Experimentally Studying the Sensornet Point-to-Point Routing Techniques Spectrum

Konrad Iwanicki, *Member, IEEE,* and Tahir Azim

***Abstract*—The performance of point-to-point routing protocols is limited by a trade-off between routing state and routing stretch. A wireless sensor network (WSN) designer can choose from a whole spectrum of routing techniques that exploit this trade-off at varying granularity. We aim to make such choices more informed; our contributions are twofold. First, we present the first point-to-point routing framework for WSNs that provides uniform implementations of four routing techniques that, from the state-stretch trade-off perspective, represent the entire spectrum. Second, using the framework we conduct an unprecedented experimental comparison of the techniques in TOSSIM and on two testbeds. In particular, we show that most techniques with sub-linear state offer a small stretch in WSNs. These results can guide the initial choice of potential routing techniques for a given WSN; the framework can facilitate the choice of a particular technique by enabling experiments in one's own settings.**

***Index Terms*—Beacon vector routing, compact routing, hierarchical routing, point-to-point routing, shortest-path routing, wireless sensor network.**

I. Introduction

NUMEROUS application proposals and a push toward an IP-based network architecture [1] demand point-to-point routing functionality from wireless sensor networks (WSNs).

Apart from the standard robustness requirement, the major challenge WSNs introduce for a protocol delivering this functionality is that the protocol must *simultaneously* ensure *small state* and *small stretch*. Routing state describes how many routing entries each node has to maintain, as compared to the size of the whole node population (N). Small routing state is fundamental to scalability, as WSN platforms are severely constrained in terms of memory. In addition, the smaller the protocol state, the lower the traffic to maintain it, which is equally important considering the bandwidth limitations of sensor nodes. Routing stretch, in turn, describes how costly the routing paths are, that is, what the ratio is between the cost of a routing path and the cost of the optimal possible path. Small routing stretch is essential for efficiency. The smaller it is, the lower the global resource consumption and end-to-end latency, and the higher the end-to-end packet delivery rates. While it is relatively easy to ensure either small state or small stretch, ensuring both simultaneously is extremely challenging.

More specifically, there exists a trade-off between routing state and stretch [2]; in short, the smaller the state, the larger the stretch, and vice versa. As we discuss in more detail in the next section, there are volumes of theoretical work on routing techniques that exploit this trade-off at different granularity. Likewise, a plethora of protocols based on these techniques have been implemented and evaluated in WSNs.

Yet, in this body of research, there is insufficient work facilitating an informed choice of a particular routing technique by a WSN system developer. While the theory explains differences in the fundamental bounds of different techniques, its applicability when comparing the average performance is limited. This is because, as is regularly demonstrated, the assumptions of theoretical analyses oversimplify the reality, and thus, experimental results usually diverge from analytical ones [3], [4], [5], [6]. Conversely, experimental implementation-based evaluation is typically concerned with variations of one [7], [4], [5] or at most two [3], [6] routing techniques. This, combined with the fact that different techniques are evaluated in different settings, limits the extent to which existing experimental results can be compared. Even though one can experimentally compare existing implementations of different techniques in one's own setting, the obtained results may be difficult to interpret. In particular, as the existing implementations of different techniques involve different design decisions, it may be hard to disqualify a particular result as an artifact of some implementation-specific design decision.

All in all, to make an informed choice of a particular routing technique, WSN system developers require means of conducting systematic on-demand comparisons of routing techniques from the whole state-stretch trade-off spectrum. In such comparisons the implementations of different techniques should incorporate the same design decisions wherever possible and should facilitate experimenting with these decisions.

This paper aims to facilitate such an informed choice; its contributions are twofold. First, we have analyzed competing routing techniques to determine where they can be made uniform and where they truly differ. As a result of that analysis, we have implemented a point-to-point routing framework that covers the full spectrum of routing techniques: from shortest-path routing [8], [9], which requires the largest state but offers the minimal stretch, through compact [6] and hierarchical [4] routing, which at a different granularity reduce their state at the expense of stretch, to constant-state routing [7], which needs the smallest state but delivers the largest stretch. Despite their seeming differences, the techniques in our framework have uniform implementations: their specific code does not exceed 12% of the protocol code. Moreover, the framework consists of standard abstractions for different aspects of routing, such as link estimation, topology maintenance, and packet forwarding. Overall, these features facilitate attributing performance results to particular design decisions within the framework and testing how novel decisions impact different routing techniques.

K. Iwanicki is with the Institute of Informatics, the University of Warsaw, ul. Banacha 2, 02-097 Warsaw, Poland (e-mail: iwanicki@mimuw.edu.pl). He was previously with Vrije Universiteit Amsterdam.

T. Azim is with the Computer Science Department, Stanford University, Gates 284, Stanford, CA 94305, USA (e-mail: tazim@cs.stanford.edu).

Second, to demonstrate the potential of the framework, we conduct an experimental comparison of the entire state-stretch trade-off spectrum of routing techniques. For the experiments we employ a low-level simulator (TOSSIM), and small- and medium-scale testbeds. We show the relative performance of the techniques from the state-stretch trade-off perspective. In particular, we demonstrate that, unlike in wired networks, routing techniques with sub-linear state can still offer a small stretch in WSNs. To the best of our knowledge, this is the first experimental comparison of representative routing techniques from the entire state-stretch trade-off spectrum. Moreover, while we do not compare the techniques with respect to other metrics, such as the energy consumption or the latency of bootstrapping and recovering after failures, we believe that our framework can facilitate such future comparisons.

The rest of the paper is organized as follows. We survey the point-to-point routing techniques spectrum from the state-stretch trade-off perspective in Sect. II. We present our framework in Sect. III and the experimental evaluation in Sect. IV. In Sect. V, we conclude.

II. Related Work

Due to resource constraints of WSNs, point-to-point routing protocols have to trade off their state for stretch. There exists an entire spectrum of routing techniques that exploit the state-stretch trade-off at varying granularity.

One end of this spectrum represents shortest-path routing protocols, such as DSR [8] and AODV [9]. In shortest-path routing (SPR), a node's routing address is the node's unique ID. The routing state, in turn, involves one routing entry for every other node: $O(N)$ entries in total. An entry for a node contains information on the shortest route to that node, notably its cost and the ID of the next hop. In this way, nodes can route to each other along the shortest possible paths, which yields the minimal possible stretch (i.e., 1). However, since the state grows linearly with N, SPR does not scale well.

To improve scalability, protocols implementing compact routing (CR) [10], [2], such as S4 [6], have been introduced. In CR, $\sqrt{N}$ nodes are selected as beacons, and each remaining node binds itself to the closest beacon, such that a node's routing address is a concatenation of the node's own ID and the closest beacon's ID. A node's routing state consists of two types of routing entries: first, one entry for every beacon node, and second, one entry for every non-beacon node that is closer to the node than to its own beacon. This organization limits the number of entries per node to $O(\sqrt{N})$ [10], [2], [6], thereby improving scalability. Yet, by doing so, it increases the routing stretch. If a node has a routing entry for the destination node of a packet, the packet is routed to the destination along the shortest path, thus the stretch is 1. However, if a node has no entry for the destination, the packet is routed toward the beacon closest to the destination. Only when it arrives at a node that has a routing entry for the destination, is it redirected toward the destination itself. In such a case, the maximal stretch can be up to 3 [10], [2], [6].

While the small maximal stretch and sub-linear state make CR an attractive technique, $O(\sqrt{N})$ routing entries may still limit scalability. Therefore, for large networks, hierarchical routing (HR) [11], [12], [4] may be more appropriate. In HR, nodes are organized into an H-level hierarchy of clusters, with one node in each cluster being a cluster head. A node's state involves one routing entry for each level-i cluster head that is within α^i hops from the node (typically $\alpha = 2$). In this way, the number of levels, H, and of routing entries can be limited to only $O(logN)$ [11], [12], [4], which offers excellent scalability. From these $O(logN)$ cluster heads, at every level, each node selects one head and becomes a member of its cluster, such that the node's routing address is a concatenation of the IDs of these heads at subsequent levels. A routed packet is forwarded toward the lowest-level cluster head of the destination node for which the present node has a routing entry. In the worst case, it is first routed toward the head of the top-level $(H-1)$ cluster of the destination. As soon as it arrives within α^{H-2} hops from the lower-level $(H-2)$ head, it is redirected toward that head, and so on, down to the level-0 head, which represents the destination node itself. Such a redirection process, however, may result in a large maximal stretch in some networks [2].

Minimizing routing state at the expense of stretch has led to constant-state routing, which represents the other end of the state-stretch spectrum. One technique ensuring constant state is geographic routing [13], [14], [5], [15]. In geographic routing, a node's routing address corresponds to the node's geographic coordinates and the routing state contains geographic coordinates of each of the node's neighbors (i.e., nodes within the node's radio range). In this way, the routing state can be $O(1)$. Routing is normally performed greedily by forwarding a message to the neighbor whose coordinates minimize the geographic distance to the destination. In some cases, greedy forwarding fails and special fall-back mechanisms are necessary to ensure progress [5], [14], [15]. Yet, these mechanisms are complex, require additional resources, often boost the maximal stretch, and cannot be easily ported to volumetric networks. Moreover, obtaining geographic coordinates for the nodes may be problematic as well.

For these reasons, many constant-state protocols, like GEM [16] and beacon vector routing (BVR) [7], use synthetic virtual coordinates instead. In BVR, for example, B nodes are selected as beacons, where B is a constant. A node's routing address is a K-element vector ($K \leq B$), the i-th element of which denotes the number of hops from the node to the i-th closest beacon. The routing table contains one entry for each beacon node and one entry for each neighbor with a B-dimensional vector representing the neighbor's distance to each of the beacons, in total still $O(1)$ entries. Routing in BVR is done greedily by forwarding a packet to the neighbor whose B-dimensional coordinates minimize some distance metric to the destination. If greedy forwarding fails, the packet is routed to the beacon closest to the destination, and greedy routing can resume whenever it can make progress. If the packet reaches the beacon and greedy routing cannot resume, the beacon initiates a scoped flood with the radius equal to the number of hops to the destination, that is, all nodes within that radius from the beacon forward the packet. Thus, when scoped flooding is necessary, the stretch of BVR can be large.

Given the above spectrum of routing techniques, choosing

a technique most suitable for a particular application may be challenging. While theory explains asymptotic bounds on the state and stretch of different techniques, it provides little information on their practical performance in WSNs. In particular, the performance of many techniques approaches the worst-case state/stretch bounds only in topologies with short diameters, which are common in wired networks [2]. In contrast, due to embedding in physical space and a limited radio range, WSNs are "geometric:" their diameters grow fast with the node population. In such topologies, many techniques perform much better [2], [11], [4]. Conversely, existing experimental results do provide information on the practical performance of different techniques. Yet, due to differences in the implementations and the experimental settings they involved, the extent to which they can be compared is limited. Consequently, the ability to conduct systematic experimental comparisons of different techniques in one's own setting is simply a necessity.

III. Point-to-Point Routing Framework

To enable such comparisons, we have designed and implemented a point-to-point routing framework for WSNs. Our framework encompasses four routing techniques that represent the entire state-stretch spectrum: shortest-path routing (SPR), which requires a linear state and offers the minimal stretch, compact routing (CR), which requires a square-root state and offers a maximal stretch of three, hierarchical routing (HR), which trades off a low polylogarithmic state for a potentially larger stretch, and constant-state routing with virtual beacon-based coordinates (BVR), which needs a small constant state, but the stretch of which can potentially be large. We describe the decomposition of each of these techniques into well-defined abstractions, and show that these diverse techniques can be implemented with modules that are mostly shared and involve only a small fraction of protocol-specific code.

A. Motivation and Principal Idea

Existing implementations of routing techniques are *monolithic* libraries combining several different pieces of functionality. This functionality often exceeds beyond what is traditionally considered a routing protocol, usually incorporating parts of lower or higher layers, such as node address resolution [6], in the routing protocol code. In other cases, the functionality covers only some aspects of routing, such as topology maintenance and next hop look-ups [4], leaving the implementation of others to the user.

Moreover, existing implementations usually adopt customized, integrated solutions for various aspects of routing, including link quality estimation, topology maintenance, and packet forwarding [6], [4], [3], [7]. As a result, when using these implementations, it is hard to test innovative solutions dealing with these aspects. More importantly, however, without in-depth knowledge about all design decisions within a given implementation, it is difficult to attribute a particular experimental result to some implementation-specific solution adopted for some aspect of routing. By and large, non-uniform monolithic implementations make conducting and interpreting experimental comparisons challenging; in the worst case, wrong conclusions may be drawn from such comparisons.

To cope with these problems, our point-to-point routing framework is *modular*: it makes extensive use of decomposition. Our framework decomposes routing protocols into widely recognized abstractions for different aspects of routing [17], [1]. Each of these abstractions is further decomposed into functionality that is specific to a particular routing technique and functionality that is common for all considered techniques.

We believe that such an in-depth decomposition enables systematic experimental comparisons of the entire routing techniques spectrum and promotes innovation. First and foremost, it facilitates understanding the performance impact of particular design decisions in various routing abstractions on different routing techniques: for each routing abstraction, there is a clear separation between code that is common for all routing techniques and code that is technique-specific. Furthermore, since the common code constitutes the great majority of the whole code base, implementations of different techniques inherently make the same design decisions and differ only where necessary, thereby facilitating systematic comparisons. Finally, as the routing abstractions comprising a protocol in our framework are well recognized, not only can one test different design decisions within the existing implementations of these abstractions, but can also develop novel implementations of these abstractions.

B. Decomposing a Routing Protocol

To perform such a decomposition, we analyzed (and implemented in TinyOS 2.1) the selected routing techniques according to a common protocol skeleton (see Fig. 1). A routing protocol according to this skeleton reflects the network layer of a reference protocol stack. The layer below, the link layer, currently corresponds to the TinyOS active message layer, but can be replaced with another implementation that supports unicast and local-broadcast datagrams. For the layers above, in addition to standard initialization and control interfaces, a routing protocol provides a minimal interface that enables the actual routing: send a packet to a destination address and receive such packets from other addresses.

Internally, in turn, the skeleton decomposes the protocol into standard abstractions for different aspects of routing, namely link estimation, routing state maintenance, next hop look-up, and packet forwarding. These abstractions have been identified by prior research [17] and the proposed IP-based network architecture for WSNs [1].

The goal of the first abstraction, *link quality estimation*, is discovering which nodes within a node's radio connectivity (the node's neighbors) form links with the node, that is, which nodes can communicate with the node with a high packet reception rate in both directions [18], [19]. By keeping estimates of the quality of the wireless links, a routing protocol can choose high-quality links for routing paths and can dynamically change these paths when the quality of some links deteriorates or improves. This increases per-hop packet delivery rates and, consequently, decreases the routing stretch.

The next abstraction, *routing state maintenance* (a.k.a. *topology maintenance*), constitutes a foundation of routing. A

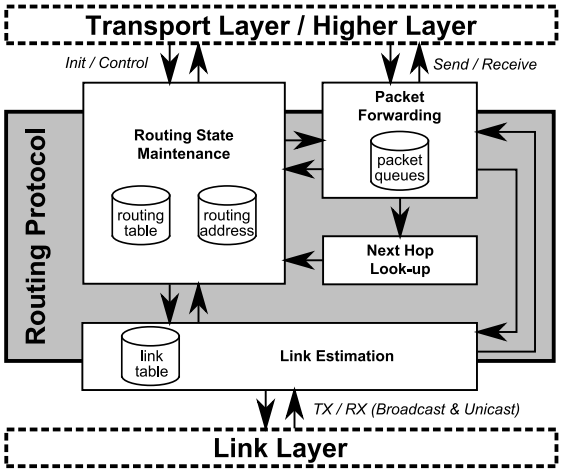

Fig. 1. A routing protocol in our framework.

node's routing state consists of a routing table, which contains information on routing paths to various destinations, and a routing address, which is used for selecting a particular path for a packet. To account for changes in the node population and the quality of wireless links, nodes have to maintain their routing tables. In addition, in some routing techniques, nodes may also be required to autonomously synthesize and maintain their routing addresses. Consequently, routing state maintenance is a crucial component of any routing technique.

While the above abstractions are necessary to enable the routing process, the process itself is encapsulated in the other two abstractions: *next hop look-up* and *packet forwarding*. Most work on a packet being routed is performed by the packet forwarding abstraction: placing the packet in a forwarding queue, waiting till it can be transmitted, transmitting it, and possibly recovering if the transmission has failed. The goal of the next hop look-up is just to provide the packet forwarding component with a list of potential next-hop candidates for the packet. Although these two abstractions are often merged into one, for reasons explained below, we decided to separate them.

C. Decomposing Routing Abstractions

To ensure that, wherever possible, the implementations of different routing techniques incorporate the same design decisions, we further decomposed each routing abstraction into technique-independent (common) and technique-specific parts. The technique-specific parts are small. Consequently, they are implemented as hook functions in the common code and are grouped together into a single TinyOS module per routing technique. By adopting opaque data types in the common code (e.g., routing addresses), the interfaces of each routing abstraction do not depend on a particular routing technique. In effect, by switching just one preprocessor definition, the user of our framework can switch between different routing techniques without changing the application code.

Due to space constraints, we cannot describe in detail how we decomposed each routing abstraction. Instead, we give an overview and refer the reader to the source code of our framework for details (see Sect. V for a pointer).

a) Link estimation: While most existing protocol implementations contain custom link estimators, research to date has demonstrated how a link estimation component should interact with other routing abstractions and has provided implementations of a few link estimation algorithms. Typically, link estimation does not depend on a routing technique. For this reason, in our framework, link estimation does not involve any technique-specific code and just requires access to the whole broadcast and unicast traffic passing through a node. More specifically, we slightly modified existing link estimators, including a beacon-based estimator [18] and a four-bit link estimator [19], to unify their interfaces and, to some extent, their semantics. By repeating this process, the user of the framework can likely port other estimators to the framework.

b) Routing state maintenance: Maintaining node routing tables, and possibly also routing addresses, is often the most intricate abstraction in a routing protocol. In each considered routing technique, the routing table of a node has a seemingly different structure: it is flat in SPR, involves two types of entries in BVR and CR, or reflects a multi-level hierarchy in HR (cf. Sect. II). Yet, by making a few observations, we can unify the routing tables of different techniques.

In all considered techniques, each routing entry corresponds to some node, referred to as a *landmark*. In other words, landmarks can be thought of as those (selected) nodes for which other nodes maintain routing entries. We can generalize that, in all the techniques, landmarks form a hierarchy, that is, each landmark has a certain *level*: in HR this is straightforward; in CR and BVR, beacon nodes are level-1 landmarks and non-beacon nodes are level-0 landmarks; in SPR, all nodes are level-0 landmarks. The landmark ID and the level thus uniquely identify a routing entry. Such an entry for a landmark is advertised to nodes within a specific distance, denoted as *scope*, from the landmark: in HR, the scope depends on the level, i, like α^i; in SPR, the scope is infinite as all nodes advertise to all other nodes; in CR, the scope is infinite for beacon nodes and equal to the distance to the closest beacon for non-beacon nodes; likewise in BVR, beacons have an infinite scope, and non beacons a scope of one hop. In other words, all nodes within the scope of a landmark maintain a routing entry for that landmark. Apart from the above fields (i.e., the ID, the level, and the scope), in every technique, an entry for a landmark has to contain the *number of hops* and a set of *next-hop neighbors* on the shortest path to the landmark, as well as maintenance fields, such as the last *sequence number* generated by the landmark and the *time-to-live* for this number.

By and large, a routing entry has conceptually the same fields for all considered routing techniques. Consequently, the implementation of a routing table in our framework is shared by the techniques. A technique that does not use some of the fields or can infer them from other fields simply does not need to store them. For example, a routing entry in SPR does not need the level and the scope fields; in HR, in turn, the level is necessary, but the scope is not. Conversely, a technique may need some additional fields, such as an adjacency flag in HR or the coordinates in the beacon space in BVR. In this case, such fields constitute a technique-specific part of the routing table and are accessed through hook functions. Overall, the

routing table provides a uniform interface encompassing entry look-ups by landmarks by IDs, iteration, and serialization.

With such a unified implementation of a routing table, the maintenance code for the table can also be unified for all techniques. There are several ways of maintaining node routing tables, the most common two being flooding and gossiping landmark advertisements. We have implemented the gossip-based version as it results in more stable routes in WSNs [4]. In essence, nodes operate in periods. In every period, each node broadcasts its whole routing table to its neighbors. The neighbors that receive the node's routing entries use these entries to update their own routing tables, following a distance-vector algorithm. When performed continuously, this algorithm allows nodes to gradually build their routing tables and maintain them when some nodes fail or the internode connectivity changes. Since the algorithm is largely independent of a routing technique, its implementation in our framework is mostly shared by all techniques. There are just a few hook functions, for example, for influencing decisions about using a given received routing entry to update a node's own entry.

In addition to maintaining routing tables, some deployments may require nodes to also autonomously synthesize and maintain their routing addresses. While address maintenance is beyond the scope of this paper, we have implemented each routing technique with both static and dynamically maintained addresses. Again, in the considered techniques, the dynamic address maintenance code can also be shared to a large extent.

c) Packet forwarding: Packet forwarding involves the activities associated with forwarding a packet, from the moment the packet is received by a node till the moment the responsibility for the packet is passed to the next-hop node(s). These activities involve detection of packet duplicates, queuing packets, looking up a list of next-hop candidates, selecting one or more next hops from the candidates, transmitting the packet, and possibly recovering if the transmission has failed. As in our framework next hop look-up is separated from the other activities, packet forwarding can be independent of a particular routing technique. If there were some dependency, however, it could be implemented with hook functions invoked from the common forwarding code. Currently, the framework offers two variants of packet forwarding, namely unicast forwarding and scoped floods, but we are considering implementing other forwarding techniques, such as opportunistic forwarding [20].

d) Next hop look-up: Next hop look-up is the only routing abstraction in our framework that is completely technique-specific. Although, making use of our earlier observations regarding the routing table, we could make the next hop look-up code shared to some extent, we decided to avoid this as it impairs the readability of the code. Even now, though, the next hop look-up consists of relatively few lines of code.

D. Framework Code Breakdown

Table I presents the routing code breakdown for our framework. As we demonstrated in the above analysis, the great majority of the code is shared by all techniques. This ensures that, wherever possible, the implementations of the techniques make the same design decisions. A clear separation between technique-specific and common code, as provided by the hook functions, facilitates distinguishing between technique-specific and other decisions. This, combined with the decomposition of a routing protocol into standard, widely-recognized routing abstractions, makes it possible to attribute performance results to particular design decisions, which, in effect, enables systematic comparisons of different routing techniques.

TABLE I
THE CODE BREAKDOWN OF PROTOCOLS WITH STATIC ADDRESSES.

Functionality	**Lines of Code** (SLOCCount)				
	Common	**SPR**	**CR**	**HR**	**BVR**
link estimation	997	0	0	0	0
routing state maintenance	1954	76	89	132	173
packet forwarding	1505	15	13	21	51
next hop look-up	0	14	39	67	154
shared by the abstractions	52	106	112	157	134
TOTAL	4508	211	253	377	512
SPEC. / (COMM.+SPEC.)	—	4.7%	5.6%	8.4%	11.4%

Apart from the routing code, our framework contains sizable additional functionality that simplifies experimentation. For brevity, we omit the description of that functionality.

IV. Experimental Evaluation

Using our framework, we conducted an experimental comparison of the selected techniques from the state-stretch trade-off perspective. The comparison employed three experimental platforms: TOSSIM, a low-level TinyOS simulator, *KonTest*, a 50^{+}-node testbed, and *MoteLab*, a 100^{+}-node testbed. We experimented with various routing scenarios, network sizes, densities, topologies, and radio models. To the best of our knowledge, this has been the first such extensive experimental comparison involving point-to-point routing techniques that represent the entire state-stretch trade-off spectrum.

From the conducted experiments, we present selected results that (1) demonstrate sample usage scenarios of our framework and (2) provide novel observations on the performance of different routing techniques. While we tried to make the comparisons fair, we acknowledge that, for example, technique-specific optimizations can alter the results. Therefore, the goal of our results is just to illustrate the performance achievable in practice by the selected techniques. Our framework, in turn, can help evaluate how adding a technique-specific optimization or changing experimental settings affects the performance of a given routing technique.

A. Verifying Protocol Correctness

A basic environment for evaluating WSN routing protocols is a topology with a unit-disk radio model. In the unit-disk model, each node has the same circular radio range and can communicate only with the nodes within this range; in addition, there is no packet loss. Engineering the internode connectivity and precomputing the node routing state is straightforward in this model. Moreover, in the absence of packet loss, the behavior of a routing protocol is completely predictable. Finally, simulating unit-disk connectivity is relatively fast as compared to other models of low-power wireless communication. For these reasons, the unit-disk model is typically used for verifying the correctness of routing protocols.

Figure 2 depicts the routing state and stretch of different techniques in unit-disk grids of different sizes simulated in TOSSIM. The routing state is measured in the number of routing entries at every node. The routing stretch, in turn, is computed as the ratio between the number of transmissions necessary to deliver a packet over a routing path between two nodes to the number of hops on the shortest possible path between these nodes. The stretch is measured over 100×99 random source-destination pairs.

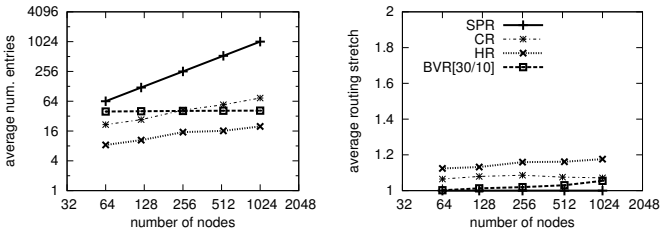

Fig. 2. The routing state (left) and the routing stretch (right) of different routing techniques in unit-disk grids of different sizes. *The numbers [30/10] for BVR correspond to B and K respectively (see Sect. II). Essentially, the figure illustrates the state-stretch trade-off.*

The figure essentially illustrates the state-stretch trade-off in point-to-point routing, thereby experimentally reinforcing prior theoretical results we surveyed in Sect. II. In terms of the way their routing tables scale with the network size, the selected techniques indeed represent the entire spectrum of choices. Likewise, the smaller their routing state, the larger their routing stretch, and thus one can choose a technique that makes the most appropriate trade-off for a given application.

The only technique that seems off in Fig. 2 is BVR. Despite having its routing state independent of the network size, BVR maintains more routing entries than HR, the state of which scales polylogarithmically with the number of nodes. This is because if fewer beacons are used in BVR, its routing stretch grows fast, as there are not enough coordinates to successfully deliver packets and, consequently, many expensive scoped floods are required (cf. the right plot in Fig. 3). Moreover, despite fewer routing entries than in CR, BVR seems to provide better routing stretch than CR. This can be explained by the fact that, whereas in the other techniques the size of a single routing entry is constant, in BVR it grows linearly with the number of beacons. As a result, the routing state, when measured in bytes, is much larger in BVR than in CR (cf. the left plot in Fig. 3). If the number of BVR beacons decreases, the routing state decreases as well, but this boosts the routing stretch (cf. the right plot in Fig. 3).

In general, there are a few problems with BVR. Even though BVR is considered a constant-state routing technique, it is not clear if the number of beacons it uses should not depend on the node population. In particular, if the number of beacons remains constant for an increasing network, more scoped floods are necessary to deliver packets and the protocol becomes more sensitive to the placement of these beacons (e.g., see the 1024-node network in Fig. 3). Furthermore, we discovered an inherent flaw in the BVR algorithm, as presented originally [7]: even with scoped floods, BVR does not guarantee packet delivery. Although a physical path between two nodes may

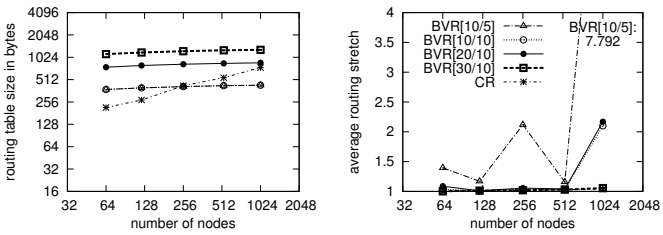

Fig. 3. The routing state in *bytes* (left) and the routing stretch (right) of BVR with different numbers of beacons in unit-disk grid networks . *The settings are the same as in Fig. 2. Decreasing the number of beacons in BVR reduces the routing state, but it also boosts the routing stretch, as more expensive scoped floods are necessary to deliver packets.*

exist, BVR may not discover this path; instead, it may push routed packets into a routing loop, which results in permanent routing failures for these two nodes. We observed such behavior for up to 5% of source-destination pairs, depending on the number of nodes and beacons. To alleviate such failures, modifications, such as duplicate detection, are necessary in the BVR packet forwarding routines.

B. Assessing Expected Protocol Performance

While the unit-disk radio model is useful for verifying the protocol correctness, it does not accurately approximate real-world performance [3], [4], [6], [7]. In the real world, low-power wireless communication suffers from noise, asymmetric links, and other phenomena. Consequently, to better predict the practical performance of a routing protocol, more accurate models of low-power wireless have to be used.

To this end, we also evaluated our framework under realistic low-power wireless models of TOSSIM. In those experiments, nodes were deployed randomly in a square area, and we varied their number as well as the size of the area. Using TOSSIM tools, we generated realistic radio gain values for each resulting deployment. We also used noise models generated from TOSSIM's Cassino Lab traces for these experiments. To measure link quality, we employed a proper link estimator (beacon-based estimator [18] in the figures); only links with at least 55% packet reception were used in routing paths. The routing state maintenance component dynamically synthesized and maintained node routing addresses with probabilistic heuristics mentioned in Sect. III-C. The packet forwarding routines recovered from unacknowledged transmissions by retransmitting packets up to 10 times per hop; the back-off between retransmissions was 1 s to cope with potential link burstiness [21]. Like before, we randomly selected 100 nodes, obtaining 100×99 source-destination pairs. Each source generated 10 packets for each destination, one packet per second, and then the process continued at the next source.

Figure 4 illustrates that, in terms of routing state, the technique implementations in our framework behave similarly in realistic networks as in idealized unit-disk networks. Each technique occupies its respective position in the state dimension of the technique spectrum, as we explained in Sect. II. However, the state maintained in realistic networks by CR and HR is larger than in unit-disk grid networks (cf. Fig. 2). This is consistent with recent results for HR [4], which attributed the state increase to the irregularities in the internode connectivity,

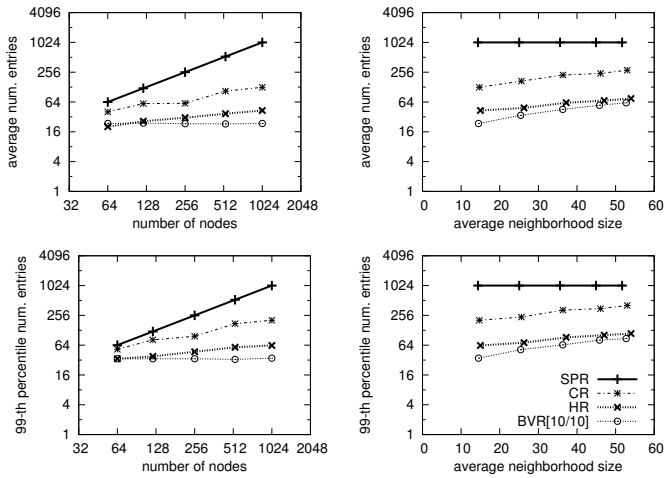

Fig. 4. The average (top) and the 99-th percentile (bottom) routing state in different networks with realistic low-power connectivity models. *The "state dimension" of the state-stretch trade-off spectrum is clearly visible.*

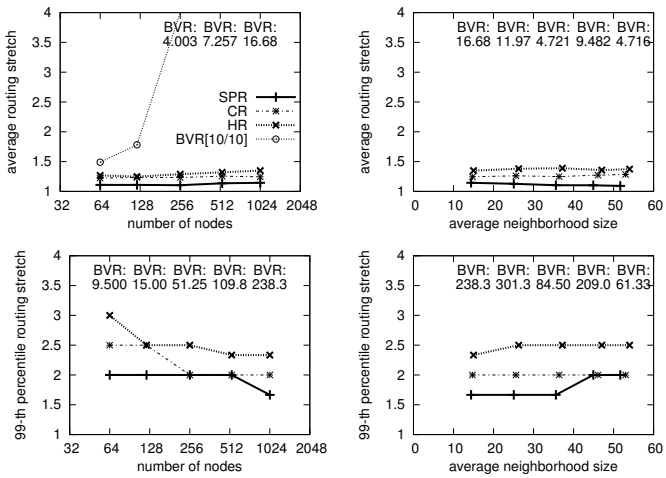

Fig. 5. The average (top) and the 99-th percentile (bottom) routing stretch in different networks with realistic low-power connectivity models. *The "stretch dimension" of the state-stretch trade-off spectrum is clearly visible.*

an inherent feature of low-power wireless communication. Also similarly to that work, the 99-th percentile routing state in our experiments is twice the average state. In other words, when provisioning routing entry memory pools for the 99-th percentile, one can expect that a node running CR or HR on average utilizes 50% of its pool. All in all, these results suggest that the technique implementations in our framework can work correctly under realistic low-power wireless communication.

Figure 5 depicts corresponding results for routing stretch. Like under unit-disk models, except for BVR, all the techniques offer a relatively small average and 99-th percentile stretch: below 1.5 and 3 respectively. This means that in practice the routing paths of CR and HR are close to the optimal paths. The small differences between different techniques just confirm that the smaller the state of a routing technique, the larger the stretch. In contrast, as explained in Sect. II and Sect. IV-A, due to scoped flooding, the routing stretch of BVR can be prohibitively large. Note also that, under realistic radio connectivity models, the routing stretch for SPR is slightly greater than one. This is because of packet loss triggering retransmissions, which increases the stretch. Nevertheless, as overall the stretch of SPR, CR, and HR is small, the mechanism our framework uses for dealing with the lossy nature of wireless links appear effective. Moreover, these mechanisms ensure that the end-to-end packet delivery rate is high, around 98% for a 1024-node network; the remaining 2% loss can be further reduced by implementing additional reliability mechanisms, such as local failure recovery [6], [1].

All in all, while theoretical results prove that the routing stretch bounds of different techniques vary considerably if *arbitrary* network topologies are considered, we showed above that, except for BVR, there is little difference in the stretch of the techniques in typical WSN topologies. This is because, due to embedding in physical space and a limited node radio range, the topologies of WSNs are usually "geometric," that is, their diameter grows fast: as N^v (where $1 \geq v > 0$). Krioukov et al. [2] suggest that in such topologies even protocols with a small state can offer a nearly optimal stretch, that is, the protocols perform much above their worst-case stretch bounds in WSNs. Not only does this indicate that small-state protocols may in practice offer excellent stretch, but also highlights the need for a means of on-demand experimental comparison of different routing techniques, which our framework aims to provide.

C. Testing Real-World Behavior

Finally, to verify that our framework can also work on real WSN hardware, we evaluated it on two testbeds. Except for the number of BVR beacons, which we had to decrease to stay within the memory budget of sensor nodes, the settings for these experiments were largely the same as those in TOSSIM. Selected testbed results are presented in Table II.

TABLE II
THE PERFORMANCE OF DIFFERENT TECHNIQUES ON THE TWO TESTBEDS.

Metric	*KonTest*				*MoteLab*			
	SPR	**CR**	**HR**	**BVR***	**SPR**	**CR**	**HR**	**BVR***
active nodes	53				104			
diameter [hops]	4				6			
avg. neighborhood	17.48				14.10			
99-th perc. neighborhood	25				27			
source-destination pairs	10×9				10×9			
avg. routing state [entries]	53	29.74	5.55	26.43	104	38.71	10.92	23.10
99-th perc. state [entries]	53	35	7	34	104	59	16	36
avg. routing stretch	1.036	1.074	1.061	1.045	1.051	1.158	1.198	2.984
99-th perc. stretch	1.333	1.667	2.0	1.5	2.0	2.0	3.0	26.5

* The [10/5] variant of BVR.

In short, there is little difference between the results on the testbeds and in TOSSIM with realistic low-power wireless models. In terms of routing state, most techniques perform exactly as predicted by the TOSSIM. The only exception is HR, which seems to offer smaller routing state in the real world. This is because while the TOSSIM topologies we used were highly irregular, with many asymmetric and flapping links, the links in both testbeds were more stable and symmetric. With more stable and symmetric links, HR can build cluster hierarchies that require less state. Likewise, in terms of routing stretch, the techniques seem to perform slightly better on the testbeds. This is again a consequence of better internode

connectivity and of a smaller scale of the testbed experiments. All in all, the testbed experiments illustrate that the routing technique implementations in our framework work seamlessly in the real world and their performance can be predicted to a large extent by low-level simulations.

V. Concluding Discussion

Having seen how different routing techniques perform in WSNs, it may be tempting to ask which of them performs best. While the original work on BVR [7] was the first to show what functionality a routing protocol for WSNs should contain, BVR performs relatively poorly and is relatively difficult to implement (cf. Table I). Therefore, it seems rather unattractive as compared to the other techniques. SPR, the dominant intra-domain routing technique in wired networks, provides the smallest stretch and is the easiest to implement in WSNs. However, the routing state of SPR may be excessive, even for some medium-sized WSN deployments. In terms of the state-stretch trade-off, CR and HR seem the most attractive. Despite their sub-linear state, due to the geometric nature of WSNs, both CR and HR in practice offer a stretch that is close to the optimal one. CR offers a slightly better stretch and is easier to implement than HR (cf. Table I), but its state is significantly larger than that of HR. Due to the resource constraints of WSNs, a large difference in state is often more important than a small difference in stretch. Therefore, HR can arguably be more attractive when it comes to performance; when it comes to the implementation complexity, in turn, CR may be a better choice. In other words, there is no one-size-fits-all point-to-point routing technique for WSNs, but, in contrast to wired networks, a few are suitable, depending on the application.

Moreover, state and stretch are just two of the many performance metrics for routing protocols. Examples of other important properties include energy efficiency, failure resilience, load balancing capabilities, and path stability. While we have been studying such properties, due to space constraints we excluded the results from this paper. Likewise, one can also fine-tune the implementations of different techniques to a given deployment, which may also affect their relative performance.

Therefore, we believe that while our comparison of the techniques spectrum is an important contribution, our framework is at least equally valuable. Not only can it enable experimentally studying various properties of the techniques spectrum in one's own settings, but can also facilitate developing novel solutions for different aspects of routing. For these reasons, we are considering publishing the framework.

References

[1] J. Hui and D. Culler, "IP is dead, long live IP for wireless sensor networks," in *Proc. ACM SenSys 2008*, Raleigh, NC, USA, November 2008.

[2] D. Krioukov, k c claffy, K. Fall, and A. Brady, "On compact routing for the Internet," *ACM SIGCOMM Comput. Commun. Review*, vol. 37, no. 3, pp. 41–52, July 2007.

[3] H. Frey and K. Pind, "Dynamic source routing versus greedy routing in a testbed sensor network deployment," in *Proc. EWSN 2009*. Cork, Ireland: Springer LNCS 5432, February 2009.

[4] K. Iwanicki and M. van Steen, "On hierarchical routing in wireless sensor networks," in *Proc. ACM/IEEE IPSN 2009:*, San Francisco, CA, USA, April 2009.

[5] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *Proc. USENIX NSDI 2005*, Boston, MA, USA, May 2005.

[6] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith, "S4: Small State and Small Stretch routing protocol for large wireless sensor networks," in *Proc. USENIX NSDI 2007*, Cambridge, MA, USA, April 2007.

[7] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica, "Beacon vector routing: Scalable point-to-point routing in wireless sensornets," in *Proc. USENIX NSDI 2005*, Boston, MA, USA, May 2005.

[8] D. B. Johnson and D. A. Maltz, *Mobile Computing*. Springer US, 1996, vol. 353, ch. Dynamic Source Routing in Ad Hoc Wireless Networks, pp. 153–181.

[9] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. IEEE WMCSA 1999*, New Orleans, LA, USA, February 1999.

[10] L. J. Cowen, "Compact routing with minimum stretch," in *Proc. ACM-SIAM SODA 1999*, Baltimore, MD, USA, January 1999.

[11] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks," *Comput. Networks*, vol. 1, no. 3, pp. 155–174, 1977.

[12] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," *ACM SIGCOMM Comput. Commun. Review*, vol. 18, no. 4, pp. 35–42, August 1988.

[13] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM MobiCom 2000*, Boston, MA, USA, August 2000.

[14] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in *Proc. ACM PODC 2003*, Boston, MA, USA, July 2003.

[15] B. Leong, B. Liskov, and R. Morris, "Geographic routing without planarization," in *Proc. USENIX NSDI 2006*, San Jose, CA, USA, May 2006.

[16] J. Newsome and D. Song, "GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information," in *Proc. ACM SenSys 2003*, Los Angeles, CA, USA, November 2003.

[17] C. T. Ee, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, and I. Stoica, "A modular network layer for sensornets," in *Proc. USENIX OSDI 2006*, Seattle, WA, USA, November 2006.

[18] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proc. ACM SenSys 2003*, Los Angeles, CA, USA, November 2003.

[19] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four-bit wireless link estimation," in *Proc. ACM HotNets-VI*, Atlanta, GA, USA, November 2007.

[20] G. Schaefer, F. Ingelrest, and M. Vetterli, "Potentials of opportunistic routing in energy-constrained wireless sensor networks," in *Proc. EWSN 2009*. Cork, Ireland: Springer LNCS 5432, February 2009.

[21] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: Measuring wireless link burstiness," in *Proc. ACM SenSys 2008*, Raleigh, NC, USA, November 2008.