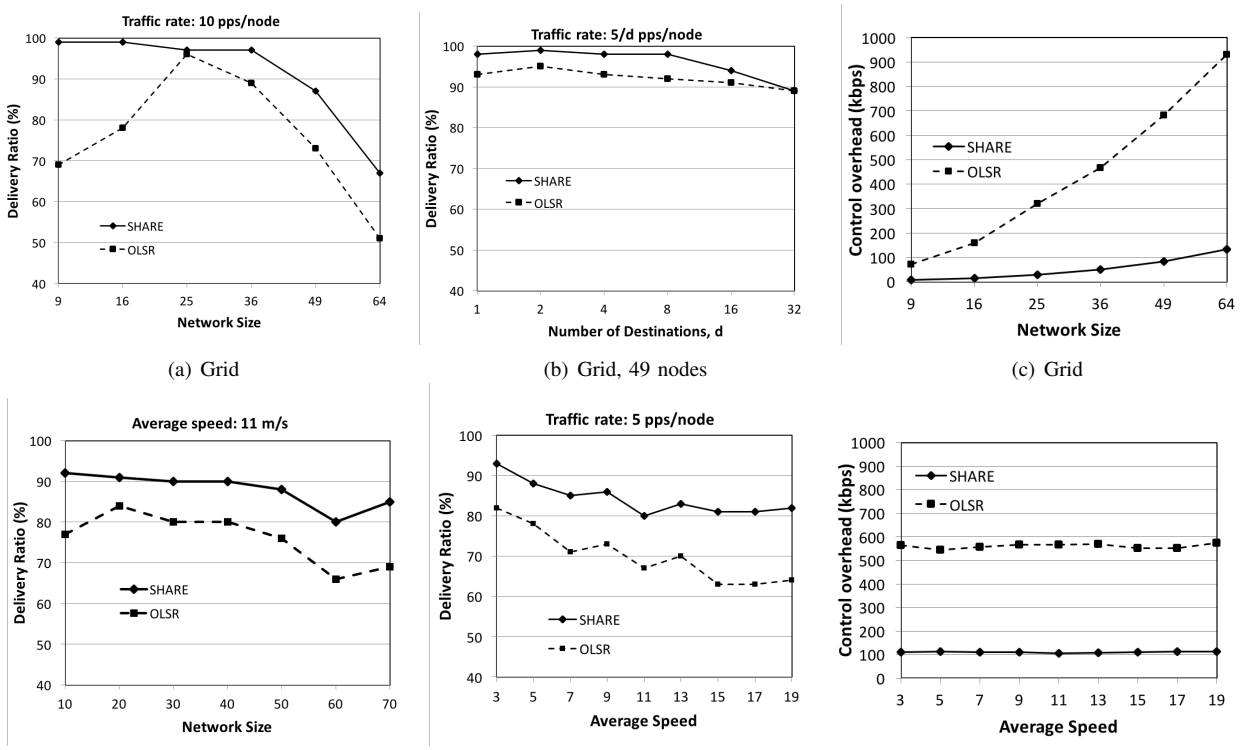
In this paper, we have proposed a new approach to scalable multi-hop routing for wireless networks, SHARE, unifying several ideas: reactive gradient routing for non-local traffic, proactive link state routing for local traffic, flooding when no gradient or link state is found, replication ("braiding") of data packets for robustness against topology changes, and a novel probabilistic truncation of gradient packet flooding to reduce control overhead. We have implemented SHARE in ns-3 using Direct Code Execution (DCE) – which can be run as is on real hardware – and evaluated SHARE against a DCE version of OLSR on both static and mobile topologies. We show for the range of parameters examined, SHARE is able to deliver up to 31% as many data packets as OLSR while consuming up to 7x less overhead. Future directions include further reducing overhead, especially for large numbers of destinations, a more generalized version of braids, QoS support, and implementation on a real-world platform. We are currently pursuing some of these directions.

ACKNOWLEDGMENTS

The authors would like to thank Wayne Phoel (DARPA) for his support of the SHARE project and his suggestions and encouragement on a number of aspects; Bow-Nan Cheng (MIT Lincoln Laboratories) for help with NRL's implementation of OLSR in ns-3 DCE; and Li Lin (BBN) for help with running simulations.

REFERENCES

[1] "Cambridge Mobile Telematics," http://www.cmtelematics.com/, accessed: 2016-04-04.
[2] "Naval Research Labaratory OLSR Routing Protocol Implementation," http://www.nrl.navy.mil/itd/ncs/products/olsr, accessed: 2016-04-20.
[3] "Network Simulator 3 (NS-3)," http://www.nsnam.org/, accessed: 2016-04-20.
[4] "Project Loon - Google," http://www.google.com/loon, accessed: 2016-04-28.
[5] "splitsecnd," https://www.splitsecnd.com/, accessed: 2016-04-04.
[6] "Toward 50 billion connected devices," https://www.ericsson.com/au/res/region_RASO/docs/2010/ericsson_50_billion_paper.pdf, accessed: 2016-12-17.
[7] U. G. Acer, S. Kalyanaraman, and A. Abouzeid, "Weak state routing for large-scale dynamic networks," in *IEEE/ACM Transactions on Networking 18:5*, 2010.
[8] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.
[9] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communication Surveys & Tutorials*, vol. 16:4, 2014.
[10] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "Bonnmotion: A mobility scenario generation and analysis tool," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010.



(a) Grid (b) Grid, 49 nodes (c) Grid

(d) Gauss-Markov (e) Gauss-Markov, 60 nodes (f) Gauss-Markov, 60 nodes

Fig. 4. Peformance of SHARE vs. OLSR for grid topology and for Gauss Markov mobility.

[11] N. Beijar, "Zone Routing Protocol," www.netlab.tkk.fi/opetus/s38030/k02/Papers/08-Nicklas.pdf, accessed: 2016-04-28.

[12] B.-N. Cheng, G. Kuperman, P. Deutsch, L. Mercer, and A. Narula-Tam, "Group-centric networking: addressing information sharing requirements at the tactical edge," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 145–151, 2016.

[13] T. Clausen and P. Jacquet, "Optimized link state routing," in *IETF RFC 3626*, 2003.

[14] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," in *Mobile Computing and Communications Review*, 2001.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *MobiCOM*, 2000.

[16] D. Johnson, N. Ntlatlapa, and C. Aichele, "SGF: a state-free gradient-based forwarding protocol for wireless sensor networks," in *ACM Transactions on Sensor Networks, 5:2*, 2009.

[17] M. Lazarus, "The troubled past and uncertain future of radio interference," *IEEE Spectrum*, vol. 53, no. 8, pp. 40–56, 2016.

[18] S.-J. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc networks," in *Wireless Communications and Networking Conference*, 2010.

[19] J. Li, C. Blake, D. S. J. D. Couto, C. Hu, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in *In ACM Mobicom*, 2001, pp. 61–69.

[20] V. Manfredi, R. Hancock, and J. Kurose, "Robust routing in dynamic MANETs," in *Proc. of the Annual Conference of the ITA (ACITA)*, 2008.

[21] A. Mohammed, Mohamed, Ould-Khaoua, L. M. Mackenzie, and J.-D. Abdulai, "Dynamic probabilistic counter-based broadcasting in mobile ad hoc networks," in *2nd International Conference on Adaptive Science & Technology*, 2009.

[22] H. Narra, Y. Cheng, E. K. Cetinkay, J. P. Rohrer, and J. P. Sterbenz, "Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in NS-3," in *Proc. WNS-3*, 2011.

[23] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren, "Scenarios for 5g mobile and wireless communications: the vision of the metis project," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 26–35, May 2014.

[24] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 40:1, 2001.

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

[26] R. Poor, "Gradient routing in ad hoc networks," in *Unpublished*.

[27] R. Ramanathan, R. Allan, P. Basu, J. Feinberg, G. Jakllari, V. Kawadia, S. Loos, J. Redi, C. Santivanez, and J. Freebersyser, "Scalability of mobile ad hoc networks: Theory vs practice," in *Military Communications Conference, 2010-MILCOM 2010.* IEEE, 2010, pp. 493–498.

[28] D. G. Reina, S. L. Toral, P. Jonhson, and F. Barrero, "Hybrid flooding scheme for mobile ad hoc networks," in *IEEE Communication Letters, Vol. 17, No. 3*, 2013.

[29] C. A. Santiváñez, R. Ramanathan, and I. Stavrakakis, "Making link-state routing scale for ad hoc networks," in *Proceedings of the 2Nd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, ser. MobiHoc '01, 2001.

[30] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," in *Wireless Communications and Networking (WCNC)*, 2003.

[31] M. Satyanarayanan, R. Schuster, M. Ebling, G. Fettweis, H. Flinck, K. Joshi, and K. Sabnani, "An open ecosystem for mobile-cloud convergence," *IEEE Communications Magazine*, 2015.

[32] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in *Proceedings of CoNEXT '13*, 2013.

[33] S. Wang, G.-H. Tu, R. Ganti, T. He, K. Leung, H. Tripp, K. Warr, and M. Zafer, "Mobile micro-cloud: application classification, mapping, and deployment," in *Proc. Annual Fall Meeting of ITA (AMITA)*, 2013.

[34] F. Ye, G. Zhong, S. Lu, and L. Zhang, "GRAdient broadcast: A robust data delivery protocol for large scale sensor networks," in *Wireless Networks 11*, 2005.