On Hierarchical Routing in Wireless Sensor Networks

Konrad Iwanicki
Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
iwanicki@few.vu.nl

Maarten van Steen
Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
steen@few.vu.nl

ABSTRACT

Hierarchical routing is a promising approach for point-to-point routing with very small routing state. While there are many theoretical analyses and high-level simulations demonstrating its benefits, there has been little work to evaluate it in a realistic wireless sensor network setting. Based on numerous proposed hierarchical routing infrastructures, we develop a framework that captures the common characteristics of the infrastructures and identifies design points where the infrastructures differ. We then evaluate the implementation of the framework in TOSSIM and on a 60-node testbed. We demonstrate that from the practical perspective hierarchical routing is also an appealing routing approach for sensor networks. Despite only logarithmic routing state, it can offer low routing stretch: the average of ~ 1.25 and the 99-th percentile of 2. Moreover, a hierarchical routing infrastructure can be autonomously bootstrapped and maintained by the nodes. By exploring the design points within our framework, the hierarchy maintenance protocol can optimize different metrics, such as the latency of bootstrapping and repairing the hierarchy after failures or the traffic volume, depending on the application requirements. Finally, we also identify a number of practical issues which we believe the applications employing hierarchical routing should be aware of.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols — *routing protocols*

General Terms

Performance, Design, Measurement, Experimentation

Keywords

Point-to-Point Routing, Hierarchical Routing, Cluster Hierarchy, Wireless Sensor Networks, Low-Power Wireless Networks, Self-Organizing Protocols, Scalability

1. INTRODUCTION

A plethora of recently proposed wireless sensor network (WSN) applications as well as a new dedicated IETF working group [27] evidence that point-to-point routing is important functionality for future low-power wireless systems [7]. In such systems, a point-to-point routing infrastructure directly affects scalability, efficiency, and reliability.

Since many of the proposed systems involve large networks, it is essential to provide routing infrastructures that concurrently offer small routing state, small routing stretch, and robustness [5, 12, 18]. Small state is crucial for scalability and efficiency. With only a few kilobytes of memory in typical sensor node platforms, minimizing the routing state enables supporting large networks. Moreover, smaller routing state often implies lower maintenance traffic, as these two are usually correlated. Small routing stretch, that is, few extra routing hops compared to the minimal possible number, is in turn essential for efficiency and reliability. It reduces resource consumption and end-to-end latency, and can improve end-to-end message delivery. Finally, since WSNs experience topology and connectivity changes due to node failures and environmental impact, robustness entails handling such changes efficiently. More specifically, to minimize resource consumption and disruption of higher-level system components, the routing infrastructure must handle the changes in the network with minimal traffic and latency.

Considering these requirements, hierarchical routing (HR) [6, 26] has often been mentioned as one of the four appealing techniques (the other three being geographic routing [12], graph embedding [5], and compact routing [18]). In HR, nodes maintain a hierarchy of clusters that provides an addressing and routing scheme. The scheme necessitates only $O(logN)$ node state (where N denotes the node population size) and offers relatively low average routing stretch. Moreover, numerous protocols that promise robust maintenance of the hierarchical routing infrastructure have been proposed.

However, despite the merits of HR, surprisingly little has been done to evaluate this technique in realistic WSN settings. The proposed HR protocols have mostly been studied with high-level simulations in idealized environments: a unit-disk connectivity model, no message loss, etc. While such high-level simulations give insight into theoretical properties of a protocol, they do not provide enough information on the practical performance of the protocol. Since numerous examples evidence that in WSNs practice typically diverges from theory, real-world applications of HR demand thorough implementation-based evaluations of this routing technique. Yet, we are not aware of any evaluation of HR 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 between them. However, because different applications may have different requirements on the routing infrastructure, one should be able to make an informed choice of a particular HR infrastructure. This requires a systematic comparison of the proposed solutions.

We contribute by bridging the gap between theory and practice. We have analyzed a few tens of hierarchical point-to-point routing infrastructures presented in major networking conferences and journals to select ones suitable for implementation on resource-constrained sensor nodes. Rather than simply implementing the selected infrastructures, however, we have developed a framework that captures their common characteristics and at the same time identifies various design points that differentiate the infrastructures and allow for exploring the design space. By building TinyOS 2.0 implementations of the framework with different decisions at these design points, we have obtained a means to systematically evaluate and compare many of the proposed infrastructures, as well as some novel ideas, in the real world.

We have conducted such evaluations in TOSSIM, a low-level simulator with a realistic low-power wireless communication model, and on our 60-node WSN testbed. Our results verify that despite only logarithmic routing state, HR can indeed offer low routing stretch: the average of ~ 1.25 and the 99-th percentile of 2. In terms of absolute values, however, these results diverge considerably from high-level simulation results, which confirms the need for practical evaluations. We also show that the performance of different techniques for maintaining the hierarchical routing infrastructure can vary dramatically; approaches based on hierarchical scoped flooding offer the lowest latency of bootstrapping and recovering the hierarchy, while approaches employing local communication minimize the traffic volume. An application can thus select the most suitable hierarchy maintenance technique or, as we demonstrate, combine some of the techniques to optimize certain metrics. Furthermore, this is just one of the identified issues that we believe the applications employing HR should be aware of. The final contribution of our work is the fact that our framework fills in the last gap in the implementations of routing techniques for WSNs, and hence, it can be used to systematically compare all the techniques.

The rest of the paper is organized as follows. We start with a survey of routing protocols for WSNs in Sect. 2, followed by a basic HR algorithm in Sect. 3. We introduce our framework and its implementation in Sect. 4, while in Sect. 5, 6, and 7, we discuss the setup and the results of the conducted experiments. Finally, we draw conclusions in Sect. 8.

2. RELATED WORK

Although point-to-point routing is a well-studied problem, WSNs pose novel challenges, in particular, severely limited effective node bandwidth and memory. These challenges constrain the use of shortest-path routing protocols, such as DSR [10] and AODV [20], in large WSNs as the control traffic and state of such protocols do not scale well. Routing protocols for large WSNs thus aim at minimizing the node state and the traffic for infrastructure maintenance. This is achieved with the following techniques: geographic routing, graph embedding, compact routing, or hierarchical routing.

In the first technique [11, 14, 17], a node's routing address corresponds to the node's geographic coordinates, and the node's routing table consists of the coordinates of the node's neighbors (i.e., the nodes within the radio range of the present node). In this way, the routing infrastructure requires only $O(1)$ state per node. However, practical, efficient geographic routing is essentially still an open research question due to the cases in which greedy forwarding toward the destination fails. Because of real-world issues, such as geographic localization errors or physical obstacles preventing radio communication, special solutions are necessary to handle such cases [12]. These solutions involve protocols for planarizing the neighborhood graph using cross-link detection (CLDP) [12] or for building hull trees [16]. However, such approaches make the maintenance or routing costly and complex (e.g., many subtle corner cases and two-phase locking to ensure consistency) [12, 16]. In addition, providing nodes with their geographic coordinates requires special hardware or additional localization algorithms, which both consume resources. Finally, there is no practical way of porting geographic routing to three dimensions, and thus, volumetric indoor networks may require different protocols.

In the second technique, graph embedding, instead of geographic coordinates, virtual coordinates synthesized by the maintenance protocol are used as node routing addresses. NoGeo [22] synthesizes coordinates through an iterative relaxation that embeds nodes in a Cartesian space. GEM [19], in turn, embeds nodes in a polar coordinate space based on a tree spanning from a base station. Finally, BVR [5] selects a fraction of nodes as beacons and constructs multiple spanning trees rooted at the beacon nodes, so that the coordinates of a node are the hop distances of the node from the beacons. An important advantage of the graph embedding protocols is that they do not require any additional localization mechanisms and can be used in volumetric deployments. However, this comes at a cost of heavy control traffic, large node state, and excessive coordinate setup time [22], or intricate recovery after a node failure [19], or costly flooding fallback due to a lack of routing guarantees [5]. Because of these drawbacks, other routing techniques are being studied in parallel.

One such a technique is compact routing, which explores the trade-off between the routing state and stretch [13]. In particular, S4 [18], the first complete compact routing protocol for WSNs, ensures a maximal routing stretch of 3 while using $O(\sqrt{N})$ node state. S4 selects $\sqrt{N}$ nodes as beacons. In addition, each node forms a virtual local cluster consisting of nodes whose distances to the present node are within their distances to the closest beacons. A node maintains the shortest-path routes to all nodes within its cluster as well as the shortest-path routes to all the beacon nodes. Thus, to reach a destination node outside its cluster, a source node first routes toward the beacon closest to the destination node. While S4 does well at bounding the maximal routing stretch, it requires $O(\sqrt{N})$ node state and maintenance traffic, which may be significant for very large networks and for some constrained sensor node platforms.

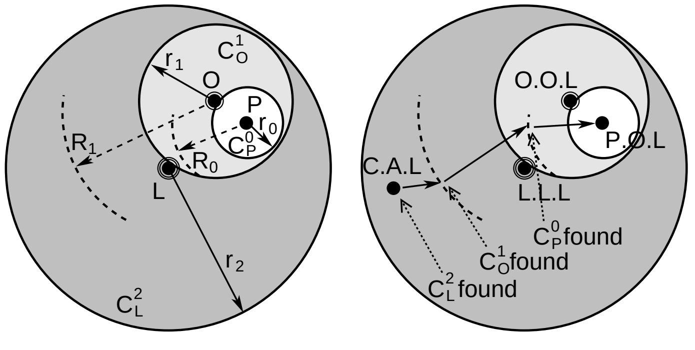
In contrast, the fourth technique, hierarchical routing (HR) [6, 26], which we describe in detail in the next section, minimizes the node state to $O(logN)$. Even though this may result in a high maximal stretch in some communication graphs, the average internode distance in the graphs of WSNs grows quickly with the node population (as $\sim N^{\nu}$, where $\nu > 0$), and thus the average stretch of HR can still be close to 1 in WSNs [13]. Therefore, considering that numerous protocols that promise robust and efficient maintenance of the hierarchical infrastructure exist (e.g., [1, 2, 3, 6, 9, 15, 24]), HR can be an attractive alternative for large deployments of highly constrained nodes [5, 15, 18]. However, as we argued in the previous section, to the best of our knowledge, HR infrastructures have been neither evaluated with actual embedded implementations run in realistic low-power communication settings nor systematically compared with each other. Therefore, since any practical application of HR in WSNs requires both practically confirmed performance and information on the trade-offs between the existing solutions, our work complements prior work on HR.

3. HR ALGORITHM

HR has many variants, all based on the same basic principles [6, 26]. We have analyzed a number of HR infrastructures proposed to date to choose an addressing and routing algorithm that, in our opinion, is well suited for implementation on resource-constrained sensor nodes. In the end, we have selected an algorithm commonly known as landmark routing [2, 15, 26], which we describe next. In addition to delivering all the aforementioned properties of hierarchical routing, the selected algorithm has many proposed protocols for bootstrapping and maintaining the routing infrastructure, which allows us to illustrate various trade-offs.

3.1 Basic Terms and Definitions

To support hierarchical routing, nodes are organized into a *multi-level hierarchy of clusters*, based on 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



(a) cluster hierarchy (b) routing algorithm

Figure 1: Hierarchical routing.

that cover the whole network. As a result, at every level of the hierarchy, each node belongs to exactly one cluster.

Each cluster has a *cluster head* that is the center of the cluster. A level-i cluster is advertised to nodes up to R_i hops away from its head (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 cluster head ($r_i \leq R_{i-1}$). Typically, $R_i = 2^i$ and $r_i = \lfloor R_i/2 \rfloor$. In this way, clusters at subsequent levels have exponentially growing diameters, and thus, the hierarchy can have $O(logN)$ levels. Since all nodes have unique identifiers, we use C_X^i to denote a level-i cluster with the head node X. A sample cluster hierarchy is depicted in Fig. 1a.

A node's *label* reflects the node's membership in the cluster hierarchy and is a concatenation of the cluster head identifiers of all the clusters the node belongs to, starting from level 0. In the example from Fig. 1a, the label of node P, a level-0 cluster head, is $P.O.L$, as P belongs to clusters C_P^0, C_O^1, and C_L^2. 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 $L.L.L$. The 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 i. An entry for a cluster head consists of the level and the identifier of the head, as well as the link-layer address of the next-hop neighbor on the shortest path to the cluster head and the length of this path. In this way, a node's routing table contains only $O(logN)$ a-few-byte-long entries [2, 15, 26].

3.2 Routing Algorithm

The routing is performed hierarchically: a packet is forwarded toward cluster heads corresponding to the elements of the destination label at subsequently decreasing levels (see Fig. 1b). More specifically, when a node receives a packet to forward, it searches its routing table for entries that correspond to the clusters represented by the elements of the destination label. The forwarding node will always find a routing entry for the top-level cluster of the destination node. However, as the packet is routed toward the destination, or even toward the top-level cluster head, 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 does not necessarily need to forward a packet even if it 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 Fig. 1b, for instance, node C (label $C.A.L$) routes a packet to P (label $P.O.L$). The packet is first routed toward P's top-level cluster head, L. During this process, however, when the packet reaches a node within R_1 hops from O, it is redirected toward O, as the node has a routing entry for C^1_O. Likewise, when the packet reaches a node within R_0 hops from P, it is routed toward P, as the node has a routing entry for C^0_P. In other words, the algorithm does not create highly undesired routing hot spots at cluster heads.

Due to maintaining only $O(logN)$ routing entries per node, HR typically does not use optimal routes. Often when nodes forward a packet, the packet moves toward higher-level cluster heads before being redirected toward the destination. We evaluate the routing stretch in our experiments.

4. IMPLEMENTING HR

The above HR algorithm is straightforward. However, to use it, the routing infrastructure must organize nodes into a hierarchy of clusters that determines the node labels and routing tables. Constructing a cluster hierarchy is already an NP-complete problem if the hierarchy is to be optimal, for instance, with respect to the routing table size [6]. Moreover, this is just the first step, as the hierarchy has to be maintained during the whole network lifetime. When the node population or connectivity changes, parts of the hierarchy may have to be repaired to account for the change. Such hierarchy maintenance can be even more problematic than its construction. Thus, the protocol for synthesizing and maintaining the node labels and routing tables is the key component in any HR infrastructure as well as in our framework.

To develop the framework, we have analyzed a number of HR infrastructures from major networking conferences and journals, focusing on the ones suitable for implementation on resource-constrained wireless devices. The framework encompasses the common elements of the selected infrastructures and defines various design points where the infrastructures differ. Since the plethora of the design points preclude detailed discussion in a 12-page paper, we give only an overview of some major ones. We then present a simplified framework and focus on a single design point that, from our experience, is determinant for hierarchy construction and maintenance, but has not been studied thoroughly to date.

4.1 Sample Design Points

The first major design point is the cluster scaling function $R_i = R(i)$, which determines cluster scaling properties at subsequent levels. To enable $O(logN)$ routing tables, $R(i)$ must depend exponentially on, i, the level; typically $R(i) = \alpha^i \times R(0)$ ($\alpha > 1$). By varying the scaling function, one can explore the state-stretch trade-off within HR. Here, we assume the most common scaling function: $R(i) = 2^i$.

One can also choose between a 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 [3, 9]. In a nonrecursive hierarchy, this may not hold [2, 15]. Maintaining a recursive hierarchy is more intricate than a nonrecursive one as the infrastructure 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. Here, we assume recursive hierarchies.

Node labels can be synthesized offline, prior to the deployment [6, 26] or at runtime, using a self-organizing algorithm [1, 2, 3, 9, 15, 24]. Low-power wireless connectivity is highly unpredictable [28], 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 paper, we use 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 merging lower-level clusters into higher-level clusters [2, 15, 9] . In a top-down algorithm, in turn, the labels are constructed from the top level by splitting higher-level clusters into lower-level clusters [25]. In this paper we concentrate on bottom-up algorithms as top-down approaches have problems adapting to varying topology parameters, such as nonuniform node densities [15].

Label maintenance algorithms can further be divided into deterministic [6] and probabilistic [1, 24], depending on their clustering heuristics. Since running multi-step deterministic algorithms on a large network of resource-constrained nodes is expensive, in this paper we use probabilistic algorithms.

Such algorithms employ a combination of local label update operations and update propagation mechanisms. The update propagation and cluster advertisement techniques constitute another design point, discussed in more detail further.

4.2 Simplified Framework

The above sample of the design points illustrates that the implementation of an HR infrastructure involves many intricate details and decisions. As a result, in this paper, we are unable to give a description of our whole framework or study the performance impact of all design decisions. Instead, 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 paper.

Principal operation: A protocol for maintaining the HR infrastructure 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 the node is the head of 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 techniques for advertising clusters and propagating label updates in Sect. 4.3. 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 Fig. 1, being initially a level-0 cluster head, node O spawned its own level-1 cluster, C_O^1, 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_P^0, to a higher-level cluster, C_O^1, 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 timely 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 extended 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 to ensure recursiveness, 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 above 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 [1], it is not required from the routing perspective. Neither when routing (cf. Sect. 3.2) nor when maintaining the routing infrastructure, higher-level cluster heads transmit more messages or perform more computation 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 this simplified HR infrastructure maintenance framework makes particular design choices, it has two properties. First, it captures the common hierarchy maintenance scheme of several proposed HR infrastructures (e.g., [1, 2, 3, 9, 15]). Second, it unifies the operation of these infrastructures to enable systematic comparison of different design decisions within this common scheme, as we discuss next.

4.3 Hierarchy Data Propagation Techniques

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. Consequently, they constitute a crucial design point in our framework, which, however, has not been thoroughly studied to date. Table 1 presents the major hierarchy information propagation techniques.

Table 1: Hierarchy information propagation techniques.

Technique	Acronym	Some protocol examples
periodic hierarchical beaconing	PHB	SCOUT [15], Safari [3], Subramanian & Katz [24], Bandyopadhyay & Coyle [1]
hierarchical distance-vector	HDV	Hagouel [6], Tsuchiya [26], L+ [2], PL-Gossip [9]
hybrid of the above two	Hybrid	proposed in this paper

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 a R_0-hop beacon every R_0 rounds, a level-1 head — a 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: 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 time 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, but 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 is explicit, by forwarding beacon messages issued per cluster, in *HDV*, it is implicit, by periodically exchanging and merging the local state 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 fast. 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 techniques. 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.

4.4 Implementation Remarks

As the implementation platform for our framework we have chosen TinyOS 2.0 and assumed a standard protocol stack. At the top, an application layer receives routing requests from a PC and periodically reports back the state of the HR infrastructure and the delivery status of the requests. We have not implemented node-id-to-label resolution, and instead, when routing, source nodes obtain destination labels with out-of-band means. This is because naming in WSNs is typically data centric, and thus, obtaining the destination address is often application specific. If necessary, the node-id-to-label mapping can be implemented as a distributed hash table on top of the cluster hierarchy [2, 3, 15].

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 HR layer. The HR 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. We use the standard link estimator based on the exponentially weighted moving average of packet reception rate [28]. Likewise, as the MAC layer, we use the standard TinyOS 2.0 CSMA/CA [21].

5. EXPERIMENTAL SETUP

We conducted our experiments in a low-level TinyOS simulator, TOSSIM, and on a 60-node testbed. The TOSSIM experiments aimed at evaluating the scalability of HR 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, the nodes were placed inside a square area and positioned in a grid, uniform, 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 1024, obtaining diameters of 15-18 hops in 1024-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 cover many 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 [28]. 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 HR assumed this model.

The aim of the testbed experiments, in turn, was to evaluate HR in a real WSN of a moderate size. The testbed consists of 60 TelosB nodes in six office rooms. The detailed information on the testbed can be found in a technical report [8]. In short, with the -15 *dBm* radio transmission power we used, the network diameter oscillated between 4 and 5 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 the testbed can serve as a representative example of a real-world WSN deployment of a moderate size.

Since we were conducting the testbed experiments for many weeks, the collected data illustrate the long-term behavior of a routing infrastructure 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 HR, in particular, and point-to-point routing, in general.

6. TOSSIM EXPERIMENTS

A routing infrastructure for WSNs should ensure small routing state, small routing stretch, and low-cost, low-latency maintenance. In the remainder of this section, we evaluate hierarchical routing with respect to these goals.

6.1 Routing State

As the metric for the routing state, we use the number of routing table entries as the entries dominate the memory consumed by an HR protocol. In our implementation, a routing entry at a node needs 8 bytes (for the fields listed in Sect. 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 4 bytes. Figure 2 compares the number of entries for different network configurations. Each data point corresponds to the average or the 99-th percentile over all nodes and ten protocol runs resulting in ten different hierarchies, each with the maximal level of 5, as this was enough for the employed cluster scaling function, $R(i)$, and the considered network diameters.

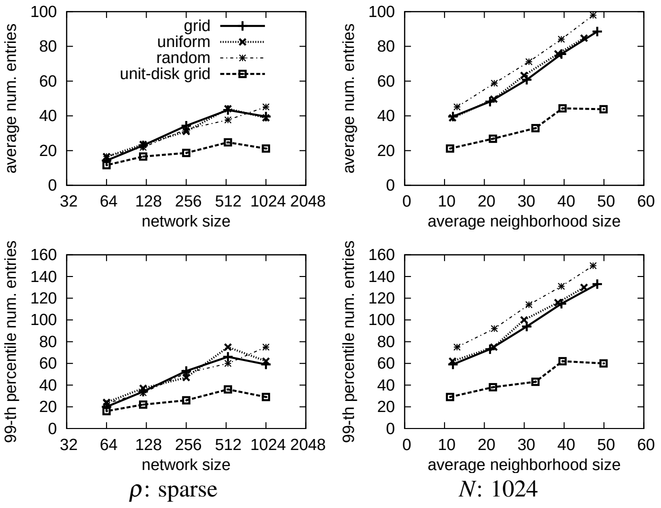


Figure 2: The average and 99-th percentile routing table size for different network sizes, densities, and topologies.

In accordance with the basic idea behind hierarchical routing, the routing state scales logarithmically with the network size N, (left plots), but there is a small constant factor, on average 4 with respect to $log_2 N$. The state also grows linearly with the network density (right plots). This is because, even though the employed cluster head promotion heuristics can adapt to increasing node density [1], they are still inherently imperfect, and thus, in denser networks more cluster heads get promoted, which increases the node routing state. We suspect that this is also the case for other routing protocols that employ probabilistic node election, such as S4 [18].

It is noteworthy that the 99-th-percentile routing state is lower than twice the average. Hence, when provisioning memory pools for routing entries, one can expect that a node on average utilizes at least 50% of its pool. This, combined with the low 99-th-percentile routing state size, is a strong argument for utilizing HR on memory-constrained nodes.

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 distance), the average node state under the unit-disk model can be more than 3 times smaller than under 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, in turn, can have further consequences for the hardware node platform. A platform with memory that, based on high-level simulations, was carefully minimized to support a given deployment, in practice may be unusable as the memory may turn out far insufficient for the routing entries. Therefore, when estimating memory for routing entries, prior results on HR should be taken with a grain of salt. This further substantiates the need for implementation-based evaluation of HR, as presented in this paper.

6.2 Routing Stretch

We measure the routing stretch with the standard metric [5, 12, 18]: 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 [28]. Figure 3 depicts the hop stretch in the hierarchies from Fig. 2. Despite using only $\sim logN$ routing entries, HR offers low 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. Fig. 2, right plots).

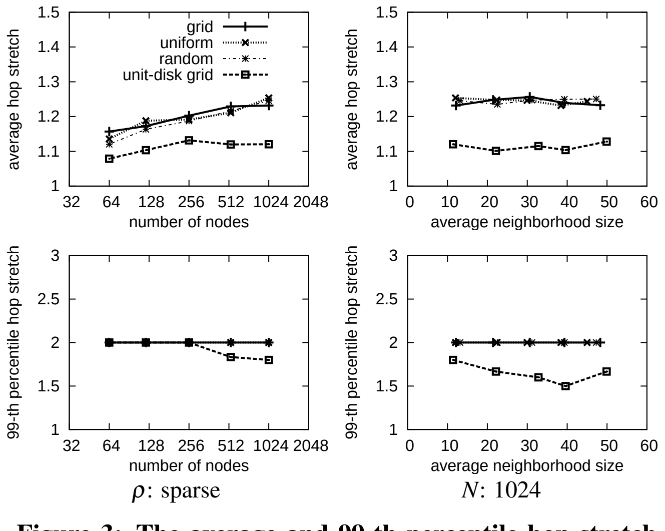


Figure 3: The average and 99-th percentile hop stretch for different network sizes, densities, and topologies.

Although the theoretical upper bound on the hop stretch between two nodes in HR is high [13], Fig. 3 shows that more than 99% of routing paths in our experiments do not exceed a hop stretch of 2 (bottom plots). Moreover, the worst path we have obtained throughout our year-long experiments, had a hop stretch of 5. This suggests that although in theory one can presumably construct a hierarchy with a high-hop-stretch path, the hierarchies synthesized by our framework offer low-hop-stretch paths with very high probability. Consequently, despite only $\sim logN$ routing state, HR may provide reasonable hop stretch bounds in practice.

Like previously, 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. Fig. 2). This is also a consequence of the aforementioned connectivity irregularities, and again evidences that in WSNs practice often diverges from theory.

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 WSNs, which in contrast exhibit high message loss. For example, a routing path may consist of few hops but involving poor, lossy links, which overall results 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 WSNs [5, 12, 18]: 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 hop stretch and the actual transmissions is low; the average transmission stretch is $\sim$1.02 and the 99-th percentile is not greater than 2. This is 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 [28], 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 $\sim logN$ state, HR may perform reasonably well in practice, which makes it particularly suitable for memory-constrained nodes.

6.3 Hierarchy Bootstrap

To study the costs of bootstrapping and maintaining the routing infrastructure, we make use of the fact that all considered protocols operate periodically, in rounds. We thus fix the duration of a round for all the protocols and study the maintenance costs 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 WSN applications the rounds are measured in the order of minutes (5 minutes in our experiments) while a message transmission takes in the order of milliseconds.

Figure 4 presents the number of rounds that different hierarchy information propagation techniques require to bootstrap the hierarchies from Fig. 2 and 3. In these experiments, we started all nodes simultaneously and let them construct the hierarchy. We consider the hierarchy as being bootstrapped when 99% of the nodes have their labels assigned and can successfully route to each other.

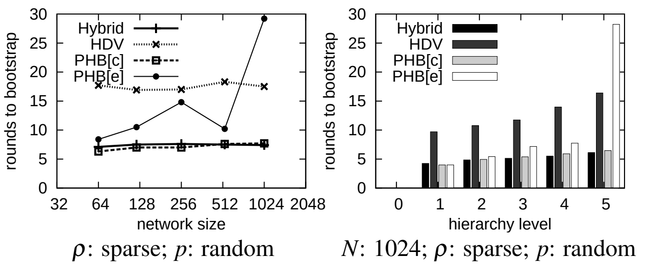


Figure 4: The bootstrap time for different network sizes and hierarchy information propagation techniques.

The hierarchy information propagation techniques 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 our novel hybrid approach (*Hybrid*). In these techniques, the number of bootstrap rounds is directly proportional to the maximal hierarchy level, 5. In every round, a single hierarchy level is constructed (the right plot) with the initial 1-2 rounds necessary for neighbor discovery and link estimation. This corresponds to the lower bound in our framework, and thus, these two techniques 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 techniques. 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 will not record a routing entry for the head for R_i rounds, which boosts the bootstrap time. In Fig. 4, 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 techniques. However, in *PHB[c]*, the unlucky node will likely receive a beacon in the next round, whereas in *Hybrid*, the unlucky node will recover through heartbeat messages.

Finally, the hierarchical distance vector (*HDV*) is the slowest. Since in this technique 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 Fig. 5 (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 protocols generate logarithmic traffic. The differences in constants associated with the logarithms, however, can be substantial (e.g., a factor of 10 between *PHB[c]* and *HDV[m]*).

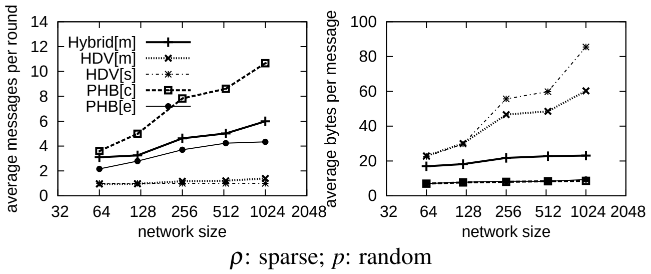


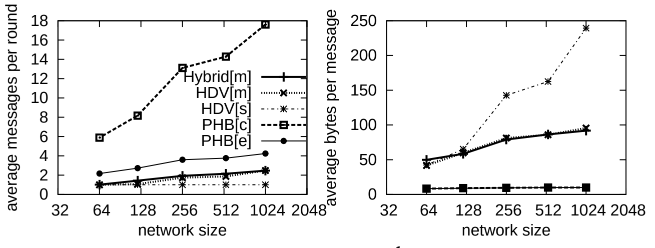
Figure 5: The per-node bootstrap traffic for different network sizes and information propagation techniques.

For efficiency, message payloads should be maximized: a protocol should preferably send fewer but longer messages. This is crucial in WSNs because a 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 Fig. 5 presents the efficiency of different hierarchy information propagation techniques. Beacon messages are small (10 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 technique, which combines beacons and heartbeats, lays in between. In the bootstrap phase, it resembles more the beaconing protocols as during this phase the hierarchy changes, hence beacons dominate over heartbeats.

6.4 Hierarchy Maintenance

After the hierarchy has been bootstrapped, the protocols continue to maintain it during system lifetime. Such maintenance generates traffic, which again depends on the hierarchy information propagation technique, as depicted in Fig. 6.

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*) — 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.

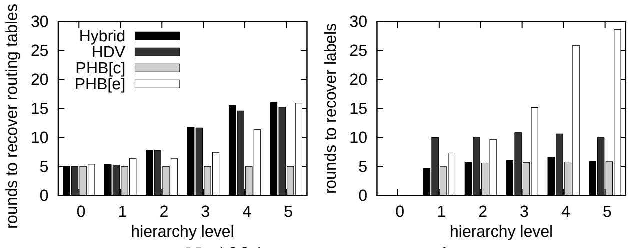


ρ: sparse; p: random

Figure 6: The per-node stable traffic for different network sizes and information propagation techniques.

Finally, since in the stable phase, unlike in the bootstrap phase, our *Hybrid* protocol generates only heartbeat messages, it performs similarly to the protocols using *HDV*.

The maintenance traffic is necessary for detecting and repairing failures in the hierarchy. To measure the failure recovery latency for every hierarchy information propagation technique, we performed the following micro-benchmarks. For each technique, after the hierarchy has been bootstrapped, we killed a single node and measured the time 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 is completely removed from the network), and all the alive nodes can successfully route to each other. Afterward, we reincarnated the dead node and let it fully rejoin the system (the rejoining latency is small and thus is omitted in the results). We then repeated the above steps for all other nodes in the network. Figure 7 presents the average results depending on the level of a failed node as cluster head.



N: 1024; ρ: sparse; p: random

Figure 7: The avg. recovery time of routing tables and labels depending on the failed node's level 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 technique, each routing entry is 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 4 consecutive rounds. Because *PHB[c]* also constructs the hierarchy fast (cf. Fig. 4), it recovers node labels most quickly (right plot), as label synthesis and recovery in the simplified framework use the same mechanisms (cf. Sect. 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 techniques 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. Fig. 4). 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 technique.

It is vital 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. Therefore, HR can be robust.

6.5 Comparison of the Techniques

To sum up, the three evaluated techniques for propagating hierarchy information differ significantly. 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).

Our 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 fast. 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 HR 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 HR infrastructures are based on this technique [1, 3, 15], which again supports our initial claims.

7. TESTBED EXPERIMENTS

We have conducted numerous testbed experiments that involved various micro-benchmarks of different design decisions within our framework as well as investigated the long-term performance of an HR infrastructure. Table 2 summarizes the routing state and hop-stretch values obtained in the experiments, while Table 3 illustrates differences between selected techniques of propagating hierarchy information.

Table 2: The routing table size and hop stretch.

Metric	Average	99th Percentile = Max.
routing table size	4.95-9.71	7-14
hop stretch	1.00-1.05	1.33-2.66
neighborhood size	19.51-23.65	26-34

Table 3: The convergence time and stable-state traffic for different hierarchy information propagation techniques.

Technique	Bootstrap Time	Stable-State Messages Per Node Per Round
PHB[e]	24	2.07413
HDV[m/s]	19	1.00000
Hybrid[m/s]	10	1.00036

Although, in general, the results from these experiments are consistent with the results from TOSSIM, there are some small differences. For example, the routing table sizes and hop stretch values are slightly smaller than the results we obtained with TOSSIM and the bootstrap times of the protocols are higher. We attribute these differences respectively to the small scale of the testbed experiments and to the fact that we did not start all the nodes simultaneously. Apart from such differences, however, the results from the testbed and TOSSIM match, which indicates that TOSSIM can simulate low-power wireless communication relatively well.

Yet, TOSSIM cannot accurately model the dynamic, random interactions of the network with the surrounding environment. Examples of such interactions include wireless noise generated by people's 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 an HR infrastructure. Wireless noise, for instance, makes some wireless links bursty [23]; such links display short periods of perfect or null packet reception. In turn, mobility, like repositioning office furniture by just half a meter, can change wireless links considerably and more permanently.

Such dynamic changes in connectivity impair the message delivery rates and the 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 is 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 [4]. We leave the evaluation of the framework with such an estimator as future work, as the estimator can also be applied to other routing protocols for WSNs.

More importantly, however, the changes in connectivity may result in the 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, applications employing HR must be designed to anticipate such changes and to recover from them. For example, in some applications, a simple support for recovery after address changes would involve statically designating a few (special) nodes as the keepers of the node label assignments. The keepers would also act as top-level cluster heads, so that they could always be reached by all nodes. When a source node receives "Destination Unreachable" message, it can contact one of the keepers to verify the destination label.

A sample two-and-a-half-day run of the *Hybrid* variant of our infrastructure, depicted in Fig. 8, demonstrates how the aforementioned phenomena impact the performance of an HR infrastructure. 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), 6 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 on 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.

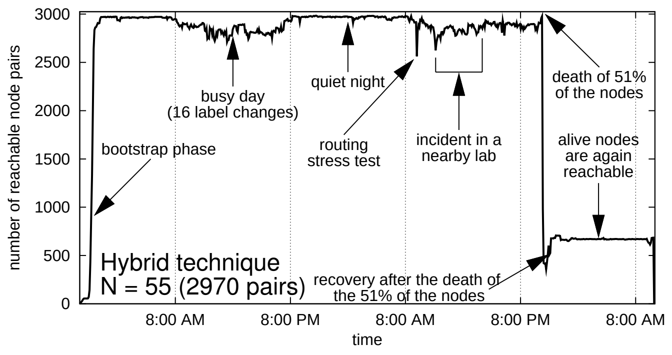


Figure 8: The pairwise node reachability in a run. (Node A can reach node B iff A can successfully route to B.)

All in all, however, HR infrastructures captured by our framework are robust. During the worst disruptions in Fig. 8, 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, we obtained similar routing success rates as reported by other WSN point-to-point routing protocols. Consequently, we hope that after some adaptation (including the aforementioned improvements), HR can be used in real-world WSNs.

8. DISCUSSION AND CONCLUSIONS

The results obtained with the embedded implementations of our framework indicate that hierarchical routing is indeed an appealing point-to-point routing paradigm for large low-power wireless networks. There are, however, some issues.

First, performance results reported for high-level simulations should be considered with caution, as we demonstrated that they can diverge significantly, in terms of both routing state and routing stretch, from the results with a more realistic communication model. Such divergence can have profound impact on a few systems aspects of WSNs using HR. Examples of such aspects include provisioning node memory for the routing infrastructure and ensuring certain quality of service with respect to the routing latency.

Second, the protocols for maintaining the routing infrastructure vary considerably in terms of performance. Our results evidence that there is no one-size-fits-all maintenance protocol. Instead, different techniques should be used when the availability and robustness are the main concern, and different when the energy efficiency is the primary objective. We believe that our results enable an informed decision.

Finally, WSN applications on top of an HR infrastructure should be partially aware of the limitations of the infrastructure. In particular, their design should anticipate disruptions in node connectivity as well as changes in node addresses, so that they can recover after such events (e.g., via buffering). This also applies to other protocols with dynamic addressing.

Apart from the above results, the presented HR framework is itself a major contribution as it allows for experimenting with various design decisions and for comparing HR against other point-to-point routing techniques. For these reasons, we have made parts of the framework sources publicly available at the following website.

http://www.few.vu.nl/~iwanicki/Ad_Hoc_Hierarchical_Routing

9. ACKNOWLEDGMENTS

K. Iwanicki was supported by a grant from Development Laboratories (DevLab), Eindhoven, The Netherlands. The authors would like to thank Albana Gaba and Arno Bakker for providing the computing power to conduct the simulations. Finally, the authors would also like to thank the shepherd of this paper, Philip Levis, and the anonymous IPSN'09 reviewers, whose feedback has helped to improve the paper.

10. REFERENCES

[1] Bandyopadhyay, S., and Coyle, E. J. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *Proc. IEEE INFOCOM 2003* (San Francisco, CA, USA, March 2003).

[2] Chen, B., and Morris, R. L^+: Scalable landmark routing and address lookup for multi-hop wireless networks. Tech. Rep. MIT-LCS-TR-837, MIT, March 2002.

[3] Du, S., Khan, A., PalChaudhuri, S., Post, A., Saha, A. K., Druschel, P., Johnson, D. B., and Riedi, R. Self-organizing hierarchical routing for scalable ad hoc networking. Tech. Rep. TR04-433, Rice Univ., March 2004.

[4] Fonseca, R., Gnawali, O., Jamieson, K., and Levis, P. Four-bit wireless link estimation. In *Proc. 6-th ACM Workshop on Hot Topics in Networks (HotNets-VI)* (Atlanta, GA, USA, November 2007).

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

[6] Hagouel, J. *Issues in Routing for Large and Dynamic Networks*. PhD thesis, Columbia Univ., May 1983.

[7] Hui, J. W., and Culler, D. E. Extending IP to low-power, wireless personal area networks. *IEEE Internet Computing 12*, 4 (July/August 2008), 37–45.

[8] Iwanicki, K., Gaba, A., and van Steen, M. KonTest: A wireless sensor network testbed at Vrije Universiteit Amsterdam. Tech. Rep. IR-CS-045, Vrije Univ. Amsterdam, August 2008. URL: http://www.few.vu.nl/~iwanicki/.

[9] Iwanicki, K., and van Steen, M. Multi-hop cluster hierarchy maintenance in wireless sensor networks: A case for gossip-based protocols. In *Proc. EWSN 2009* (Cork, Ireland, February 2009), Springer-Verlag LNCS vol. 5432.

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

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

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

[13] Krioukov, D., k c claffy, Fall, K., and Brady, A. On compact routing for the Internet. *ACM SIGCOMM Computer Communication Review 37*, 3 (July 2007), 41–52.

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

[15] Kumar, S., Alaettinoglu, C., and Estrin, D. SCalable Object-tracking through Unattended Techniques (SCOUT). In *Proc. IEEE ICNP 2000* (Osaka, Japan, November 2000).

[16] Leong, B., Liskov, B., and Morris, R. Geographic routing without planarization. In *Proc. USENIX NSDI 2006* (San Jose, CA, USA, May 2006).

[17] Leong, B., Mitra, S., and Liskov, B. Path vector face routing: Geographic routing with local face information. In *Proc. IEEE ICNP 2005* (Boston, MA, USA, November 2005).

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

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

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

[21] Polastre, J., Hill, J., and Culler, D. Versatile low power media access for wireless sensor networks. In *Proc. ACM SenSys 2004* (Baltimore, MD, USA, November 2004).

[22] Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., and Stoica, I. Geographic routing without location information. In *Proc. ACM MobiCom 2003* (San Diego, CA, USA, September 2003).

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

[24] Subramanian, L., and Katz, R. H. An architecture for building self-configurable systems. In *Proc. ACM MobiHoc 2000* (Boston, MA, USA, August 2000).

[25] Thaler, D., and Ravishankar, C. V. Distributed top-down hierarchy construction. In *Proc. IEEE INFOCOM 1998* (San Francisco, CA, USA, March-April 1998).

[26] Tsuchiya, P. F. The landmark hierarchy: A new hierarchy for routing in very large networks. *ACM SIGCOMM Comput. Commun. Review 18*, 4 (August 1988), 35–42.

[27] Vasseur, J. P., and Culler, D. Routing Over Low power and Lossy networks (ROLL). http://tools.ietf.org/wg/roll/, April 2008. Internet Engineering Task Force Working Group.

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