A Case for Hierarchical Routing in Low-Power Wireless Embedded Networks

KONRAD IWANICKI, University of Warsaw
MAARTEN VAN STEEN, VU University Amsterdam

Hierarchical routing has often been mentioned as an appealing point-to-point routing technique for wireless sensor networks (sensornets). While there is a volume of analytical and high-level simulation results demonstrating its merits, there has been little work evaluating it in actual sensornet settings. This article bridges the gap between theory and practice.

Having analyzed a number of proposed hierarchical routing protocols, we have developed a framework that captures the common characteristics of the protocols and identifies design points at which the protocols differ. We use a sensornet implementation of the framework in TOSSIM and on a 60-node testbed to study various trade-offs that hierarchical routing introduces, as well as to compare the performance of hierarchical routing with the performance of other routing techniques, namely shortest-path routing, compact routing, and beacon vector routing. The results show that hierarchical routing is a compelling routing technique also in practice. In particular, despite only logarithmic routing state, it can offer small routing stretch: an average of ∼1.25 and a 99th percentile of 2. It can also be robust, minimizing the maintenance traffic or the latency of reacting to changes in the network. Moreover, the trade-offs offered by hierarchical routing are attractive for many sensornet applications when compared to the other routing techniques. For example, in terms of routing state, hierarchical routing can offer scalability at least an order of magnitude better than compact routing, and at the same time, in terms of routing stretch, its performance is within 10–15% of that of compact routing; in addition, this performance can further be tuned to a particular application. Finally, we also identify a number of practical issues and limitations of which we believe sensornet developers adopting hierarchical routing should be aware.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*

General Terms: Algorithms, Design, Performance, Experimentation

Additional Key Words and Phrases: Point-to-point routing, hierarchical routing, compact routing, small-state routing, small-stretch routing, wireless sensor networks, self-organizing protocols

ACM Reference Format:
Iwanicki, K. and van Steen, M. 2012. A case for hierarchical routing in low-power wireless embedded networks. ACM Trans. Sensor Netw. 8, 3, Article 25 (July 2012), 34 pages.
DOI = 10.1145/2240092.2240099 http://doi.acm.org/10.1145/2240092.2240099

This paper is a revised and extended version of Iwanicki and van Steen [2009b].
This work was supported by the Foundation for Polish Science under grant HOMING PLUS/2010-2/4 and a START scholarship (K. Iwanicki). Authors' addresses: K. Iwanicki, Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw, ul. Banacha 2, 02-097 Warszawa, Poland; email: iwanicki@mimuw.edu.pl (corresponding author); M. van Steen, VU University, Faculty of Sciences, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands; email: steen@cs.vu.nl.

DOI 10.1145/2240092.2240099 http://doi.acm.org/10.1145/2240092.2240099

1. INTRODUCTION

The principal idea behind numerous applications proposed recently for low-power wireless embedded networks (sensornets) is that of pairing wireless sensor nodes with similar actuator nodes. These applications illustrate a more general paradigm shift in network architectures for sensornets: from custom architectures designed solely for centralized many-to-one data collection, toward more universal Internet-like architectures which rely on any-to-any communication [Hui and Culler 2008]. This paradigm shift requires novel sensornet architectures to support a fundamental Internet functionality: point-to-point routing.

The way point-to-point routing is supported is crucial for sensornet applications. In particular, the performance of a protocol delivering the routing functionality directly affects scalability, efficiency, and reliability of the applications on top. Since many of the proposed sensornet applications involve large node populations, it is essential to provide point-to-point routing protocols that can simultaneously ensure the following properties [Fonseca et al. 2005; Iwanicki and van Steen 2009b; Kim et al. 2005; Mao et al. 2007].

—*Small routing state.* Small routing state is crucial for scalability. With only a few kilobytes of memory in typical sensor node platforms, minimizing the routing state enables supporting larger networks. Moreover, a smaller routing state often implies lower maintenance traffic, which is important for efficiency, considering the bandwidth limitations of sensornets.
—*Small routing stretch.* Routing stretch describes the cost of the paths selected by a routing protocol compared to the cost of the optimal possible paths, for example, the ratio between the number of routing hops actually taken by a packet and the minimal possible number. The fewer hops there are on a routing path, the lower the global resource consumption and end-to-end packet latencies and the higher the end-to-end packet delivery rates. Small routing stretch is thus crucial for efficiency and reliability.
—*Robustness.* Robustness is in turn important for reliability and scalability. Sensornets experience topology and connectivity changes due to node failures and environmental impact; the larger the network, the more changes are to be expected. Therefore, to be reliable and scalable, a routing protocol has to be able to handle such changes. To minimize resource consumption and disruption of applications on top, the routing protocol should handle the changes in the network with minimal traffic and latency.

Considering these requirements, from the available spectrum of routing techniques (which we discuss in detail in the next section), hierarchical routing [Kleinrock and Kamoun 1977; Tsuchiya 1988] has often been mentioned as an appealing technique. In hierarchical routing, nodes are organized into a hierarchy of clusters that governs node addressing and packet forwarding. This organization can necessitate as little as $O(logN)$ routing state per node (where N denotes the node population size) and can offer a relatively small routing stretch in some network topologies. Moreover, numerous protocols that promise distributed formation and robust maintenance of a cluster hierarchy have been proposed.

However, despite these merits of hierarchical routing, surprisingly little work has been done to evaluate this technique in actual sensornets. The proposed protocols have mostly been studied with high-level simulations in idealized environments: a unit-disk connectivity model, no message loss, etc. While high-level simulations give insight into graph-theoretical properties of a protocol, they do not provide enough information on the practical performance of the protocol. Since numerous examples evidence that in sensornets, practice typically diverges from theory, any potential real-world

applications of hierarchical routing demand thorough implementation-based evaluations. Yet, we are not aware of any evaluation of hierarchical routing that employs an actual embedded protocol implementation. Moreover, prior simulations usually consider only a single protocol without comparing it against alternative solutions and discussing the trade-offs those solutions introduce. They also do not compare hierarchical routing against other representative routing techniques from the available spectrum. However, different applications may expect different trade-offs from a routing protocol, which requires a sensornet system developer to have knowledge not only about the performance differences between hierarchical routing and other techniques, but also about various possible solutions within hierarchical routing itself. To this end, systematic experimental comparisons of hierarchical routing and other competing routing techniques, as well as of different solutions within hierarchical routing itself, are necessary.

In this article, we bridge the gap between theory and practice. We have analyzed a number of hierarchical point-to-point routing protocols presented in major networking conferences and journals. Based on that analysis, we have developed a framework that allows us to implement hierarchical routing for sensornets. The framework captures common characteristics of many proposed hierarchical routing protocols and at the same time identifies various design points that differentiate the protocols and allow for exploring the design space. By building TinyOS 2.0 implementations of the framework, we have obtained a means to empirically compare hierarchical routing against other representative point-to-point routing techniques. In addition, the design points within the framework allow us to systematically evaluate various design options within hierarchical routing itself.

We have conducted experimental evaluations of the framework in TOSSIM, a low-level simulator with a realistic low-power wireless communication model, and on our 60-node sensornet testbed, KonTest. In addition, we have also run several experiments on the publicly available 100+-node testbed, MoteLab. Our results show that, despite only logarithmic routing state, hierarchical routing can offer small routing stretch with an average of $\sim$1.25 and a 99th percentile of 2. Compared to other representative routing techniques, namely shortest-path routing [Perkins and Royer 1999], compact routing [Mao et al. 2007], and beacon vector routing [Fonseca et al. 2005], the trade-off between routing state and routing stretch offered by hierarchical routing is attractive for many sensornet applications. For example, we show that in terms of routing state, hierarchical routing offers scalability of at least an order of magnitude better than compact routing, and at the same time, in terms of routing stretch, its performance is within 10–15% of the performance of compact routing. Furthermore, we demonstrate that by varying the solutions at the design points within the framework, one can tune hierarchical routing to a particular application. For instance, one can trade off state for stretch within hierarchical routing itself or can emphasize particular robustness metrics, such as the latency of reacting to changes in the network or the traffic used to maintain the cluster hierarchy. All in all, the results demonstrate that hierarchical routing is a compelling point-to-point routing technique for low-power wireless embedded networks.

The rest of the article is organized as follows. We start with an analysis of related work on routing in Section 2 followed by a more in-depth discussion of hierarchical routing in Section 3. We present the developed hierarchical routing framework and the setup for the conducted experiments in Section 4 and Section 5, respectively. We study the experimental results obtained in TOSSIM and on the testbed in Section 6 and Section 7, respectively. Finally, in Section 8, we offer a concluding discussion.

2. RELATED WORK

Although point-to-point routing is a well-studied problem, sensornets pose novel challenges, which are directly associated with the aforementioned three requirements:

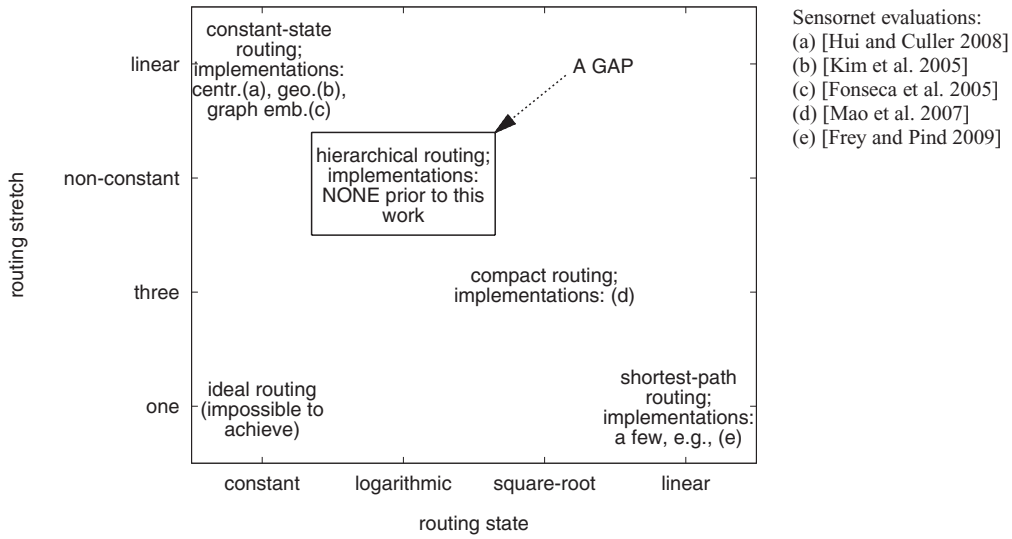


Fig. 1. The spectrum of routing techniques with respect to the state-stretch trade-off and their sensornet evaluations.

small routing state, small routing stretch, and robustness. While it is relatively easy for a routing protocol to satisfy some of these requirements, satisfying all three simultaneously is extremely difficult and, in practice, entails some trade-offs. Let us analyze the spectrum of routing techniques from the perspective of one such fundamental trade-off: the trade-off between routing state and routing stretch [Krioukov et al. 2007].

One end of the spectrum of routing techniques, when viewed from the state-stretch trade-off perspective (cf., Figure 1), corresponds to shortest-path routing. Examples of shortest-path routing protocols for wireless networks include AODV [Perkins and Royer 1999] and DSR [Johnson and Maltz 1996]. In shortest-path routing, each node is addressed by a flat unique identifier and essentially maintains a routing entry with the shortest path for every other node in the network. This yields the minimal possible routing stretch, 1, but at the same time results in a large routing state, which scales as $O(N)$ per node, where N denotes the total node population size. Even though shortest-path routing has become the dominant intra-domain routing technique in the Internet, its applicability to memory- and bandwidth-constrained sensornets is limited because of the large state. In general, routing techniques for sensornets aim at minimizing routing state while trying to incur as small an increase in stretch as possible.

The most extreme examples of this trend are constant-state routing techniques, which constitute the other end of the state-stretch trade-off spectrum (cf., Figure 1). One example of a constant-state routing technique is centralized routing, implemented in CentRoute [Stathopoulos et al. 2007] and in a protocol by Hui and Culler [2008]. The principal idea behind centralized routing is designating an additional powerful node as the main router and organizing all resource-constrained nodes into a spanning tree rooted at the main router such that each node maintains only a single routing entry, that is, to the parent node in the tree. The main router, in contrast, maintains information about the topology of the entire network. When a source node needs to route a packet to some destination node, the packet is first forwarded up the tree to the main router. Then, based on the global connectivity information, the main router computes the shortest route to the destination node and source-routes the packet down

the tree along this route. A major advantage of this approach is only $O(1)$ routing state at each resource-constrained node. However, because of such a small state, the stretch in centralized routing is large, especially in large networks. In extreme cases, it can be proportional to N. Moreover, as the top-level nodes in the tree together forward all packets in the network, they may become routing hot spots in large networks, especially since they have incomparably fewer resources than the main router. For these reasons, alternative constant-state routing techniques have been sought.

One such alternative constant-state routing technique is geographic routing [Bose et al. 1999; Karp and Kung 2000; Kuhn et al. 2003]. In geographic routing, a node is addressed by its geographic coordinates. Its routing state, in turn, consists of the coordinates of its neighbors, that is, the nodes within the radio range of the node. In this way, assuming that the density of the network is independent of the total node population size, geographic routing requires only $O(1)$ state per node. However, a practical, efficient geographic routing protocol is essentially still an open research problem. While normally a geographically routed packet is forwarded greedily to the neighbor with the coordinates minimizing the distance to the destination, difficulties arise when such a neighbor does not exist and a so-called routing void is encountered. Because of real-world issues, such as geographic localization errors or physical obstacles preventing radio communication between nearby nodes, special solutions are necessary for handling routing voids [Kim et al. 2005]. These solutions involve protocols for planarizing the neighborhood graph using so-called cross-link detection (CLDP) [Kim et al. 2005] or for building hull trees [Leong et al. 2006]. However, such approaches make routing state maintenance or packet forwarding costly and complex: they introduce many subtle corner cases or necessitate two-phase locking to ensure consistency [Kim et al. 2005; Leong et al. 2006]. In addition, providing nodes with their geographic coordinates requires special hardware or additional localization algorithms, which both consume node resources and often do not work efficiently in some environments. Finally, there is no practical way of porting geographic routing to three dimensions, and thus, volumetric indoor networks may require different protocols.

To cope with these problems introduced by geographic coordinates, a number of constant-state protocols employ virtual coordinates that are synthesized using so-called graph embedding. For example, in beacon vector routing (BVR) [Fonseca et al. 2005], B nodes are selected as beacons, where B is, in principle, independent of the total node population size, N. A node is addressed by a K-element vector of virtual coordinates ($K \leq B$), the ith element of which denotes the number of hops from the node to the ith closest beacon. In other words, the address of the node corresponds to virtual coordinates in the B-dimensional space induced by the B beacon nodes. The routing state of the node, in turn, contains one routing entry for each beacon node and one entry for each neighbor with the B-dimensional coordinates of that neighbor—in total, still $O(1)$ entries. Like in geographic routing, a packet in BVR is forwarded greedily at each hop: the neighbor of which the B-dimensional coordinates minimize the distance to the destination is chosen as the next hop. If greedy forwarding encounters a routing void, the packet is routed toward the beacon closest to the destination, and greedy forwarding can resume as soon as it can make progress. If the packet reaches the beacon closest to the destination and greedy routing still cannot resume, as a last resort, the beacon initiates a scoped flood with the radius equal to the number of hops to the destination, that is, the beacon broadcasts the packet and all nodes within that radius from the packet. Therefore, when scoped flooding is used to deliver a packet, the routing stretch of BVR is large.

Other constant-state routing techniques based on graph embedding employ similar ideas. For example, NoGeo [Rao et al. 2003] synthesizes virtual node coordinates through an iterative relaxation that embeds nodes in a Cartesian space. For the same

purpose, GSpring [Leong et al. 2007] uses spring relaxation which, in addition, guarantees that a resulting coordinate space will lack routing voids. A lack of routing voids is also guaranteed by another graph-embedding protocol, which employs conformal mapping with Ricci flows [Sarkar et al. 2009]. GEM [Newsome and Song 2003], in turn, embeds nodes in a polar coordinate space based on a tree spanning from one node, which bears some similarities with centralized routing. In general, an important advantage of the constant-state techniques based on graph embedding is that they do not require any additional localization mechanisms and can be used in volumetric deployments. However, this comes at a cost of excessive coordinate setup time [Leong et al. 2007; Rao et al. 2003; Sarkar et al. 2009], intricate recovery after a node failure [Leong et al. 2007; Newsome and Song 2003; Sarkar et al. 2009], or stretch-exploding flooding fallback due to a lack of routing guarantees [Fonseca et al. 2005; Rao et al. 2003]. Because of these drawbacks, other routing techniques are of increasing interest.

In contrast to shortest-path routing and constant-state routing, which constitute the two extremes of the state-stretch trade-off spectrum, alternative techniques aim to find some middle ground between state and stretch. Compact routing [Krioukov et al. 2007], the third main technique from the spectrum (cf., Figure 1), aims to ensure a maximal routing stretch of three while using $O(\sqrt{N})$ routing state per node. In compact routing, which has been adopted to sensornets by the S4 protocol [Mao et al. 2007], $\sqrt{N}$ nodes are selected as beacons, and each of the remaining nodes binds itself to the closest beacon. In this way, a node in S4 is addressed by a concatenation of its own unique identifier and the identifier of its closest beacon. Furthermore, a node maintains a routing entry for every beacon node, $O(\sqrt{N})$ entries, and a routing entry for every nonbeacon node that is closer to the node than to its own beacon, again $O(\sqrt{N})$ entries. This $O(\sqrt{N})$ routing state enables simple, yet small-stretch packet forwarding. If a node has a routing entry for the destination of a packet, the packet is forwarded directly to the destination, yielding a stretch of one. In contrast, if a node does not have an entry for the destination, the packet is first forwarded toward the beacon closest to the destination, but as soon as it reaches a node with a routing entry for the destination, it is redirected toward the destination. In such a case, the routing stretch does not exceed three. While S4 does well at bounding the maximal routing stretch, the $O(\sqrt{N})$ routing state it requires per node may still be significant for very large networks of memory- and bandwidth-constrained wireless nodes. In other words, for some sensornet applications, a different middle ground is more appropriate, that is, one emphasizing smaller state.

Such a small routing state is the objective of the fourth main routing technique, hierarchical routing [Kleinrock and Kamoun 1977; Tsuchiya 1988], which we describe in detail in the next section. In short, hierarchical routing can necessitate as little as $O(logN)$ routing state per node, thereby being extremely appealing for resource-constrained sensornets. However, as we argued in the previous section, to the best of our knowledge, no hierarchical routing protocol has been implemented and evaluated in actual sensornets (cf., Figure 1).

It may be speculated that there have been two major reasons for the lack of sensornet implementations and real-world evaluations of hierarchical routing. First, due to maintaining small routing state, the stretch of hierarchical routing can be large [Krioukov et al. 2007]. Second, hierarchical routing, especially the formation and maintenance of a cluster hierarchy, is a complex problem [Amis et al. 2000; Hagouel 1983], even disregarding the resource constraints of sensor nodes.

We debunk these arguments in the following sections, founding our work on the two following observations. First, even though a logarithmic routing state may indeed result in a high maximal routing stretch in some network topologies, due to their embedding

in physical space and a limited node radio range, the topologies of sensornets are geometric, that is, the average internode distance in such topologies grows quickly with the node population (as $\sim N^{v}$, where $v > 0$ and does not decrease with a growing N). Theory shows that in such topologies, the stretch of hierarchical routing can still be close to one [Funke et al. 2006; Kleinrock and Kamoun 1977]. Second, even though hierarchy formation and maintenance is a complex problem, our recent results with probabilistic algorithms show that it can be solved efficiently [Iwanicki and van Steen 2009a]. Therefore, it may be possible to implement hierarchical routing on resource-constrained sensornets. In summary, not only does our work complement theoretical results on hierarchical routing by filling in the gap in the sensornet evaluations of the routing technique spectrum (cf., Figure 1), but, equally important, it shows that hierarchical routing is a compelling technique in this spectrum.

3. HIERARCHICAL PACKET FORWARDING

Hierarchical routing has many variations, but all of them are based on the same basic principles [Kleinrock and Kamoun 1977; Tsuchiya 1988]. We have analyzed a number of hierarchical routing protocols proposed to date in order to choose addressing and routing algorithms that, in our opinion, are well suited for implementation on resource-constrained sensor nodes. For the sake of brevity, in this article, we narrow our interest to algorithms based on a so-called *landmark hierarchy* [Tsuchiya 1988], commonly known as landmark routing, which we describe next. In addition to delivering the aforementioned properties of hierarchical routing, landmark routing presents some interesting design options and has many proposed protocols for bootstrapping and maintaining the state necessary for routing, which allows us to illustrate various trade-offs. A comparison of algorithms employing different hierarchy types, notably an *area hierarchy* [Kleinrock and Kamoun 1977], can be found in a more detailed account of our research [Iwanicki 2010].

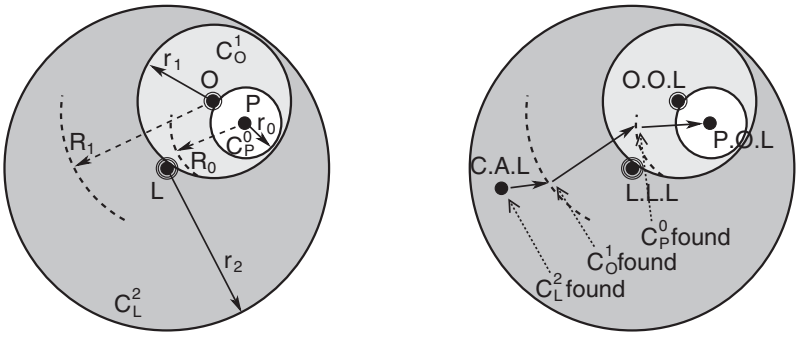
3.1. Basic Terms and Definitions

To support hierarchical packet forwarding, nodes are organized into a *multilevel hierarchy of clusters* based on their connectivity. At level 0, every node belongs to its own singleton cluster. Neighboring singleton clusters are logically grouped into level-1 clusters, which, in turn, are grouped into level-2 clusters, and so on, until there is one or a few top-level clusters that cover the entire network. In effect, at every level of the hierarchy, each node belongs to exactly one cluster.

In its center, each cluster has a *cluster head* (a *landmark*). A level-i cluster is advertised to nodes up to R_i hops away from its head, where R_i depends exponentially on i. A node can be a member of a level-i cluster if it is up to r_i hops away from the head of that cluster, where $r_i \leq R_{i-1}$. For example, $R_i = 2^i$ and $r_i = \lfloor R_i/2 \rfloor$. In this way, clusters at subsequent levels have exponentially growing diameters, and thus, to cover the entire network, $O(logN)$ hierarchy levels are sufficient. Since all nodes have unique identifiers, we use C^i_X to denote a level-i cluster with head node X. A sample cluster hierarchy is depicted in Figure 2(a).

A node's *label* reflects the node's membership in the cluster hierarchy and is a concatenation of the identifiers of the heads of all the clusters the node belongs to at subsequent levels, starting from level 0. In the example from Figure 2(a), the label of node P (a level-0 cluster head) is $P.O.L$, as P belongs to clusters C^0_P, C^1_O, and C^2_L. The label of node O (a level-1 cluster head) is $O.O.L$, and the label of node L (a level-2 cluster head) is $L.L.L$. A node's label acts as the node's routing address.

A node's *routing table* contains entries for all the cluster heads the node receives advertisements from, that is, for each level-i cluster head that is up to R_i hops away from the node, for all levels i. An entry for a cluster head consists of the level and the



(a) Sample cluster hierarchy. (b) Hierarchical packet forwarding.

Fig. 2. Hierarchical routing in a landmark hierarchy.

identifier of the head, the link-layer address of the next-hop neighbor on the shortest path to the head, and the length of this path, as well as some maintenance fields [Iwanicki 2010] which we omit here for brevity. In this way, a node's routing table can contain as few as $O(logN)$ entries that, in addition, are just a few bytes long.

3.2. Hierarchical Forwarding Algorithm

With the preceding organization of node routing addresses and routing tables, packet forwarding can be performed hierarchically: a packet is forwarded toward cluster heads corresponding to the elements of the destination label at subsequently decreasing levels (see Figure 2(b)). More specifically, when a node receives a packet to forward, it searches its routing table for entries that correspond to the cluster heads represented by the elements of the destination label. The forwarding node will always find a routing entry for the head of the destination's top-level cluster: the last element of the destination's label. However, as the packet is routed toward the destination, or even toward that top-level cluster head, subsequent forwarding nodes are also likely to find entries for lower-level label elements in their routing tables. A forwarding node always uses the lowest-level entry found for the elements of the destination label. It forwards the packet to the next-hop neighbor indicated by that entry. Once a node has found a routing entry for a level-i element of the destination label, all subsequent forwarding nodes are guaranteed to find entries for elements at levels lower than or equal to i. In this way, the packet is guaranteed to gradually reach its destination.

A cluster head need not necessarily forward a packet, even if its identifier is one of the elements in the destination label of the packet. It is very likely that before the packet reaches this cluster head, it is redirected toward a lower-level cluster head of the destination node. In Figure 2(b), for instance, node C (label $C.A.L$) initiates routing of a packet to P (label $P.O.L$). The packet is first forwarded toward P's top-level cluster head, L. During this process, however, when the packet reaches a node within R_1 hops from P's lower-level cluster head, O, it is redirected toward O, as the forwarding node must have a routing entry for C_O^1. Likewise, when the packet reaches a node within R_0 hops from P, it is routed toward P, as the node must have a routing entry for C_P^0. In other words, the algorithm does not create highly undesired routing hot spots at cluster heads.

Due to maintaining only a logarithmic number of routing entries per node rather than $O(N)$ entries, hierarchical packet forwarding typically does not use optimal routes. Often when nodes forward a packet, the packet moves toward higher-level cluster

Table I. Sample Design Points and Our Design Decisions

Design Point	Our Choice (and Sample Alternatives)
cluster hierarchy properties	various explored
label synthesis and maintenance (when)	self-organized at runtime (alternative: offline)
label synthesis and maintenance (how)	bottom-up (alternative: top-down)
hierarchy guarantees	probabilistic (alternative: deterministic)
hierarchy update propagation mechanisms	various explored

heads before being redirected toward the destination. This is why routing stretch in hierarchical routing can often be greater than one.

4. IMPLEMENTING HIERARCHICAL ROUTING IN SENSORNETS

The previous hierarchical forwarding algorithm is fairly straightforward. However, being able to use it requires organizing nodes into a cluster hierarchy that determines their labels and routing tables. Constructing a cluster hierarchy is already an NP-complete problem if, for instance, the hierarchy is to be optimal with respect to the routing table size [Hagouel 1983]. Moreover, this is just a first step, as the hierarchy has to be maintained during the whole network lifetime. When the node population or the internode connectivity changes, parts of the hierarchy may have to be repaired to account for the change. Such maintenance can be even more problematic than hierarchy construction. Therefore, the mechanisms for synthesizing and maintaining the node labels and routing tables are fundamental components of any hierarchical routing protocol, as well as of our framework.

For this reason, to develop the framework, we have analyzed a number of well-known hierarchical routing protocols, concentrating on the ones suitable for implementation on resource-constrained low-power wireless devices. The framework encompasses the common elements of the selected protocols and defines various design points where the protocols differ. For brevity, we give just a sample of the design points (see Table I for a summary). We then focus on two selected design points that significantly affect the performance of hierarchical routing. The first one enables exploring the trade-off between state and stretch within hierarchical routing itself. The second one, which has not been studied thoroughly to date, allows for trading off various aspects of robustness.

4.1. Sample Design Points

A sample design decision any hierarchical routing protocol has to make includes the properties of the cluster hierarchy, such as the size, the number, and the organization of clusters at subsequent levels. These properties determine the node routing state and affect the routing stretch. As such, they constitute the first of the aforementioned two selected design points, which we explore in more detail further.

Given some target hierarchy properties, the nodes have to be assigned labels, so that the hierarchy reflected in those labels satisfies the selected properties. The node labels can be synthesized offline prior to deployment [Hagouel 1983; Tsuchiya 1988] or at runtime, using a self-organizing algorithm [Bandyopadhyay and Coyle 2003; Chen and Morris 2002; Du et al. 2004; Iwanicki and van Steen 2009a; Kumar et al. 2000; Subramanian and Katz 2000]. Low-power wireless connectivity is highly unpredictable [Woo et al. 2003], and thus, after the deployment, it may turn out that a pre-constructed hierarchy is invalid because nearby nodes, which have been expected to communicate, cannot hear each other. In addition, reconstructing node labels offline after a cluster head failure may be inefficient in large networks. Therefore, in this article, we focus on runtime label synthesis and maintenance algorithms.

Such algorithms can operate in a bottom-up or top-down fashion. In a bottom-up algorithm, the labels are constructed from level 0 by forming higher-level clusters from

lower-level clusters [Chen and Morris 2002; Kumar et al. 2000; Iwanicki and van Steen 2009a]. In a top-down algorithm, in turn, the labels are constructed from the top level by splitting higher-level clusters into lower-level clusters [Thaler and Ravishankar 1998]. In this article, we concentrate on bottom-up algorithms, as top-down approaches have problems adapting to some topologies, such as those with nonuniform node densities [Kumar et al. 2000].

Label maintenance algorithms can further be divided into deterministic [Hagouel 1983] and probabilistic [Bandyopadhyay and Coyle 2003; Subramanian and Katz 2000], depending on their clustering heuristics. Since running multistep distributed deterministic algorithms on a large network of resource-constrained nodes can be expensive, in this article, we employ probabilistic algorithms.

Such algorithms are typically based on a combination of local label-update operations and update propagation mechanisms. The update propagation and cluster advertisement methods impact the robustness of the hierarchical routing protocol. Therefore, they constitute the second selected design point, which we discuss in more detail shortly.

4.2. Framework Overview

The preceding sample of the design points illustrates that implementing a hierarchical routing protocol involves many subtle decisions. Since we are unable to study all such decisions, we focus just on the two aforementioned ones, which illustrate two important trade-offs that significantly affect the performance of hierarchical routing. To this end, we make all the aforementioned assumptions and present a common protocol for maintaining the node labels and routing tables that was used in the experiments presented in this article.

Principal Operation. A protocol for maintaining the cluster hierarchy reflected in the node labels and routing tables operates in rounds that are local for each node. In every round, a node is allowed to issue (broadcast) one message that advertises the cluster of which the node is the head and propagates any label updates for this cluster. The nodes receiving the message refresh their routing entries for the cluster, adopt any label updates if they are members of the cluster, and possibly rebroadcast the message. We discuss different methods for advertising clusters and propagating label updates in Section 4.4. At the end of its round, each node analyzes its routing table to learn about any changes in the network that have occurred since the last round. If the changes require hierarchy modification, the node updates its label locally. Such a local label update is then propagated to the affected nodes in the node's cluster in subsequent messages issued by the node. This simple mechanism is used both for synthesizing and maintaining node labels.

Hierarchy Construction. Initially, each node is a top-level head of its level-0 singleton cluster. Hence, the node's label consists only of the node's identifier. Whenever a top-level head discovers in its routing table an entry for another cluster head at the same or a higher level, it must either spawn a new higher-level cluster itself or join the higher-level cluster of the other node. In the first case, it would extend its label with its own identifier, promoting itself to a higher-level cluster head. In the second case, it would extend its label with the identifier of the other node. For example, in Figure 2, being initially a level-0 cluster head, node O spawned its own level-1 cluster, C^1_O, by extending its label with its own identifier at level 1 (from O to $O.O$) and effectively promoting itself to a level-1 head. In contrast, P joined its level-0 singleton cluster, C^0_P, to a higher-level cluster, C^1_O, by extending its label with O at level 1 (from P to $P.O$). A similar situation occurred at level 2 for L and O.

Joining an existing cluster is preferred, as it decreases the number of clusters at consecutive levels. However, depending on the distance to the cluster head, joining may

not always be possible. In particular, a level-i cluster head must not join its cluster to a higher-level cluster if the head of the higher-level cluster is more than r_{i+1} hops away.

When no joining is possible, cluster heads must not promote themselves to higher levels at the same time, as this would not guarantee the exponential drop in the number of clusters at subsequent levels. Hence, a head probabilistically defers its promotion by drawing a random promotion time slot, s, and then waiting for s time slots. If within these s time slots other nearby cluster heads promote themselves, the head may join its cluster to one of their clusters; otherwise, the head promotes itself. To ensure that a head deferring a promotion learns about newly spawned higher-level clusters, the time slot at level i is longer than the propagation time of a cluster advertisement from a level-i head, which is proportional to the cluster advertisement radius, R_i.

A cluster head that extends its label, either by spawning its own or joining an existing higher-level cluster, embeds the label update in its subsequent message. In this way, the members of the cluster can also update their labels consistently, and other heads deferring a promotion can learn about the new cluster.

Hierarchy Recovery. When a cluster head has died, it no longer issues messages advertising its cluster. As a result, other nodes do not refresh the routing entries for that cluster head. If a node has not received a cluster advertisement refreshing an entry for a certain number of rounds, the entry is evicted from the routing table. If a level-i cluster head discovers that the entry for its parent level-i+1 cluster head has been evicted, it concludes that its cluster must not be a subcluster of the no longer existing level-i+1 cluster. Consequently, it cuts its label down to level i. Later, by virtue of the preceding hierarchy construction mechanisms, the disconnected cluster of this node will join some other higher-level cluster, thereby restoring the hierarchy.

Label cutting could also be used for rotating cluster heads. While this may be important to balance energy consumption between nodes in applications that use hierarchical clustering for centralized data collection [Bandyopadhyay and Coyle 2003], it is not required from the routing perspective. Neither when routing (cf., Section 3.2) nor when maintaining the routing infrastructure do higher-level cluster heads have to transmit more messages or perform more computations than lower-level heads. Moreover, cluster-head rotation effectively changes node routing addresses, which introduces additional problems for applications. Consequently, cluster-head rotation is not a part of our framework.

While the preceding protocol framework makes particular design choices, it has two properties. First, it captures the common hierarchy maintenance scheme of several hierarchical routing protocols proposed to date (e.g., [Bandyopadhyay and Coyle 2003; Chen and Morris 2002; Du et al. 2004; Iwanicki and van Steen 2009a; Kumar et al. 2000]), and at the same time, makes implementing these protocols feasible on resource-constrained sensor nodes. Second, it unifies the operation of these protocols to enable systematic comparison of different design decisions within this common scheme, in particular, the two selected design points which we discuss next.

4.3. Cluster Hierarchy Properties

The first selected design point we would like to analyze in more depth in this article are the properties of the cluster hierarchy, such as the size, the number, and the organization of clusters at subsequent levels. To begin with, by varying the cluster scaling function in a landmark hierarchy, $R_i = R(i)$, one can control the size and the number of clusters at subsequent levels and, thereby, the number of routing entries per node. In this way, one can explore the state-stretch trade-off within hierarchical routing itself.

Choosing a different hierarchy type, for example, an area hierarchy [Kleinrock and Kamoun 1977] instead of a landmark hierarchy [Tsuchiya 1988], can have a similar

Table II. Different Hierarchy Types in Sample Hierarchical Routing Protocols Proposed to Date

Hierarchy Type	Acronym	Some Protocol Examples
area hierarchy	AH	Kleinrock and Kamoun [1977], Safari [Du et al. 2004], Hagouel [1983], Subramanian and Katz [2000], original PL-Gossip [Iwanicki and van Steen 2009a]
landmark hierarchy	LH	Tsuchiya [1988], SCOUT [Kumar et al. 2000], Bandyopadhyay and Coyle [2003] L+ [Chen and Morris 2002], this article

Table III. Different Hierarchy Information Propagation Methods in Sample Hierarchical Routing Protocols

Method	Acronym	Some protocol examples
periodic hierarchical beaconing	PHB	SCOUT [Kumar et al. 2000], Safari [Du et al. 2004], Subramanian and Katz [2000], Bandyopadhyay and Coyle [2003]
hierarchical distance-vector	HDV	Hagouel [1983], Tsuchiya [1988], L+ [Chen and Morris 2002], PL-Gossip [Iwanicki and van Steen 2009a]
hybrid of the preceding two	Hybrid	proposed in this article

effect. In particular, Table II shows the two common hierarchy types employed by sample hierarchical routing protocols proposed to date.

Finally, in the case of landmark hierarchy there is also a choice between recursive and nonrecursive hierarchy. In a recursive hierarchy, two members of the same level-i cluster are also members of the same level-i+1 cluster [Du et al. 2004; Iwanicki and van Steen 2009a]. In a nonrecursive hierarchy, this need not hold [Chen and Morris 2002; Kumar et al. 2000]. Maintaining a recursive hierarchy is more intricate than a nonrecursive one, as the maintenance mechanisms must ensure that nodes with labels equal at level i also have their labels equal at all levels $j \geq i$. However, this property enables more efficient, per-cluster notifications of label changes. In contrast, in a nonrecursive hierarchy, the notifications must be performed per node.

Although, for the sake of brevity, in this article we concentrate on recursive landmark hierarchies, we do illustrate how different cluster scaling functions, R_i, allow for exploring the state-stretch trade-off within hierarchical routing itself. A more extensive study of the impact of different hierarchies on state and stretch in hierarchical routing can be found in the more detailed account of our research [Iwanicki 2010].

4.4. Hierarchy Information Propagation Methods

The second selected design point we analyze in more depth here is the methods for propagating cluster hierarchy information among nodes. To diffuse label updates (extensions and cuts) and to advertise its cluster to other nodes, in every round each node issues a message. The pattern according to which such messages are issued and the way they propagate hierarchy information determine the latency and the traffic of hierarchy bootstrapping and maintenance. Both these are crucial robustness aspects, but the trade-off between them has not been thoroughly studied to date. Table III presents the major hierarchy information propagation methods.

Periodic Hierarchical Beaconing. Each level-i cluster head periodically issues a beacon message that is flooded in the network up to R_i hops. A beacon message contains the label of the cluster head, a sequence number, and a hop count. A node receiving the beacon refreshes the routing entry for the cluster head, adopts any label updates performed by the head (if it belongs to the head's cluster), and rebroadcasts the beacon if its hop count is smaller than R_i.

Often the inter-beacon interval of a cluster head is proportional to the advertisement radius of the cluster head, R_i. For example, a level-0 head issues an R_0-hop beacon every R_0 rounds, a level-1 head issues an R_1-hop beacon every R_1 rounds and so on. This amortizes the high costs of forwarding higher-level beacons over many rounds but increases the latency of bootstrapping and recovering the hierarchy. To mitigate this increase, a cluster head is also allowed to issue a beacon immediately after it has changed its label locally. We have implemented both variants, that is, with (*PHB[e]*) and without (*PHB[c]*), the exponentially increasing inter-beacon period.

Hierarchical Distance Vector. Nodes run an enhanced distance-vector protocol. At random times in every round, each node broadcasts its state in a heartbeat message that is received by the node's neighbors and is not forwarded by them. The state contains the node's label, routing table, and some consistency data corresponding to the label that allow for propagating label updates. The neighbors receiving the message refresh their routing tables using a standard distance-vector algorithm, ensuring that a level-i routing entry does not travel more than R_i hops. The neighbors can also adopt any fresh label updates performed by the heads of the clusters they share with the heartbeat issuer. Hence, whereas in *PHB*, cluster advertising and update propagation are explicit, by forwarding beacon messages issued per cluster, in *HDV*, it is implicit, by periodically exchanging and merging the local states of neighboring nodes.

If a node's state does not fit in a frame, it has to be fragmented. We have implemented *HDV* with two forms of fragmentation: (1) the MAC layer fragments the message into multiple frames that are sent after a single preamble (*HDV[s]*), and (2) the protocol fragments its state into multiple one-frame messages, each with its own preamble (*HDV[m]*).

Hybrid Approach. Since hierarchical beacons flood the network, they propagate information quickly. However, when issued periodically, they result in myriads of inefficient short transmissions. Moreover, with the exponential beacon issuing interval which reduces the number of transmissions, if a node misses a beacon for a high-level cluster, it may not be able to route to the members of the cluster for a long time.

In contrast, as the routing entries in a heartbeat message advertise many clusters, *HDV* generates lower traffic. In addition, even if in some round a node misses a heartbeat with a cluster advertisement, it will likely receive the advertisement in subsequent rounds. However, since *HDV* propagates information by merging the state of neighboring nodes only once per round, it may take up to R rounds to propagate an advertisement over R hops, which is inferior to *PHB*.

The *Hybrid* approach we propose here combines the advantages of these two methods. In this approach, nodes normally run the *HDV* protocol, thereby generating lower traffic. However, when a cluster head changes its label, for instance, as a result of some failure, it issues a beacon message to rapidly propagate the change among the members of its cluster or to advertise a new cluster. In this way, hierarchy bootstrapping and recovery after failures can be faster.

We study the performance of all the three approaches, including their variations.

4.5. Implementation Remarks

As the implementation platform for our framework, we have chosen TinyOS 2.0 with a standard protocol stack. At the top, an application layer receives routing requests from a PC and periodically reports back the routing state of the nodes and the delivery status of the routing requests, which is done out of band (through a USB cable attached to each node). We have not implemented node label resolution (i.e., obtaining the label of a destination node by a source node) using the wireless communication. Instead, when routing, source nodes obtain destination labels also with out-of-band means, because

obtaining the destination address is not necessarily a responsibility of a routing protocol and is often application specific, as we discuss in Section 8.2.

Below the application layer, our transport layer ensures hop-by-hop delivery of the routed messages. To this end, it employs standard message queuing, link-layer acknowledgments, and retransmissions. When routing, to obtain the link-layer address of the next-hop neighbor, the transport layer contacts the hierarchical routing layer. The hierarchical routing layer corresponds to our framework and is responsible for selecting routing hops and maintaining node labels and routing tables. It makes use of one-hop connectivity information provided by the link quality estimation layer below. In this article, we assume the standard TinyOS 2.0 link estimator based on the exponentially weighted moving average (EWMA) of packet reception rate (PRR) [Woo et al. 2003] and on using link blacklisting, but we have also experimented with the four-bit link estimator [Fonseca et al. 2007]. Likewise, as the MAC layer, we use the standard TinyOS 2.0 CSMA/CA [Polastre et al. 2004].

5. EXPERIMENTAL SETUP

We conducted our experiments mainly in a low-level TinyOS simulator, TOSSIM, and on our 60-node testbed, KonTest [Iwanicki et al. 2008]. In addition, we run several experiments on the publicly available 100+-node testbed, MoteLab [2005]. The TOSSIM experiments aimed at evaluating the scalability of hierarchical routing in realistic networks. TOSSIM is a low-level simulator that incorporates a realistic signal propagation and noise model derived from real-world deployments. Using the tools for this model and real-world data, we generated a number of representative topologies with realistic communication properties.

In all topologies, nodes were placed inside a square area and positioned in a grid, uniform (i.e. perturbed grid) or random fashion, which covers a broad range of node placement strategies, p, from very regular (grid) to completely irregular (random). In addition, to study the protocol scaling properties, we exponentially varied the number of nodes, N, from 64 to 1,024, obtaining diameters of 15–18 hops in 1,024-node networks. To also investigate the impact of node density, ρ, we varied the size of the deployment area, obtaining different node densities from $\sim$11 (sparse) to $\sim$48 (dense) high-quality neighbors per node on average. The resulting network configurations aim to cover many representative practical deployment scenarios.

Using the aforementioned TOSSIM tools and real-world signal-strength traces, for each configuration we generated a realistic connectivity and noise environment. In these environments, there were many asymmetric links and nearby nodes often could not communicate—phenomena that are common in the real world [Woo et al. 2003]. All in all, our TOSSIM results should predict the real-world protocol behavior well.

Finally, to verify how well previous simulations match this behavior, for each configuration, we also generated an environment with a unit-disk connectivity model, and no message loss. In this model, each node has the same circular radio range and can communicate only with the nodes within this range. Hence, unlike in the real world, in the unit-disk model, the connectivity is very regular. To the best of our knowledge, all previous evaluations of hierarchical routing assumed this model.

The aim of the testbed experiments, in turn, was to evaluate hierarchical routing in real sensornets of moderate sizes. The first testbed, KonTest [Iwanicki et al. 2008], consists of 60 TelosB nodes in six office rooms; 53–55 of the nodes were active during the experiments. With the -15 dBm radio transmission power we used, the network diameter oscillated between, four and five hops, and the node density was highly heterogeneous: from 8 to 34 neighbors per node. In addition, there were many asymmetric links and considerable noise during office hours. We thus believe that KonTest can serve as a representative example of a real-world sensornet deployment of a small size.

Moreover, since the experiments we were conducting on KonTest lasted many weeks, the collected data illustrate the long-term behavior of a routing protocol and even contain emergency events, such as an incident in a nearby chemistry lab, followed by an evacuation of the whole building. As such, the data provide noteworthy information on the real-world performance of hierarchical routing, in particular, and point-to-point routing, in general.

In contrast, the experiments on the publicly available MoteLab testbed [MoteLab 2005] were mostly microbenchmarks. MoteLab consists of 190 TelosB nodes dispersed on three floors in the Maxwell Dworkin Laboratory at Harvard University. During the experiments 115–119 nodes were active, and 103–105 of them constituted a single connected network (the remaining nodes were isolated). The network diameter did not exceed six hops, and the node density was again highly nonuniform, from 4 to 30+ neighbors per node. To sum up, the MoteLab experiments should give a relatively realistic indication of the performance of different routing techniques in moderate-size networks.

6. TOSSIM EXPERIMENTS

A routing protocol for sensornets should simultaneously ensure small routing state, small routing stretch, and robustness. In the remainder of this section, we evaluate hierarchical routing with respect to these goals. More specifically, by exploring the two aforementioned selected design points in our framework, we first study the trade-off between routing state and routing stretch offered by hierarchical routing and then the trade-off between various aspects of robustness.

6.1. State-Stretch Trade-Off

6.1.1. Routing State. As a metric of routing state, we use the number of routing entries because the entries dominate the memory consumed by a routing protocol. In our implementation, a routing entry at a node requires eight bytes (for the fields listed in Section 3.1 and some additional maintenance counters) plus 4+ bytes of overhead for a hash table. A routing entry transmitted in a heartbeat message, in turn, is compressed to four bytes.

Figure 3 compares the number of entries in different network configurations. Each data point corresponds to the average or the 99th percentile over all nodes and ten protocol runs resulting in ten different hierarchies, each with the maximal level of five, as this was enough for the employed cluster scaling function, $R(i) = 2^i$, and the considered network diameters.

In accordance with the design goals of hierarchical routing, the routing state scales logarithmically with the network size, N, (left plots), but there is a small constant factor of on average four with respect to $log_2 N$. The state also grows linearly with the network density (right plots), because even though the employed cluster head promotion heuristics can adapt to increasing node density [Bandyopadhyay and Coyle 2003], they are still inherently imperfect, and thus, in denser network, more nodes are promoted to cluster heads, which increases the node routing state.

It is noteworthy that the 99th-percentile and the maximal (not plotted) routing state is lower than twice the average. As a result, when provisioning routing entry memory pools for the maximal or even the 99th-percentile value, one can expect that a node on average utilizes at least 50% of its pool. This, combined with the low maximal and 99th-percentile routing table size, is a strong argument for utilizing hierarchical routing on memory-constrained sensor nodes. Moreover, even a slightly underprovisioned memory pool of a node need not be catastrophic. If the node always prefers entries for higher-level clusters and for the missing lower-level entries, it is allowed to use the local recovery mechanisms developed by Mao et al. [2007]: the routing success rate need

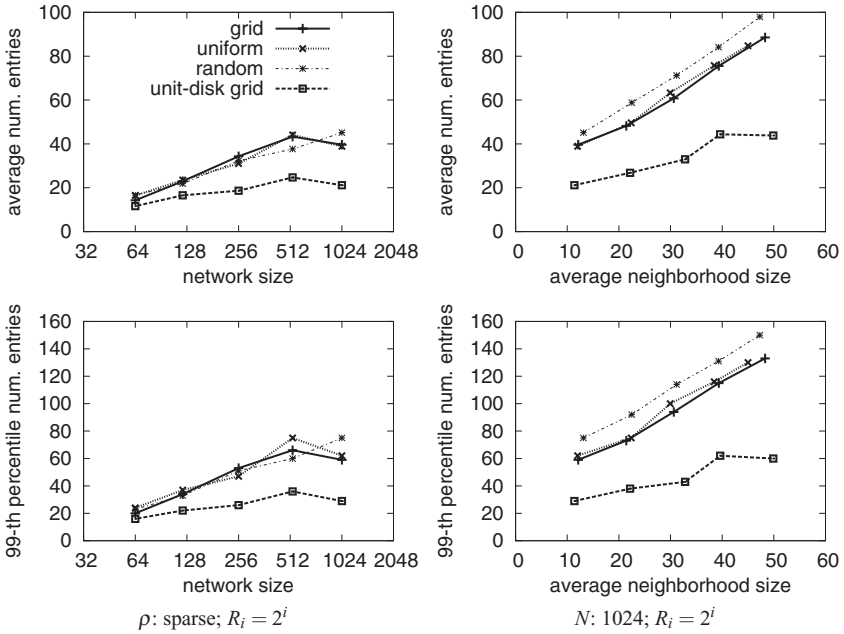


Fig. 3. The average and 99th-percentile routing table size for different network sizes, densities, and topologies.

not suffer, but the routing stretch may grow. All in all, not only can the routing entry memory pools be small in hierarchical routing, also, provisioning the pools need not be much more difficult for hierarchical routing than for other routing techniques.

Finally, the results for the the unit-disk model ("unit-disk grid" in the plots) deviate significantly from the results for realistic low-power communication. In a network with the same number of nodes and a similar density (measured in terms of connectivity rather than physical placement), the average node state in the unit-disk model can be more than three times smaller than in the realistic model. This is mainly due to the aforementioned irregularities in real-world low-power wireless connectivity that further impair the cluster head promotion heuristics. In general, the more irregular the connectivity is, the bigger the routing state. Since the unit-disk model does not exhibit this phenomenon, the results for this model deviate from the results for realistic communication. This reinforces our argument about the need for implementation-based evaluation of hierarchical routing, as presented in this article.

6.1.2. Routing Stretch. We measure routing stretch with a standard metric [Fonseca et al. 2005; Kim et al. 2005; Mao et al. 2007]: the *hop stretch* (*dilation*). The hop stretch of a routing path between two nodes is the ratio of the number of hops on this routing path to the number of hops on the shortest path between the two nodes in the internode connectivity graph. A *hop* is defined over a wireless link with at least 55% bidirectional packet reception rate, as measured by the employed link estimator. Figure 4 depicts the hop stretch between all pairs of nodes in the hierarchies from Figure 3.

Despite a logarithmic number of routing entries, hierarchical routing offers small hop stretch that scales gracefully with the network size (left plots). Moreover, the hop stretch does not grow with the increase in node density (right plots), which is a consequence of the aforementioned growth in the routing table size (cf., Figure 3, right plots).

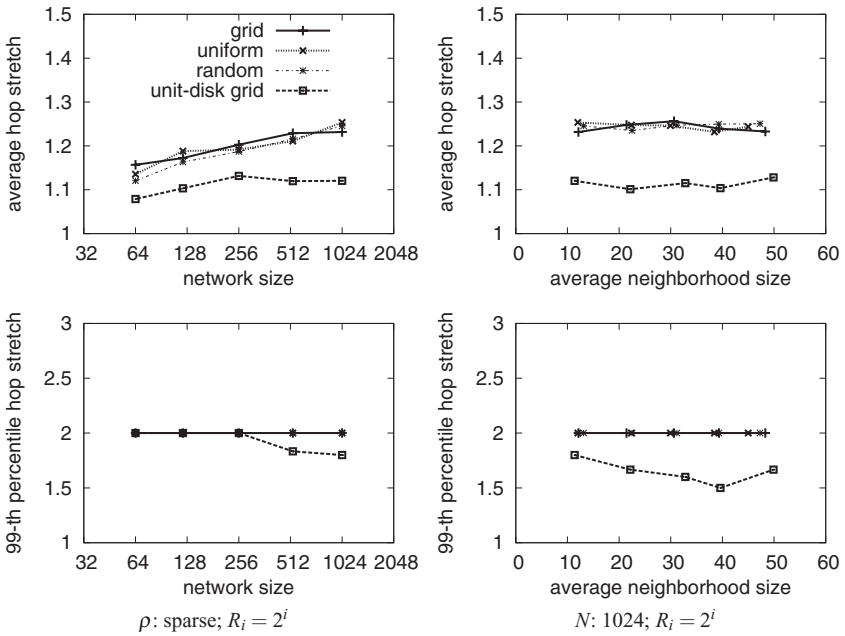


Fig. 4. The average and 99th-percentile hop stretch for different network sizes, densities, and topologies.

Although the theoretical upper bound on the hop stretch between two nodes in hierarchical routing is large [Krioukov et al. 2007], Figure 4 shows that more than 99% of routing paths in our experiments do not exceed a hop stretch of two (bottom plots). Moreover, the worst path we have obtained throughout a few years of experiments, had a hop stretch of six. This confirms our observation from Section 2 that although one can presumably construct a hierarchy with a high-hop-stretch path if arbitrary network topologies are allowed to be considered, due to their geometric nature, the topologies of sensornets allow for hierarchies with low-hop-stretch paths [Funke et al. 2006]. In addition, our framework is able to construct such hierarchies with very high probability. All in all, despite only logarithmic routing state, hierarchical routing can provide reasonable hop stretch bounds in practice.

Like before, the results for the unit-disk communication diverge significantly from the results for the realistic communication. In the unit-disk model, the hop stretch is much lower than in the realistic model, even though the routing tables in the realistic model are four times bigger than in the unit-disk model (cf., Figure 3). This is also a consequence of the aforementioned connectivity irregularities and again evidences that in sensornets, practice often diverges from theory.

To get more insight into the hop stretch values offered by hierarchical routing, let us consider Figure 5. The figure illustrates the relationship between the length of the shortest possible path between two nodes and the hop stretch of a corresponding path offered by hierarchical routing in the sparse (left) and dense (right) 1,024-node randomly deployed networks from Figure 4. It can be observed that routing paths between nearby nodes, that is, nodes within few physical hops, tend to have a larger stretch than routing paths between nodes farther apart. In particular, the paths between nodes that are within seven physical hops in the sparse network and two physical hops in the dense network have their stretch above the average (the dotted horizontal line). In

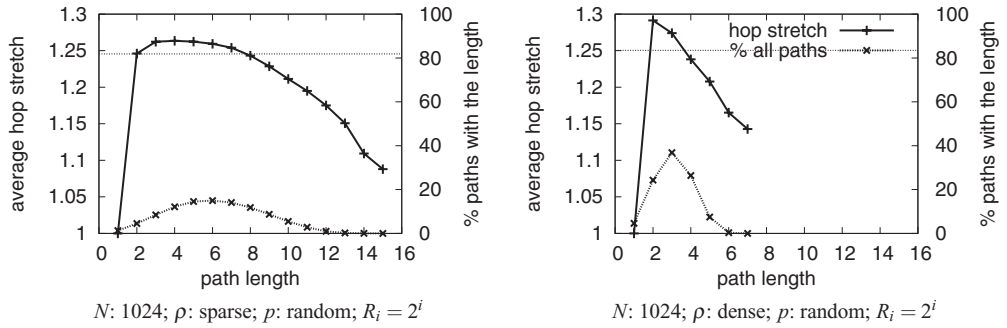


Fig. 5. The average hop stretch of a path vs. the length of the path for different network densities.

contrast, the stretch between more distant nodes is smaller and drops quickly with the internode distance.

This illustrates a so-called *boundary effect* in hierarchical routing [Mao et al. 2007]: two nodes, A and B, that are physically close can belong to different clusters; in such a case, a packet routed from A to B is initially forwarded toward the head of B's cluster rather than directly to B, which can dilate the packet's routing path. This effect is a negative consequence of the routing state aggregation, as achieved by clustering (recall the state-stretch trade-off). In geometric networks, however, as the internode distance grows, the negative effect of the state aggregation for a given hierarchy level diminishes: a packet forwarded toward the head of a distant cluster containing the destination is more likely also to travel in the direction of the destination itself. Consequently, even though short paths can exhibit a larger hop stretch, the longest routing paths, which are also the most expensive, have the smallest hop stretch, that is, their cost is nearly optimal. In other words, the efficiency of hierarchical routing grows with the growth of the path cost.

Finally, since the hop stretch is a metric from wired and unit-disk networks which assume virtually no per-hop message loss, it may not fully reflect the actual number of routing transmissions in sensornets, which in contrast exhibit high message loss. For example, a routing path may consist of few hops but over poor, lossy links, resulting in many (re)transmissions when routing a message over this path. To quantify the discrepancy between the hop stretch and the actual transmissions, we use the standard metric from sensornets [Fonseca et al. 2005; Kim et al. 2005; Mao et al. 2007]: the *transmission stretch*. The transmission stretch of a routing path between two nodes is the ratio of the number of transmissions to deliver a message using the path to the number of hops on the path. When measuring the impact of hop selection on the transmission stretch, we minimized the message loss due to collisions and congestion by routing only one message at any given moment.

The transmission stretch results (not plotted) indicate that in our framework, the discrepancy between the number of hops and actual transmissions is low; the average transmission stretch is $\sim$1.02 and the 99th percentile is not greater than 2, because the framework uses neighbor tables that are large enough to allow each node to select only high-quality links as routing hops. Thus, due to the bimodality of wireless links [Srinivasan et al. 2006], even though a hop is defined over a link with $\geq$55% packet reception, most of the hops in fact exhibit nearly 100% packet reception, which effectively minimizes the transmission stretch. Overall, the results for the hop and transmission stretch indicate that despite only logarithmic state, hierarchical routing may perform reasonably well in practice, which makes it particularly suitable for large networks of memory-constrained wireless embedded nodes.

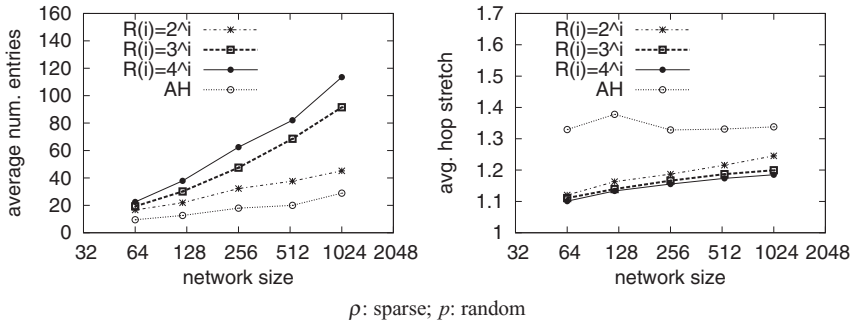


Fig. 6. The influence of the hierarchy type and properties on the state and stretch of hierarchical routing.

6.1.3. Exploiting the State-Stretch Trade-Off. Moreover, by changing the hierarchy properties, one can exploit the trade-off between state and stretch within hierarchical routing itself. For example, Figure 6 shows how by varying the cluster scaling function in a landmark hierarchy, $R(i)$, or the hierarchy type (an area hierarchy instead of a landmark hierarchy), one can reduce stretch for an increase in state and vice versa. The more detailed account of our work [Iwanicki 2010] contains a more extensive study of this phenomenon. To sum up, not only does hierarchical routing offer a state-stretch trade-off that is appealing for many sensornet applications, but also this trade-off can be tuned further to best suit a particular application.

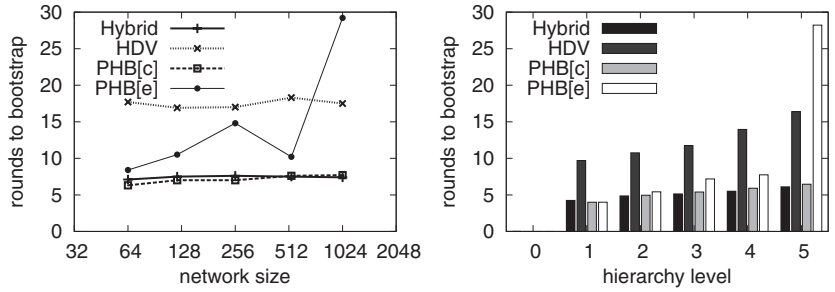
6.2. Robustness Trade-Offs

Apart from offering small state and small stretch, a routing protocol for sensornets should be robust: it should handle changes in the network with minimal traffic and latency. This requirement also entails a trade-off between the volume of maintenance traffic and the latency of reacting to changes in the network. Intuitively, the more traffic nodes can exchange in a time unit, the faster they can react to changes in the network. In the remainder of this section, we study how different methods for propagating hierarchy information exploit this trade-off when bootstrapping and recovering the hierarchy.

To this end, we make use of the fact that all these considered methods operate periodically in rounds. We thus fix the duration of a round for all the methods and study their performance in terms of rounds. Such an approach is accurate when a round is a few orders of magnitude longer than a message transmission. This is typically the case as, in low-data-rate sensornet applications, the rounds are measured in the order of minutes (five minutes in our experiments), while a message transmission takes in the order of milliseconds.

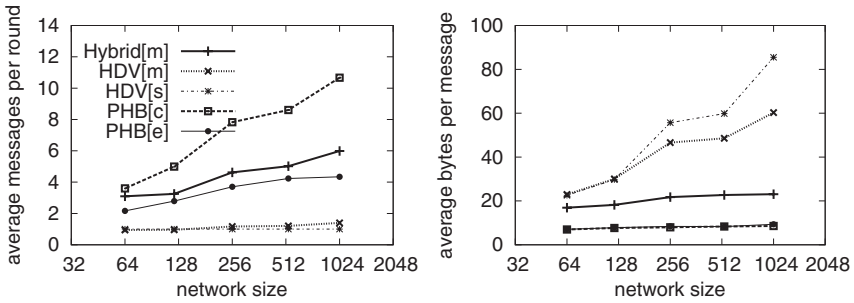
6.2.1. Hierarchy Bootstrap. Figure 7 presents the number of rounds that different hierarchy information propagation methods require to bootstrap the hierarchies from Figure 3 and 4. In these experiments, we started all nodes simultaneously and let them construct a hierarchy. We considered the hierarchy as being bootstrapped when 99% of the nodes had their labels assigned and could successfully route to each other.

The hierarchy information propagation methods that offer the fastest bootstrap are the periodic hierarchical beaconing with beacons issued in every round irrespective of the cluster-head level (*PHB[c]*) and the novel hybrid approach (*hybrid*). In these methods, the number of bootstrap rounds is directly proportional to the maximal hierarchy level, five (provided that neighbor discovery and link estimation can be performed within 1–2 rounds). Essentially, in every round, a single hierarchy level is constructed



ρ: sparse; p: random — N: 1024; ρ: sparse; p: random; $R_i = 2^i$

Fig. 7. The bootstrap latency for different network sizes and hierarchy information propagation methods.



ρ: sparse; p: random; $R_i = 2^i$

Fig. 8. The per-node bootstrap traffic for different network sizes and information propagation techniques.

(the right plot). This corresponds to the lower bound in our framework, and thus, these two methods are optimal with respect to the bootstrap time.

In theory, the periodic hierarchical beaconing with the exponential beacon issuing pattern (*PHB[e]*) should perform like these two methods. In practice, however, it does not due to message loss. If a node misses a beacon message from a level-i cluster head, it and potentially some of its neighbors would not record a routing entry for the head for R_i rounds which boosts the bootstrap time. In Figure 7, this can be observed at level 4 ($R_i = 16$), at which a missed beacon from a level-4 cluster head delayed hierarchy construction at level 5 for 16 rounds. This can also happen in the two optimal methods. However, in *PHB[c]*, the unlucky node would likely receive a beacon in the next round, whereas in *Hybrid*, the unlucky node would recover through heartbeat messages.

Finally, the hierarchical distance vector (*HDV*) is the slowest. Since in this method information is propagated through periodic state merging once per round, in the worst case, it may take R rounds to advertise a cluster over R hops. This also requires extending the slot duration in the cluster head promotion heuristics. As a result, the bootstrap time depends mostly on the network diameter (15 hops in the figure).

With the message cost of bootstrapping, the relationship between the techniques is opposite, as depicted in Figure 8 (left plot). A node running hierarchical distance vector sends one message per round if a message can consist of a few frames (*HDV[s]*), or a logarithmic number of messages with a tiny constant if the routing state is manually fragmented into one-frame messages (*HDV[m]*). Likewise, other methods generate logarithmic traffic. The differences in constants associated with the logarithms, however, can be substantial (e.g., a factor of ten between *PHB[c]* and *HDV[m]*).

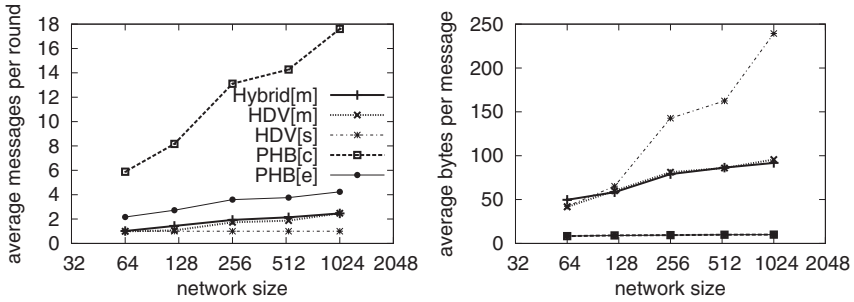


Fig. 9. The per-node stable traffic for different network sizes and information propagation methods.

For efficiency, message payloads should be maximized: a protocol should preferably send fewer but longer messages. This is important in sensornets because message transmission or reception typically involves a large energy overhead due to synchronizing the transmitter and receivers to have their radios on. The right plot in Figure 8 presents the efficiency of different hierarchy information propagation methods. Beacon messages are small (ten bytes in our implementations), and thus, protocols based on periodic hierarchical beaconing (*PHB*) are inefficient when bootstrapping the hierarchy. In contrast, heartbeat messages in hierarchical distance-vector protocols (*HDV*), propagate information more efficiently, as each heartbeat contains the whole routing state of a node. Our hybrid method, which combines beacons and heartbeats, lies in between. In the bootstrap phase, it resembles more the beaconing protocols, as during this phase, the hierarchy changes, hence beacons dominate over heartbeats.

6.2.2. Hierarchy Maintenance. After the hierarchy has been bootstrapped, the protocols continue to maintain it during the system lifetime. Such maintenance generates traffic, which again depends on the hierarchy information propagation method, as depicted in Figure 9.

During maintenance, like during bootstrap, the hierarchical distance-vector protocols (*HDV*) send the fewest and the longest messages, while the periodic hierarchical beaconing protocols (*PHB*) send the most and the shortest messages. For instance, the difference in the number of messages sent per round between *PHB[c]* and *HDV[s]* is nearly a factor of 18. Finally, since in the stable phase (unlike in the bootstrap phase), the *Hybrid* protocol generates only heartbeat messages, it performs similarly to the protocols using *HDV*.

The maintenance traffic is necessary for detecting and repairing failures of the hierarchy. To measure the failure recovery latency for every hierarchy information propagation method, we performed the following microbenchmarks. For each method, after the hierarchy had been bootstrapped, we killed a single node and measured the time required to recover the hierarchy. By recovery, we mean a state in which neither the label nor the routing table of any alive node contains the identifier of the failed node (i.e., the information about the failed node had been completely removed from the network), and all the alive nodes could successfully route to each other. Afterward, we reincarnated the dead node and let it fully rejoin the system (the rejoining latency was small and thus is omitted in the results). We then repeated these steps for all other nodes in the network. Figure 10 presents the average results depending on the level of a failed node as cluster head.

Periodic hierarchical beaconing with beacons issued in each round irrespective of the issuer's level (*PHB[c]*) performs best. Since in this method, each routing entry is

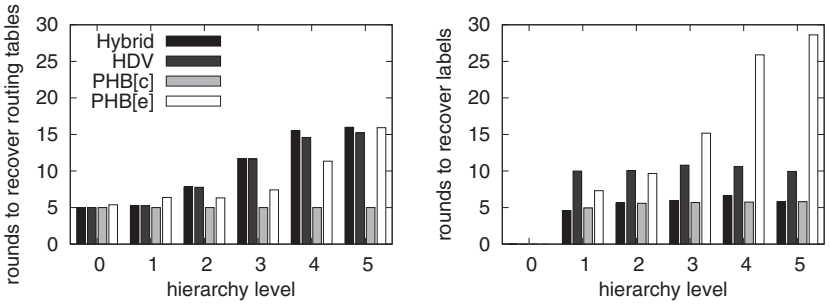


Fig. 10. The average recovery latency of routing tables and labels depending on the failed node's level as cluster head.

refreshed in every round, detecting a cluster head failure is fast, irrespective of the head's level (left plot). In our experiment, we assumed that an entry is considered dead if it is not refreshed for four consecutive rounds. Because *PHB[c]* also constructs the hierarchy quickly (cf., Figure 7), it recovers node labels most quickly (right plot), as label synthesis and recovery in the framework use the same mechanisms (cf., Section 4.2). In contrast, in the *HDV* and *Hybrid* approaches, detecting a failure of a cluster head is proportional to the distance to the head. Hence, these methods repair node routing tables and labels slower. *Hybrid*, however, is more efficient than *HDV* in label recovery (right plot), because it constructs the hierarchy much faster (cf., Figure 7). Finally, in periodic hierarchical beaconing with exponential inter-beacon interval (*PHB[e]*), the time to detect a cluster head failure is proportional to the head's advertisement radius, R_i, which for most of the nodes is longer than the actual distance to the head. Consequently, *PHB[e]* is the slowest method.

It is important to note that these results present the *total recovery time* of all nodes affected by a level-i cluster head failure, which is much longer than the average recovery time of a node. Moreover, as most nodes are only level-0 cluster heads, a failure of a random node requires few and local-only repair activities. With some redundancy in the next-hop candidates for routing entries, routing is virtually undisrupted by a failure of a level-0 head.

6.2.3. Exploiting the Robustness Trade-Offs. To sum up, the three evaluated methods for propagating hierarchy information differ in their robustness. For a given round length, periodic hierarchical beaconing with one-round inter-beacon intervals (*PHB[c]*) bootstraps and recovers the hierarchy fastest but uses myriads of inefficient short messages. As such, it is most suitable for applications in which heavy traffic is less important than quick construction and recovery of the routing infrastructure. In contrast, protocols based on hierarchical distance vector (*HDV*) generate the lowest traffic with the lowest energy overhead on transmitted protocol data, but they take time to construct and recover the hierarchy. Consequently, they are more appropriate for applications that operate on tighter energy budgets but can tolerate long periods of disruption (e.g., delay tolerant systems).

The novel *Hybrid* approach may be a good alternative for both these techniques. It offers optimal hierarchy bootstrap like *PHB[c]*, uses mostly low and efficient traffic like *HDV*, and recovers after failures relatively quickly. Moreover, the only issue that slows recovery in *Hybrid*, as compared to *PHB[c]*, is the slow failure detection mechanism inherited from *HDV*. In some applications, however, failure detection can be improved with an ICMP-like protocol. If hierarchical routing encounters a routing error, it can

return a type-3 ICMP message ("Destination Unreachable") to the source, which then marks a given cluster as failed, yielding almost immediate failure detection.

Finally, periodic hierarchical beaconing with the exponential beacon issuing pattern (*PHB[e]*) in practice offers neither fast bootstrap and recovery nor efficient traffic. Thus, this technique is unappealing for real-world applications. Yet, surprisingly many proposed hierarchical routing protocols are based on this technique [Bandyopadhyay and Coyle 2003; Du et al. 2004; Kumar et al. 2000], which again reinforces our initial argument for the need for implementation-based evaluation of hierarchical routing.

Apart from the preceding microbenchmarks, we have conducted numerous experiments with other failure scenarios. For example, one of the testbed experiments presented in Section 7 involves a simultaneous massive failure of 50% of nodes. Other scenarios, many of which can be found in the more detailed account of our research [Iwanicki 2010], involved constantly changing node populations, massive correlated failures, and network partitions. In general, the protocols are able to deal with such failures, and the trade-offs entailed by the repair activities are consistent with the ones just discussed.

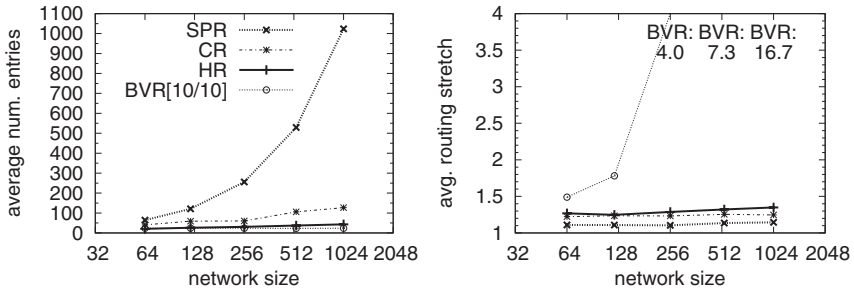
6.3. Hierarchical Routing vs. Other Techniques

Hitherto, we have seen how hierarchical routing satisfies the three requirements for a sensornet point-to-point routing protocol: small routing state, small routing stretch, and robustness. We will now proceed to study how the performance of hierarchical routing compares with the performance of other routing techniques.

We conducted comparative experiments using a general point-to-point routing library [Iwanicki and Azim 2010], which we had codeveloped together with authors of the TinyOS 2.0 implementation of compact routing, the main competing routing technique for sensornets (cf., Section 2). Overall, the library encompasses four routing techniques—hierarchical routing (HR), compact routing (CR), shortest-path routing (SPR), and beacon vector routing (BVR)—that together represent the entire state-stretch trade-off spectrum (cf., Figure 1). It implements these four techniques in a highly uniform manner. In particular, it decomposes a routing protocol into components corresponding to well-recognized routing abstractions, such that in effect, the great majority of its code is shared by all the techniques. Since the hierarchical routing framework described in this article also employs well-recognized routing abstractions, integrating it into the library was relatively straightforward, requiring as little as 8.4% hierarchical-routing-specific code. By and large, the library allowed us to conduct systematic experimental comparisons of routing techniques from the entire spectrum.

The setup for those comparisons was largely the same as for the experiments presented in the previous sections (cf., Section 5). In short, we tested the four routing techniques in a number of networks with different sizes, densities, and node placement strategies, and under various failure scenarios. Moreover, we tested various protocol configurations, as we wanted to make the comparison as fair as possible.

6.3.1. State-Stretch Trade-Off. Like in the previous experiments, the first trade-off we analyze is the one between routing state and stretch. Figure 11 presents the state (left) and the stretch (right) for each of the four techniques in a sample representative experiment that involved randomly deployed networks with exponentially growing sizes. For hierarchical routing, the standard cluster scaling function was used, $R_i = 2^i$. The numbers $[B/K]$ for beacon vector routing denote, respectively, the total number of beacons, B, and the number of beacons used in a node's routing address, K (cf., Section 2). The [10/10] configuration has been chosen to make the comparisons as fair as possible; other configurations are discussed in the more detailed account of our work [Iwanicki 2010]. Note also that unlike in previous figures, routing stretch in



N: 1024; ρ: sparse; p: random; for HR $R_i = 2^i$

Fig. 11. The state-stretch trade-off in the spectrum of routing techniques.

Figure 11 is measured as a product of hop stretch and transmission stretch and, hence, simply represents the ratio between the total number of transmissions necessary to deliver a packet and the minimal possible number of hops between the packet's source and destination nodes. Combining hop stretch with transmission stretch into a single metric is necessary, because hop stretch alone does not capture the large cost of the scoped flooding fallback mechanism used by BVR (cf., Section 2).

To begin with, even though due to occasional retransmissions, its stretch is in practice greater than one, shortest-path routing (SPR) ensures the smallest possible stretch (right plot). This is one of the reasons why it is the dominant intra-domain routing technique in the Internet, where the main objective of routers is reliable and timely message delivery, and the memory and bandwidth available to fulfill this objective are plentiful. In contrast to Internet routers, however, sensor nodes are typically severely constrained in terms of memory and bandwidth. Consequently, the linear dependency of routing state on the node population size (left plot) precludes practical deployments of shortest-path routing in sensornets beyond some tens of nodes.

On the other end of the state-stretch spectrum, there is beacon vector routing (BVR). In theory, it offers a constant state (left plot) and, thus, should be well suited for resource-constrained sensor nodes. In practice, however, if the state is too small (too few nodes are selected as beacons), the stretch of BVR grows substantially (right plot), as the protocol has to use expensive scoped flooding to deliver packets (cf., Section 2). Moreover, the state of BVR is small only when measured in terms of the number of routing entries. In contrast, when measured in bytes (not plotted), the state is much larger, because the size of a single routing entry in BVR is not constant, like in the other considered techniques, but grows proportionally to the number of beacons. Therefore, while the original work on beacon vector routing [Fonseca et al. 2005] was the first to address the need for a point-to-point routing protocol for sensornets, compared to the other techniques, beacon vector routing seems rather unattractive. We elaborate on these arguments in the more detailed account of our research [Iwanicki 2010].

The two techniques that seem most practical for sensornets from the state-stretch trade-off perspective are compact routing (CR) and hierarchical routing (HR). Although they have the same objectives, that is, significantly reducing routing state while increasing stretch only slightly, their accomplishment of these objectives differs. Hierarchical routing offers a substantially smaller state than compact routing. This means that it requires less memory to store the state and less traffic to maintain it, thereby allowing for scalability of an order of magnitude better. Compact routing, in contrast, offers a slightly smaller stretch, with the difference being in the order of 10–15% compared to hierarchical routing. This means that packets routed using this technique may exhibit slightly lower end-to-end delays and slightly higher delivery rates, and

Table IV. Basic Properties of the Two Employed Experimental Testbeds

Metric	KonTest	MoteLab
connected active nodes (N)	53–55	103–105
diameter [hops]	4–5	5–6
density (ρ) [neighbors]	8–34 (avg.: 17.48)	4–30$^+$ (avg.: 14.10)

globally, they may generate slightly less traffic. The choice between hierarchical and compact routing will thus likely depend on the particular application: applications that route lots of packets would preferably use compact routing, whereas applications that require large networks would likely employ hierarchical routing instead. Nevertheless, in sensornets, a large reduction in state for a small increase in stretch is often more appealing. For this reason, hierarchical routing is indeed worth considering, at least from the perspective of the state-stretch trade-off.

6.3.2. Robustness Trade-Offs. Apart from the preceding experiments related to the state-stretch trade-off, we conducted studies on the robustness trade-offs within the four selected representative routing techniques. The results of these experiments are similar to the results for hierarchical routing presented in Section 6.2; consequently, due to space constraints, we do not present any plots here.

In short, depending on the method used to populate and maintain node routing tables (e.g., beaconing, distance vector, or hybrid), a routing protocol can minimize either the latency of bootstrapping and maintaining the node routing state or the volume of traffic necessary for the maintenance. Like in hierarchical routing, in the other routing techniques, the volume of the maintenance traffic is directly proportional to the amount of state the nodes maintain, which reinforces the argument about the need for a minimal routing state. Moreover, in all the techniques, a failure of a node has on average only a local impact and requires few repair activities. Finally, many local failure recovery heuristics are applicable to either of the techniques. In general, any of the selected four routing techniques can be implemented in a way that emphasizes a particular robustness metric, such as the latency of reacting to changes in the network or the volume of the maintenance traffic. In other words, the techniques offer a comparable performance when it comes to robustness.

7. TESTBED EXPERIMENTS

In addition to the preceding low-level simulations, we have conducted numerous testbed experiments, which we divide into two classes. The first class corresponds to microbenchmark experiments that evaluated the trade-offs offered either by different design decisions within hierarchical routing itself or by different point-to-point routing techniques. The microbenchmarks were run on both the testbeds mentioned in Section 5: KonTest and MoteLab. The second class, in turn, represents experiments that assessed the long-term behavior of the hierarchical routing protocols offered by our framework. The long-term experiments were conducted solely on KonTest. Table IV summarizes the properties of the two testbeds.

7.1. Micro-Benchmarks

The objective of the microbenchmark experiments was to validate the low-level simulation results in actual sensornets. Figure 12 presents the routing state and routing stretch obtained for our hierarchical routing protocols on KonTest with 55 active nodes.

The figure illustrates that the size of the node routing tables (left plot) can be small in real-world settings. In the figure, a node stores 5.11 routing entries on average and seven entries in the worst case; in general, the worst-case routing state we have observed on KonTest involved 14 entries, which is even better than the results from TOSSIM—a phenomenon we attribute to the small scale of KonTest. In addition, like

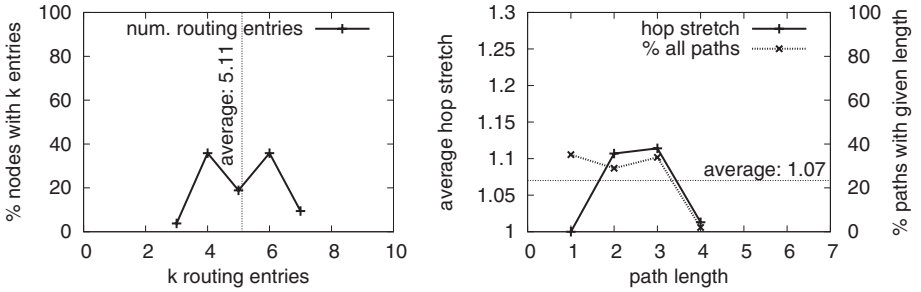


Fig. 12. The state (left) and stretch (right) of hierarchical routing on KonTest.

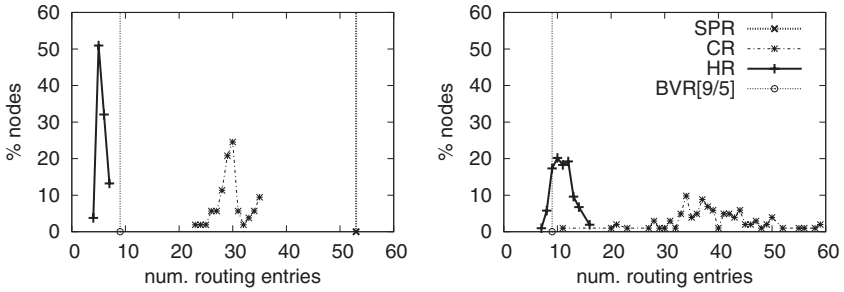
Table V. Convergence Time and Traffic for Different Hierarchy Information Propagation Methods on KonTest

Technique	Bootstrap Time		Stable-State Messages Per Node Per Round
	Rounds	Minutes	
PHB[e]	24	12:00	2.07413
HDV	19	9:30	1.00000
Hybrid	10	5:00	1.00036

in the TOSSIM experiments, the worst-case state does not exceed twice the average, which facilitates provisioning memory pools for the node routing tables. Similarly, the hop stretch (right plot) is small. In the figure, the average hop stretch is 1.07; overall, it has not exceeded 1.15 on average and 2.66 in the worst case (not shown). In addition, like in the TOSSIM experiments, the hop stretch decreases with the physical distance between nodes, albeit this phenomenon is barely visible due to the small scale of the experiment.

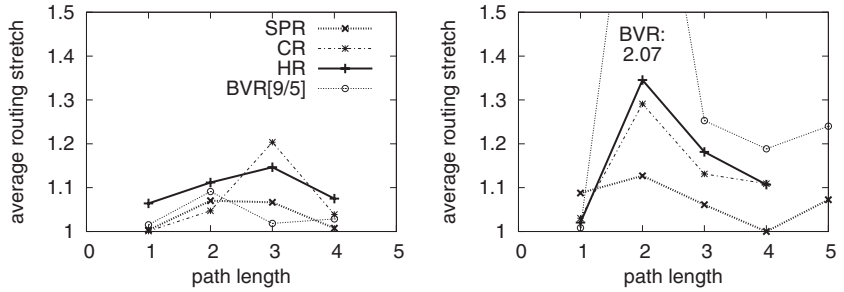
Table V illustrates differences between three selected methods of propagating cluster hierarchy information when run on KonTest with 55 active nodes. Like in TOSSIM, the *Hybrid* method offers the fastest hierarchy bootstrap, while the hierarchical distance vector (*HDV*) performs much slower (cf., Section 6.2). The differences in bootstrap time, however, are less pronounced than in TOSSIM, as *Hybrid* seems to be slower on KonTest (ten rounds) than in TOSSIM (5–6 rounds, cf., Figure 7). This is simply a consequence of the fact that we were unable to start all the testbed nodes simultaneously. As in the simulations, both *Hybrid* and *HDV* generate relatively low traffic. In contrast, the popular periodic hierarchical beaconing with an exponential beacon issuing interval (*PHB[e]*) is again the slowest method and requires more messages. Finally, the table also shows the actual physical time necessary to bootstrap the cluster hierarchy. With the round length of 30 seconds, bootstrapping lasts a few minutes, which is longer than, for example, bootstrapping a data-collection tree [Hui and Culler 2008]. This, however, is to be expected, because a multilevel cluster hierarchy is a more complex structure than a flat tree. Moreover, we confirmed that one can shorten the round to a few seconds without significantly degrading performance and are working on further reducing the maintenance latencies.

We conducted similar experiments to compare the performance of hierarchical routing (HR) against the performance of the other three routing techniques from Section 6.3, namely shortest-path routing (SPR), compact routing (CR), and beacon vector routing (BVR). To begin with, Figure 13 presents the distribution of the routing state size across the node population on KonTest with 53 active nodes (left) and MoteLab with 104 active nodes (right).



T: KonTest; N: 53 — T: MoteLab; N: 104 (SPR: 104 — omitted)

Fig. 13. The routing state of different routing techniques on the two testbeds.



T: KonTest; N: 53; 90 node pairs — T: MoteLab; N: 104; 90 node pairs

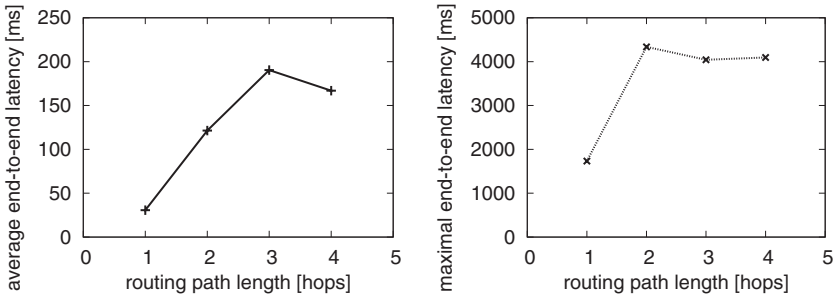
Fig. 14. The routing stretch of different routing techniques on the two testbeds.

As a constant-state routing technique, BVR offers the smallest routing state but, as explained in Section 6.3.1, only in terms of the number of routing entries per node (nine on both KonTest and MoteLab).[1] When measured in bytes (not plotted), the routing state in BVR is actually much larger than the state in any other routing technique, both on KonTest and MoteLab. HR also offers a very small routing state but, in contrast to BVR, not only in terms of the number of entries per node (on average, 5.5 on KonTest and 10.92 on MoteLab) but also in terms of bytes. The state in CR is relatively small as well (on average 29.74 on KonTest and 38.71 on MoteLab) but, nevertheless, larger than the state in HR. Finally, the state in SPR is proportional to the size of the entire node population (53 entries on KonTest and 104 on MoteLab) and, hence, the largest. Overall, the state results obtained on the testbeds are consistent with the corresponding results from TOSSIM (cf., Section 6.3).

Figure 14 presents the corresponding results for routing stretch. Again, to take into account the scoped flooding mechanism of BVR, routing stretch is measured as a product of hop stretch and transmission stretch (cf., Section 6.3.1).

By and large, the routing stretch results are also consistent with that of TOSSIM—the larger the state of a routing technique, the smaller the stretch. The few small differences and irregularities can again be attributed to the small scale of the testbeds. For example, the stretch in BVR is relatively small on KonTest compared to the other techniques, because small networks do not trigger the expensive scoped

[1]The [9/5] rather than the [10/5] variant of BVR is an artifact of the beacon election mechanisms in the implementation of BVR employed in the experiments. More specifically, when electing beacons, the mechanisms have a tolerance of 10% in our experiments, yielding nine beacons rather than ten most of the time. However, nine (or even five) beacons instead of ten are still sufficient considering the network size (cf., Section 6.3.1).



T: KonTest; *N*: 53; 90 flows; 1 flow at a time; 1 packet per second

Fig. 15. The end-to-end routing latencies depending on the path length for hierarchical routing on KonTest.

flooding mechanism of BVR. In general, however, the trade-offs between routing state and stretch that we studied with TOSSIM in the previous section are similar to the trade-offs encountered with real hardware as demonstrated in this section. In particular, the average routing stretch of SPR, CR, HR, and BVR[9/5] in Figure 14 are, respectively, 1.051, 1.158, 1.198, and 1.507 on MoteLab, which (considering the state-stretch trade-off) is in line with the routing state displayed by these techniques (cf., Figure 13). All in all, the testbed experiments indicate that TOSSIM can simulate low-power wireless communication relatively well.

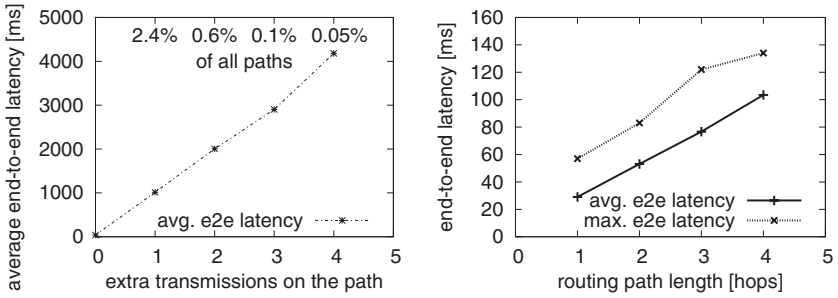
7.2. Long-Term Experiments

Yet, TOSSIM cannot accurately model the dynamic, random interactions of a network with the surrounding environment. Examples of such interactions include wireless noise generated by 802.11 laptops and changes in signal propagation due to mobility in the surrounding environment. These interactions, however, make the internode connectivity dynamic and hence impact the performance of a routing protocol. Wireless noise, for instance, makes some wireless links bursty [Srinivasan et al. 2008]: such links display short periods of perfect or null packet reception. Mobility, in turn (e.g., repositioning office furniture by just half a meter), can change wireless links considerably and more permanently.

Such dynamic changes in connectivity impair message delivery rates and transmission stretch. A next routing hop for which the link quality has been estimated as high using the maintenance messages (i.e., heartbeats or beacons) may deteriorate when the actual data are routed. To alleviate this, our framework allows for redundancy in the next-hop candidates for each routing entry. Moreover, the applications using the framework can employ a link estimator that considers not only the maintenance traffic but also the actual routed data traffic [Fonseca et al. 2007]. Such mechanisms have been shown sufficient in achieving end-to-end message delivery rates of about 98% [Hui and Culler 2008]. The results obtained by us are largely similar and thus are omitted here for brevity.

Changes in connectivity also affect the end-to-end packet latencies. Figure 15 depicts the average (left) and the maximal (right) end-to-end latency of a packet routed over a path depending on the length of this path. The experiment involved one flow at a time with a rate of one packet per second. In total, 90 flows between different source-destination pairs were tested.

While the figure shows that for the average end-to-end latency, there is some correlation with the path length, there seems to be no correlation for the maximal latency. This is due to retransmissions that occur occasionally but that significantly impact the end-to-end latencies. More specifically, to alleviate the effect of a particular type



T: KonTest; *N*: 53; 90 flows; 1 flow at a time; 1 packet per second

Fig. 16. The end-to-end routing latencies on KonTest depending on the number of extra transmissions (left) and for packets that did not require extra transmissions (right).

of link changes—the aforementioned link burstiness—on the end-to-end packet delivery rates, our framework adopts a strategy proposed by Srinivasan et al. [2008]. If a forwarded packet is unacknowledged by the next-hop neighbor, the forwarding node defers retransmission for some period—one second in the experiments depicted in Figure 15. Considering that a routing path on KonTest typically does not exceed four hops, a one-second delay significantly impairs the end-to-end latency of the packet. This phenomenon is illustrated in more detail in the left plot of Figure 16, which presents the correlation between the number of extra transmissions on a routing path and the average end-to-end latency of the path. Essentially, the number of extra transmissions (and of the resulting one-second delays) determines the end-to-end latency of a packet: one extra transmission leads to end-to-end latencies of $\sim$1 s, two extra transmissions lead to latencies of $\sim$2 s, and so on.

However, thanks to the link quality estimation mechanisms, most packets (96.79% in the experiments—see the left plot of Figure 16) do not require any extra transmissions. For such packets, the end-to-end latency is directly proportional to the number of routing hops (see the right plot of Figure 16). It can even be estimated as roughly 25 ms per hop. In other words, most packets experience a relatively low end-to-end latency. Nevertheless, due to transient changes in connectivity and the mechanisms for handling such changes, some packets do experience higher latencies. Such latencies not only directly affect the application layer, but also the end-to-end reliability mechanisms of the transport layer. Therefore, sensornet application developers should be aware of such issues.

Finally, changes in connectivity may more permanently break paths between some node pairs. This may occasionally disrupt the cluster hierarchy and hence may result in changes of node labels. Since a node's label is the node's routing address, a change in the label disrupts the application on top of the routing infrastructure. Therefore, application developers employing hierarchical routing should anticipate such changes and should provide some recovery means. For example, in applications in which a sensor network involves a few special, more powerful, interconnected nodes (e.g., border routers), a simple support for dynamic addresses could designate the special nodes to maintain the node-label mapping. The special nodes would also act as the top-level cluster heads so that they could always be reached by all sensor nodes. When a source node wanted to query the label of a destination or received a "Destination Unreachable" message, it would contact the nearest special node to obtain the destination's current label. In contrast, in a network without special nodes, the node label resolution could be implemented with a distributed hash table on top of the cluster hierarchy, an approach that has been evaluated not only for hierarchical routing [Chen and Morris 2002; Du

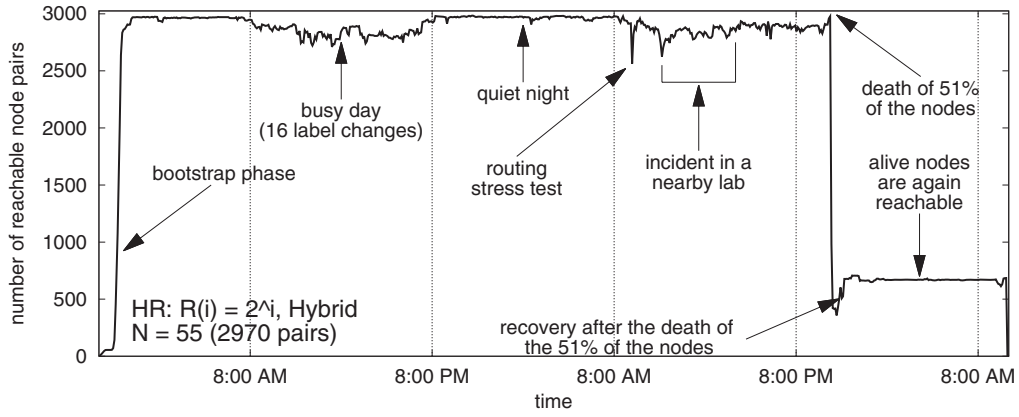


Fig. 17. The pairwise node reachability in a run. (Node A can reach node B iff A can successfully route to B.)

et al. 2004; Kumar et al. 2000], but also for compact routing and beacon vector routing [Mao et al. 2007].

A sample two-and-a-half-day run of the *Hybrid* variant of our framework, depicted in Figure 17, demonstrates how all the aforementioned phenomena impact the performance of hierarchical routing. The pairwise routing reachability between nodes is lower and more variable during working hours than during nights. During the first day (from 8:00 AM to 8:00 PM), six nodes changed their labels as the result of connectivity changes, amounting to 16 label changes in total during that day. In contrast, there were no label changes during the subsequent night. Those connectivity changes during working hours resulted most likely from the aforementioned noise and mobility in the testbed surroundings. Emergency events, such as the one we experienced, can also affect the routing infrastructure behavior. Although it is rather difficult to verify this, a crowd of people with their laptops storming through a corridor with the testbed office rooms may disrupt connectivity. In effect, many rounds after the building is deserted are required for the infrastructure to fully recover.

All in all, however, hierarchical routing as provided by our framework is relatively robust. During the worst disruptions in Figure 17, more than 84% of all $N\times(N-1)$ routing paths were valid. In addition, the network was resilient to massive failures and typically required few rounds to recover. Moreover, the end-to-end routing success rates we obtained are similar to those reported by other sensornet point-to-point routing protocols. Consequently, taking everything into account, hierarchical routing does have the potential to work in the real world.

8. CONCLUDING DISCUSSION

Although hierarchical routing seems a promising point-to-point routing technique for low-power wireless networks, little work has been done to evaluate this technique in actual sensornets. This article bridges the gap between theory and practice by experimentally identifying some of the major advantages, limitations, and open problems of hierarchical routing in sensornets, which we briefly summarize below.

8.1. Advantages of Hierarchical Routing

To begin with, our hierarchical routing framework developed in TinyOS proves that hierarchical routing need not necessarily be too complex to enable practical implementations on resource-constrained sensor nodes. Conversely, customizable and relatively

simple mechanisms can be used to implement hierarchical routing for tiny wireless embedded devices.

Moreover, the experimental evaluations of the implementations show that in practice, hierarchical routing can offer excellent performance compared to other point-to-point routing techniques. In particular, compared to its main competitor, compact routing, hierarchical routing offers scalability of at least an order of magnitude better in terms of routing state, and at the same time, its routing stretch is only 10–15% worse than that of compact routing. Overall, the results demonstrate that hierarchical routing has the potential to be considered for real-world applications of low-power wireless embedded networks.

Finally, hierarchical routing can be tuned to individual applications. For example, by changing the properties of the cluster hierarchy, one can explore the trade-off between routing state and routing stretch within hierarchical routing itself. Likewise, by varying the mechanisms used for propagating hierarchy information in the network, one can obtain a protocol that emphasizes a particular aspect of robustness, such as the latency of reacting to changes in the network or the volume of traffic used to bootstrap and maintain the cluster hierarchy. Such adaptability of hierarchical routing is another argument in favor of adopting this routing technique.

8.2. Limitations of Hierarchical Routing

However, there are also a few limitations and other issues related to hierarchical routing which we believe sensornet systems developers should be aware of.

Let us start with the performance results reported for high-level simulations of hierarchical routing. As we demonstrate in this article, they can diverge significantly from the results with more realistic communication models or ones obtained with real hardware. Such a divergence can have a profound impact on a few systems aspects of sensornets based on hierarchical routing. Examples of such aspects include provisioning node memory for the routing protocol or ensuring a certain quality of service with respect to the routing latency. For this reason, high-level simulation results should be considered with caution.

Another issue is related to the addressing mechanisms of hierarchical routing. Apart from the fact that the hierarchical labeling may produce long node addresses, which by itself may constitute a challenging engineering problem, the addresses are dynamic. There are two major problems associated with such dynamic addresses. First, in general, an application running on a node is unable to know a priori the address of a destination node. Second, the address of the destination node may change while the application is active. These problems are inherent not only to hierarchical routing, but also to other techniques, such as compact routing and graph embedding. Moreover, it is often argued that they are outside the scope of the core routing layer: they can be dealt with in higher layers, using solutions such as the Internet Domain Name System or the mobility extensions for IPv6, or can be handled in an application-specific manner, as discussed in Section 7.2. However, if dynamic addresses are indeed a significant problem for some application, the application has to employ another routing technique, such as shortest-path routing or centralized routing, instead of hierarchical (or compact) routing.

Variable addresses are just one of the consequences of general network dynamics, the most severe of which result from node mobility. To avoid problems stemming from continuous rapid changes of the network topology, most routing protocols (including the ones presented in this article) assume immobile nodes, which allows the self-organization mechanisms to reach a stable state. Although previous high-level simulations suggest that hierarchical routing can handle limited mobility [Chen and Morris 2002; Kumar et al. 2000], those results should also be analyzed with caution. In

particular, the radio range and mobility rates assumed in the simulations incur network dynamics that are much lower than those we have observed in experiments with existing hardware and just walking-speed mobility. In those experiments, the mobility-unaware protocols from this article simply failed, because not only their self-management mechanisms but also their lower layers could not keep up with the changes in the network topology. Indeed, mobility changes a lot [Dutta and Culler 2009] and thus constitutes an important avenue for future research, as it is not obvious even that point-to-point routing is the best communication primitive for mobile sensornets.

It is not difficult to extend this list with other issues, such as security, end-to-end reliability, or efficient network capacity utilization, that need to be addressed before hierarchical routing can truly be used in real-world applications. Somewhat encouraging may be the fact that many of these issues are not unique to hierarchical routing, but are also inherent in other routing techniques.

8.3. Conclusions and Future Work

To sum up, this article demonstrates that hierarchical routing is indeed a promising point-to-point routing technique for large sensornets. The performance it offers in terms of routing state, routing stretch, and robustness is appealing compared to other state-of-the-art routing techniques. Moreover, this performance can further be tuned to individual applications.

At the same time, the article makes it clear that like the other routing techniques, hierarchical routing also has some issues and limitations, many of which are not necessarily trivial engineering problems; conversely, they may require future in-depth research.

We argue that such future research on point-to-point routing in sensornets should shift from a network-only to a systems perspective and should place more emphasis on applications, because, although existing sensornet point-to-point routing solutions (including the ones presented here) are heralded as attractive, to the best of our knowledge, they have not been evaluated in complete real-world systems. Consequently, it is not clear which of the trade-offs they offer are the most important and, likewise, which of the issues they introduce are the most problematic for particular application classes.

We believe that the existing body of work, in particular, this article, will facilitate implementing prototype large-scale sensornet applications employing point-to-point routing. We are ourselves looking into prospects of such applications based on the protocols studied in this article. We anticipate that the results of such field experiments could feed the networking community with novel exciting routing problems.

ACKNOWLEDGMENTS

The authors would like to thank Tahir Azim for helping with the experiments on the MoteLab testbed and Albana Gaba and Arno Bakker for providing the computing power to conduct the long-running TOSSIM simulations. The authors are also very grateful to the anonymous reviewers of the *ACM Transactions on Sensor Networks*, whose insightful comments significantly improved the quality of this article. Finally, the authors would also like to thank the shepherd of the early version of this article, Philip Levis, and the anonymous reviewers whose feedback has helped to stimulate the research presented in this article.

REFERENCES

AMIS, A. D., PRAKASH, R., VUONG, T. H. P., AND HUYNH, D. T. 2000. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*. 32–41.

BANDYOPADHYAY, S. AND COYLE, E. J. 2003. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*. 1713–1723.

BOSE, P., MORIN, P., STOJMENOVIĆ, I., AND URRUTIA, J. 1999. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM'99).* 48–55.

CHEN, B. AND MORRIS, R. 2002. L^+: Scalable landmark routing and address lookup for multi-hop wireless networks. Tech. Rep. MIT-LCS-TR-837, Massachusetts Institute of Technology, Cambridge, MA.

DU, S., KHAN, A., PALCHAUDHURI, S., POST, A., SAHA, A. K., DRUSCHEL, P., JOHNSON, D. B., AND RIEDI, R. 2004. Self-organizing hierarchical routing for scalable ad hoc networking. Tech. Rep. TR04-433, Rice University, Houston, TX.

DUTTA, P. AND CULLER, D. 2009. Mobility changes everything in low-power wireless sensornets. In *Proceedings of the 12th USENIX Workshop on Hot Topics in Operating Systems (HotOS XII).*

FONSECA, R., GNAWALI, O., JAMIESON, K., AND LEVIS, P. 2007. Four-bit wireless link estimation. In *Proceedings of the Sixth ACM Workshop on Hot Topics in Networks (HotNets-VI).*

FONSECA, R., RATNASAMY, S., ZHAO, J., EE, C. T., CULLER, D., SHENKER, S., AND STOICA, I. 2005. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI'05).* 329–342.

FREY, H. AND PIND, K. 2009. Dynamic source routing versus greedy routing in a testbed sensor network deployment. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN'09). Lecture Notes in Computer Science*, vol. 5432, Springer-Verlag, Berlin, 86–101.

FUNKE, S., GUIBAS, L. J., NGUYEN, A., AND WANG, Y. 2006. Distance-sensitive information brokerage in sensor networks. In *Proceedings of the 2nd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'06)*. Lecture Notes in Computer Science, vol. 4026, Springer-Verlag, Berlin, 234–251.

HAGOUEL, J. 1983. Issues in routing for large and dynamic networks. Ph.D. dissertation, Columbia University, New York, NY.

HUI, J. AND CULLER, D. 2008. IP is dead, long live IP for wireless sensor networks. In *Proceedings of the 6th ACM International Conference on Embedded Networked Sensor Systems (SenSys'08).* 15–28.

IWANICKI, K. 2010. Hierarchical routing in low-power wireless networks. Ph.D. dissertation, VU University, Amsterdam.

IWANICKI, K. AND AZIM, T. 2010. Experimentally studying the sensornet point-to-point routing techniques spectrum. In *Proceedings of the 7th IEEE International Conference on Networked Sensing Systems (INSS'10).* 6–13.

IWANICKI, K., GABA, A., AND VAN STEEN, M. 2008. KonTest: A wireless sensor network testbed at Vrije Universiteit Amsterdam. Tech. Rep. IR-CS-045, VU University, Amsterdam. August.

IWANICKI, K. AND VAN STEEN, M. 2009a. Multi-hop cluster hierarchy maintenance in wireless sensor networks: A case for gossip-based protocols. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN'09).* Lecture Notes in Computer Science, vol. 5432, Springer-Verlag, Berlin, 102–117.

IWANICKI, K. AND VAN STEEN, M. 2009b. On hierarchical routing in wireless sensor networks. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'09).* 133–144.

JOHNSON, D. B. AND MALTZ, D. A. 1996. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, T. Imielinski and H. F. Korth, Eds Vol. 353. Springer US, New York, NY, 153–181.

KARP, B. AND KUNG, H. T. 2000. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual ACM International Conference on Mobile Computing and Networking (MobiCom'00).* 243–254.

KIM, Y.-J., GOVINDAN, R., KARP, B., AND SHENKER, S. 2005. Geographic routing made practical. In *Proceedings of the Second USENIX Symposium on Networked Systems Design and Implementation (NSDI'05).* 217–230.

KLEINROCK, L. AND KAMOUN, F. 1977. Hierarchical routing for large networks. *Comput. Netw. 1,* 3, 155–174.

KRIOUKOV, D., CLAFFY, K. C., FALL, K., AND BRADY, A. 2007. On compact routing for the Internet. *ACM SIGCOMM Comput. Commun. Rev. 37,* 3, 41–52.

KUHN, F., WATTENHOFER, R., ZHANG, Y., AND ZOLLINGER, A. 2003. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of the 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC'03).* 63–72.

KUMAR, S., ALAETTINOGLU, C., AND ESTRIN, D. 2000. Scalable object-tracking through unattended techniques (SCOUT). In *Proceedings of the Eighth IEEE International Conference on Network Protocols (ICNP'00).* 253–262.

LEONG, B., LISKOV, B., AND MORRIS, R. 2006. Geographic routing without planarization. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI'06).* 339–352.

LEONG, B., LISKOV, B., AND MORRIS, R. 2007. Greedy virtual coordinates for geographic routing. In *Proceedings of the 15th IEEE International Conference on Network Protocols (ICNP'07).* 71–80.

MAO, Y., WANG, F., QIU, L., LAM, S. S., AND SMITH, J. M. 2007. S4: small state and small stretch routing protocol for large wireless sensor networks. In *Proceedings of the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI'07)*. 101–114.

MOTELAB. 2005. Harvard sensor network testbed. http://motelab.eecs.harvard.edu/.

NEWSOME, J. AND SONG, D. 2003. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys'03)*. 76–88.

PERKINS, C. E. AND ROYER, E. M. 1999. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*. 90–100.

POLASTRE, J., HILL, J., AND CULLER, D. 2004. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys'04)*. 95–107.

RAO, A., RATNASAMY, S., PAPADIMITRIOU, C., SHENKER, S., AND STOICA, I. 2003. Geographic routing without location information. In *Proceedings of the 9th Annual ACM International Conference on Mobile Computing and Networking (MobiCom'03)*. 96–108.

SARKAR, R., YIN, X., GAO, J., LUO, F., AND GU, X. D. 2009. Greedy routing with guaranteed delivery using Ricci flows. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'09)*. 121–132.

SRINIVASAN, K., DUTTA, P., TAVAKOLI, A., AND LEVIS, P. 2006. Some implications of low-power wireless to IP routing. In *Proceedings of the 5th ACM Workshop on Hot Topics in Networks (HotNetsV)*. 31–36.

SRINIVASAN, K., KAZANDJIEVA, M. A., AGARWAL, S., AND LEVIS, P. 2008. The β-factor: Measuring wireless link burstiness. In *Proceedings of the 6th ACM International Conference on Embedded Networked Sensor Systems (SenSys'08)*. 29–42.

STATHOPOULOS, T., GIROD, L., HEIDEMANN, J., AND ESTRIN, D. 2007. Centralized routing for resource-constrained wireless sensor networks. Tech. rep., University of California, Los Angeles, CA.

SUBRAMANIAN, L. AND KATZ, R. H. 2000. An architecture for building self-configurable systems. In *Proceedings of the 1st Annual ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc'00)*. 63–73.

THALER, D. AND RAVISHANKAR, C. V. 1998. Distributed top-down hierarchy construction. In *Proceedings of the 7th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'98)*. 693–701.

TSUCHIYA, P. F. 1988. The landmark hierarchy: A new hierarchy for routing in very large networks. *ACM SIGCOMM Comput. Commun. Rev. 18,* 4, 35–42.

WOO, A., TONG, T., AND CULLER, D. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys'03)*. ACM, Los Angles, CA, USA, 14–27.

Received July 2010; revised November 2010; accepted May 2011