Probabilistic Addressing: Stable Addresses in Unstable Wireless Networks

Muhammad Hamad Alizai, Tobias Vaegs, Olaf Landsiedel*,
Stefan Götz, Jó Ágila Bitsch Link, Klaus Wehrle
Communication and Distributed Systems, RWTH Aachen University, Germany
{firstname.lastname}@rwth-aachen.de

ABSTRACT

Unreliable connectivity and rapidly changing link qualities make it challenging to establish stable addressing in wireless networks. This is especially difficult in communication scenarios where nodes determine their own addresses based on the underlying connectivity in the network.

In this paper, we present Probabilistic ADdressing (PAD), a virtual coordinate based addressing mechanism that efficiently deals with dynamic communication links in wireless networks. It assigns probabilistic addresses to nodes without needing to pessimistically estimate links over longer periods of time. The routing metric predicts the current location of a node in its address distribution. Our prototype implementation over real testbeds compares PAD with Beacon Vector Routing (BVR), a prominent routing approach in sensornets, and S4, a cluster based extension of BVR and state-of-the-art in point-to-point routing in sensornets. Our results from three widely used testbeds indicate that PAD achieves 3-7 times more stable addressing than BVR, and minimizes the number of transmissions in the network by 26% when compared with S4 under challenging networking conditions.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*

General Terms

Algorithm, Design, Experimentation, Performance

Keywords

Virtual Coordinates, Addressing, Tree Construction

*Work done while at RWTH Aachen University. New affiliation: School of Electrical Engineering, KTH, Sweden.

IPSN'11, April 12–14, 2011, Chicago, Illinois.

1. INTRODUCTION

Address updates are expensive in wireless networks where nodes have to determine their own addresses based on the underlying connectivity, a common situation in many wireless network deployments. In such networks, rapidly changing link conditions do not only affect packet delivery and routing topology, but also the topological location and addresses of nodes, requiring frequent location updates to be distributed in the network.

A plethora of solutions [6, 12, 14, 21, 23, 25] has been presented, both in ad hoc and sensornets, for situations where location information is not available at the nodes and geographic methods cannot be used for routing. The majority of these location independent addressing and routing schemes are based on tree construction primitives: Ranging from simple data collection [17] and dissemination [12] to complex virtual coordinates based point-to-point routing [14] in ad hoc and sensornets, tree-construction has established itself as common building block for location independent routing.

However, to ensure stable trees and addressing in the network, most tree-construction based addressing and routing schemes put excessive focus on tree maintenance and stability while adaptability gets compromised to a large extent. The underlying technique is to employ a long-term link estimator [14] and select parent(s) only among neighbors with consistently high quality links. Although it results in consistent addressing and stable routing trees across the network, this long term binding restricts the network in how well it can adapt to link dynamics [1, 26].

Nonetheless, tree based addressing and routing infrastructures suffer heavily from rapid topological changes due to varying link conditions in the network. Such situations often occur in a sparse network with a low density of nodes, where a node might have no reliable communication partner at all. In such situations we see frequent address changes and thus a significant overhead due to regular updates in the address database [3, 28]. Moreover, it would also result in inconsistent routing trees, introducing typical routing pathologies such as packet loss, loops, and stranded nodes.

In this paper we show how to retain the benefits of tree based addressing and routing schemes without maintaining explicit trees in the network. The basic concept is the same: Determine a node's location based on the vector of hop counts from a set of landmarks[1] in the network. However, the execution of this concept is substantially different. In contrast to the existing approaches, our solution neither

[1]Often referred to as beacon nodes. We use the term landmark to distinguish it from *beacon* packets.

relies on long term link estimation nor maintains any explicit parent-child relationships in the network. Instead, this approach, named PAD, assigns probabilistic addresses to nodes. The basic idea is that a node learns from its past locations and calculates the probability distribution over its recent locations. This probability distribution is then used as address of the node. Hence, a node's location is defined in terms of the probability that it exists in a certain location and remains independent from the packet loss at shorter time scales. All other nodes in the network predict the current location of a node in its distribution. As a result, PAD decouples addressing from routing, allowing to adapt routing paths to the very recent network conditions. The design of PAD is inspired by *atomic orbitals* [8] that describe the probability of finding the electrons of an atom in specific regions. Thus, the location of an electron is defined in terms of the probability that it exist at a particular location around the nucleus. An alternative view on the PAD approach is that it uses fuzzy instead of sharp coordinates for nodes.

This paper makes the following key contributions:

Address Stability: Compared to other addressing and routing schemes, PAD requires 3-7 times fewer address updates in a global location directory. At the same time, it maintains a small amount of state and requires considerably less effort and complexity in its mechanisms and implementation. We show that such stable addressing can be achieved even by considering only very recent link conditions instead of pessimistically overhearing and estimating links over a time period in the order of minutes (or hours).

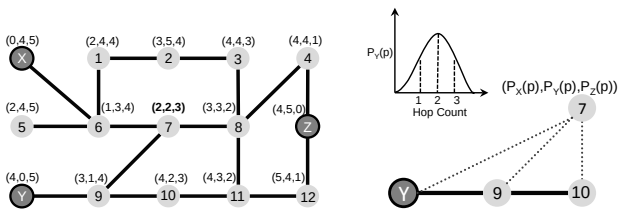
Address Monotony: Once an address update occurs, the difference between the old and new location of a node is 3-12 times smaller for PAD than for comparable approaches. This implies that the changes in PAD's addresses are gradual, which helps routing success. Our evaluation shows that this phenomenon allows PAD to maintain more up-to-date yet stable node locations in the network.

Responsiveness: By decoupling addressing from routing and link estimation, PAD can respond rapidly to changes in link quality which existing routing algorithms naturally avoid. As a result, each data packet can be forwarded on a different path depending upon the very recent network conditions. Our comparative analysis on three testbeds shows that even a simple routing strategy over PAD reduces the number of transmissions by 26% in testing network conditions when compared with S4 [21] – state-of-the-art cluster based point-to-point routing in sensornets.

The remainder of this paper is structured as follows. Section 2 presents the background, introduces the design space and identifies target environments. We describe the PAD design and algorithmic details in Section 3. Section 4 thoroughly evaluates PAD regarding stability and adaptability. We present a simple routing strategy over PAD in Section 5. Results from our routing evaluation are discussed in Section 6. Finally, we discuss prominent related works in Section 7 before concluding the paper in Section 8.

2. PRELIMINARIES

Our key contribution is to enable stable addressing by introducing probabilistic addresses for wireless networks. Before delving into the details, we revisit tree construction based routing schemes and put our contribution in a simple context. We also consider our target environments – data centric (sensornets) and ID centric (ad hoc and mesh) networks – and shed light on how these environments may benefit from the addressing mechanism presented here.



(a) Coordinate establishment in virtual coordinate based routing protocols. X, Y, and Y denote landmarks.

(b) Probabilistic addressing: Represent a node address as probability distribution over the past connectivity.

Figure 1: Addressing based on Virtual Coordinates

2.1 Tree-Based Routing Schemes

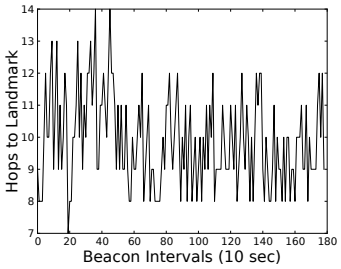
We can divide tree-based routing schemes into two categories: (1) *Address-free* collection (many-to-one) and dissemination (one-to-many) trees employed in situations where the identity of a node is not important. The goal here is to collect data from every node at a tree root or to deliver data to every node in the network and (2) *Address-based* point-to-point routing where a node's identity is essential for communication. Such schemes form multiple trees in the network and assign *virtual coordinates* (i.e. addresses) to nodes based on the vector of hop distances to the tree roots. Here we focus on the latter case, where addresses are important for networking the nodes.

Figure 1(a) shows an example of a virtual coordinate based address establishment in a network with three tree roots (landmarks), labeled X, Y, and Z. These landmark nodes advertise themselves by repeatedly sending beacons. Based on these beacons, each node S (recursively) determines the number of hops $h(S, L_i)$ to each landmark L_i. The result can be viewed as a set of routing trees with the landmarks as their roots and with, for example, the hop count as a routing metric. A node S's coordinates $\vec{c}(S)$ in the virtual coordinate system are the λ-dimensional vector $< h(S, L_1), \ldots, h(S, L_\lambda) >$ with λ as the total number of landmarks. In our example in Figure 1(a), node 7 has a three-dimensional address vector $< 2, 2, 3 >$ where each vector component represents the node's hop distance to the landmarks X, Y, and Z, respectively.

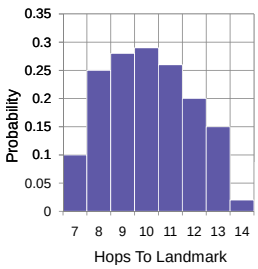
Typically, routing is performed greedily over these addresses. The idea is to let a node S choose a next hop T that minimizes the remaining distance $d(T, D)$ to the destination D (e.g. to select a neighbor as a next hop whose coordinates are most similar to those of the destination node). Widespread routing metrics include the *sum distance* (see Section 5.2) or the absolute component-wise difference [14]:

$$d(T, D) = \sum_{i=1}^{\lambda} |h(T, L_i) - h(D, L_i)| \quad (1)$$

However, real-world deployments are confronted with lossy links that may falsely influence the hop distance from landmarks. It means that traversing one hop can require more than one transmission. Therefore, the "best" next hop is the one that results in the least number of transmissions necessary to reach the destination. Routing protocols, such as BVR [14] or S4 [21], typically use link estimators to iden-



(a) Development of hop distance from a landmark over a period of 30 minutes.



(b) Distribution of hop distance from a landmark.

Figure 2: In a pathological case, a node's distance from a landmark can vary significantly over time. A link estimator is typically used to filter out such dynamics. In sparse networks with challenging link conditions, even link estimators would struggle to maintain a stable routing topology. Assigning static virtual coordinates in such dynamic situations often results in unstable addressing.

tify neighbors with stable links that minimize the expected number of transmissions (ETX) [9] for a successful delivery. Thus, only a selected subset of neighbors – offering an ETX below a certain threshold – are used in calculating the hop distance from the landmarks. Nonetheless, a node's current address vector still represents the hop distance over the path with minimum ETX.

2.2 Basic Idea

To establish a basic understanding of the design space we are working on in this paper, consider Figure 1(b), which represents a small segment of the network in Figure 1(a). The basic design philosophy of tree based routing protocols restricts nodes to select a single parent for each landmark and define their coordinates based on the hop counts achieved over these parents. As a result, a main challenge in tree construction based routing is that the changes at one node induce changes in all child nodes further down the tree. For example, node 7's virtual location with respect to landmark Y will heavily rely on the path $7 \rightarrow 9 \rightarrow Y$. Each time a node changes its hop distance from a landmark, all child nodes have to modify their hop distances to that landmark as well. As a result, any node failure or changes in the quality of the links (due to data loss) on this path will not only trigger a change in the routing topology but also in the virtual coordinates (location in the network) of node 7. To cope with this challenge, maintaining trees and virtual coordinates across the network which are particularly consistent is understandably the main objective of tree-based routing protocols. Therefore, they willingly concede performance penalties to achieve this objective.

In this paper, we oppose this philosophy and propose to break the stringent parent-child relationship. For example, let's suppose that node 7 can also reach landmark Y over the unreliable paths $7 \rightarrow Y$ and $7 \rightarrow 10 \rightarrow 9 \rightarrow Y$ (as shown in Figure 1(b)). We define a node's location on the basis of all possible paths that can be used to reach the landmarks regardless of the estimated quality of these paths, just like an orbital function describes possible quantum states of an electron around an atom [8]. We expect unstable coordinates based on these paths to exhibit a quantifiable, stable pattern. Nodes can keep track of the changes in their own coordinates and learn the associated patterns over time. Figure 2(a) depicts such a scenario from a real testbed by showing the development of a node's distance from a landmark over time. A similar argument can be made for cases where a node has no reliable neighbor over longer periods of time. In such a dynamic[2] network, assigning static addresses to nodes often results in inconsistent trees, because a node's distance from the landmarks changes rapidly.

The central idea is to locate and address a node using these patterns instead of its absolute, current coordinates. Our addressing mechanism, i.e. PAD, therefore addresses a node in the form of a probability distribution (see Figure 1(b) and 2(b)) instead of a static location. Other nodes can then use this probability distribution as the destination address for packets to this node. Overall, PAD decouples addressing from routing and exposes multiple locations and paths to a node. This gives routing protocols the flexibility to exploit interesting communication opportunities on short-term stable paths towards the destination.

2.3 Target Environments

In this section, we investigate the target environments that can benefit from the approach presented in this paper.

2.3.1 *Sensornets*

For a long time, routing in sensornets was limited to the simple collection and dissemination primitives that do not require to reach a specific node based on its identifier. However, a vast majority of current applications in sensornets – such as data centric storage [12], data query methods [15,20], pursuer-evader games [10], industrial automation, and cyber physical systems in particular – demand individual addressing support. Approaches like BCP [22] and BRE [2] provide mechanisms to exploit path diversity and link dynamics in sensornets but do not consider addressing. Moreover, the networking conditions reported in the literature [27, 29] are especially challenging in sensornets mainly due to two reasons: (1) harsh environmental conditions from a networking point of view and (2) a rapidly changing network topology resulting from frequent node failures.

2.3.2 *Ad Hoc and Mesh Networks*

Ad hoc and mesh networks are address centric, i.e. here the goal is to assign a unique identifier to each node and share resources such as an Internet connection among the participants. The presence of intermediate and bursty links has been recurrently reported in the literature [1, 26]. Such networks also present challenging conditions due to interference from other coexisting networks on the same frequency band and due to rapidly growing and shrinking numbers of participants. Opportunistic routing [5] provides an elegant solution to exploit path diversity in such networks. We share the same spirit, but differ significantly in detail. Moreover, opportunistic routing neither deals with addressing, as it operates on fixed geographic locations and IP addresses, nor focuses on challenging conditions in the network and their corresponding impact on addressing.

Our discussion in the remainder of this paper focuses on sensornets, but the ideas presented apply just as well to ad hoc and mesh networks.

[2]We only employ the dynamics that occur due to frequently changing link qualities and node failures.

3. PROBABILISTIC ADDRESSING EXPLAINED

Approaches such as BVR and S4 attempt to filter out the variability in a node's coordinates, which is caused by network dynamics, to obtain a stable address. In contrast, PAD incorporates this variability into a node's address by encoding a limited history of the node's coordinates. The idea is to learn from the dynamics exposed by a node's coordinates and express them in the form of probabilities. The routing algorithm can then determine a node's coordinates by predicting its current location in its probability distribution.

To arrive at a stable address, a PAD-node needs to iteratively (1) collect its coordinate history, (2) calculate and encode its address, and (3) disseminate its coordinate and addressing information to its neighbors via beacons. The following sections explain these steps in detail.

3.1 Coordinate History

PAD-nodes determine their network coordinates based on the beaconing mechanisms of established virtual coordinate systems [6, 14]. Thus, in a network with λ landmarks, the node S has the coordinates $\vec{c}(S) =< h(S, L_1), \ldots, h(S, L_\lambda) >$ as discussed in Section 2.1. Note that such coordinates reflect the current shortest paths between S and each of the landmarks without any further filtering, link estimation, or link quality information.

All nodes in the network determine their coordinates periodically, once per beacon interval. In PAD, each node S collects its σ most recent coordinates in its coordinate history, which is a table of size σ comprising coordinate vectors:

$$\mathcal{H}(S) = \begin{pmatrix} \vec{c_1}(S) \\ \vdots \\ \vec{c_\sigma}(S) \end{pmatrix} = \begin{pmatrix} < h_1(S, L_1) & \ldots & h_1(S, L_\lambda) > \\ \vdots & \ldots & \vdots \\ < h_\sigma(S, L_1) & \ldots & h_\sigma(S, L_\lambda) > \end{pmatrix}$$

In each beacon interval, S updates this history by adding its latest coordinates and evicting the oldest if necessary.

3.2 Address Calculation

After updating its coordinate history, node S recalculates its PAD address as follows. For each landmark L_i (i.e. each column in $\mathcal{H}(S)$), it determines which hop count values for L_i the history contains and how often they occur. In other words, it calculates the frequency distribution of unique hop counts for landmark L_i as the set of tuples:

$$\mathcal{F}_i(S) = \{(h_1(S, L_i), f_1), \ldots, (h_\delta(S, L_i), f_\delta)\}$$

where the tuple $(h_j(S, L_i), f_j)$ consists of the unique hop count value $h_j(S, L_i)$ and its absolute frequency (or number of occurrences) f_j in $\mathcal{H}(S)$.

For example in an increasingly unstable network, the number δ of tuples in $\mathcal{F}_i(S)$ would grow as the shortest paths from S to L_i increasingly vary in their length. At the same time, the absolute frequencies f_j of each of the hop count values would decrease as their sum, by construction, cannot exceed σ. In a very stable network, however, the history would report hop counts for only one or a few shortest paths between S and L_i, so $\mathcal{F}_i(S)$ would contain only one or few elements with rather large absolute frequencies.

After node S determined the frequency distributions $\mathcal{F}_i(S)$ for all landmarks, it constructs its routable address simply as the vector:

$$\vec{a}(S) =< \mathcal{F}_1(S), \ldots, \mathcal{F}_\lambda(S) >$$

With this information the probability distribution of a node's coordinates can be readily derived from its address. Thus, a PAD address contains a notion of path quality reflected by the number of different paths and their variance in length. Our results in Section 4 indicate that, given a suitable history size σ, the set of frequency distributions in an address stabilizes in the long run and is then largely unaffected by short-term link conditions. Furthermore, we observed that – even under challenging link conditions – the frequency distributions contain only a small number of unique hop counts, allowing for small address dissemination messages.

After calculating its new address $\vec{a}(S)$, S compares it to its previous address $\vec{a}'(S)$ by calculating a difference value d (e.g. via Pearson's χ^2-test). If d exceeds the threshold ϵ, S needs to update its address in the address database[3] of the network. Since an address update is an expensive operation, it is important to choose the threshold ϵ appropriately as discussed in the evaluation in Section 4.

Note that the node coordinates $\vec{c}(S)$ are deliberately based on the minimum hop distance metric $h(S, L_i)$ in PAD because: (1) it is a simple and established selection criterion, (2) it simplifies efficient routing and helps to avoid loops, and (3) it helps to keep the number of paths represented in PAD-addresses low.

3.3 Address Dissemination

Our approach is based on periodic beacon exchanges among neighbors. It is not bound to any specific beacon exchange rate or technique and, in principle, it should work with any technique presented in the literature, such as adaptive beaconing [17].

In PAD, each node S broadcasts a beacon packet once per beacon interval with the following information:

- **Node Coordinates:** The current vector $\vec{c}(S)$ of minimum hop distances $h(S, L_i)$ from S to each landmark L_i. To reemphasize, this does not use or contain any link quality information.
- **Node Address:** The current address $\vec{a}(S)$ of S, i.e. the frequency distribution of its coordinate history.
- **Neighbors**: A list of neighbors from which S received a beacon in the last beacon interval. This piece of information is used to identify neighbors with symmetric links for routing. This mechanism is similar to the regular exchange of reverse ETX messages for each neighboring link as used by current routing protocols.
- **Sender ID:** The unique ID[4] of S.
- **Sequence Number:** A sequence number for the beacon packet assigned to it by S.

The size of the beacon packets depends on the number λ of landmarks and the number of *symmetric neighbors*. In general, PAD allows to trade off transmission overhead against memory overhead in how address information is disseminated in beacons. The first option is to include a node's

[3]Location services for PAD are out of scope for this paper. There are well established location services for virtual coordinate based routing protocols, such as [7, 24].

[4]We distinguish the term *ID* from *address*. Each node in the network has a unique and immutable ID as opposed to its address, which is its relative location used for routing.

Testbed	Available Nodes	Average Node Degree	Tx Power Level Used
MoteLab	93	7.2	0 dBm
Indriya	125	18.5	−25 dBm
Twist	94	23.3	−25 dBm

Table 1: Basic characteristics of the three testbeds. All of them comprise IEEE 802.15.4-based TMote Sky nodes. Node degrees, i.e. average number of one hop neighbors, were derived for the respective transmission power levels.

address in its beacons which increases the beacon size. The second option is to only transmit a node's current coordinates instead of the aggregated PAD address. In this case, the neighbors that receive the beacon need to store a history of theses coordinates and compute the PAD address themselves, which increases the CPU and memory overhead.

4. EVALUATING PAD

Our evaluation of PAD focuses on two aspects: (1) We need to choose an appropriate history size σ and error level ϵ between PAD coordinate distributions (cf. Section 3.3). (2) We need to thoroughly compare PAD with existing virtual coordinate based addressing approaches to observe potential benefits and drawbacks of our approach. PAD is implemented for TinyOS 2.1.0, and has been tested in the TOSSIM simulator and on IEEE 802.15.4-based Tmote Sky platforms.

We first briefly discuss our experimental setup and the testbeds we used in our experiments.

4.1 Testbeds and Experimental Setup

Evaluation on real testbeds is mandatory to explore the efficacy of the concepts presented in this paper. We used three widely used IEEE 802.15.4 based testbed deployments for our evaluation, namely MoteLab [4], TWIST [18], and Indriya [11]. All three testbeds are indoor deployments – nodes are deployed on multiple floors of buildings – with coexisting IEEE 802.11 deployments. We used different transmission power levels to stress-test PAD under varying networking conditions and topological characteristics.

MoteLab is a 184 node deployment at the Harvard University on three different floors. Among the three testbeds, MoteLab is the sparsest deployment – only 93 nodes were active during our tests – with an average node degree of 7. MoteLab serves as a sanity check for PAD evaluation as it presents very challenging network conditions (see Figure 3). **Indriya** is a 127 node deployment at the National University of Singapore on three different floors. The network topology of Indriya is very similar to MoteLab, however, the overall network connectivity in Indriya is better than in MoteLab. 125 nodes were at our disposal, and we could reduce the transmission power to −25 dBm to increase the network's diameter. **TWIST** is a 100 node deployment (94 available) at TU Berlin. TWIST is the densest deployment among the three, and path lengths are quite small: Most of the nodes can reach each other directly when transmitting at full transmission power. Therefore, to create a multihop network we reduce the transmission power to −25 dBm. The major characteristics of these testbeds are shown in Table 1. Figure 3 shows the CDFs of link qualities on all the three testbeds and clearly points to the challenging nature of MoteLab: Almost 60% of the links have PRRs below 0.8 compared to just

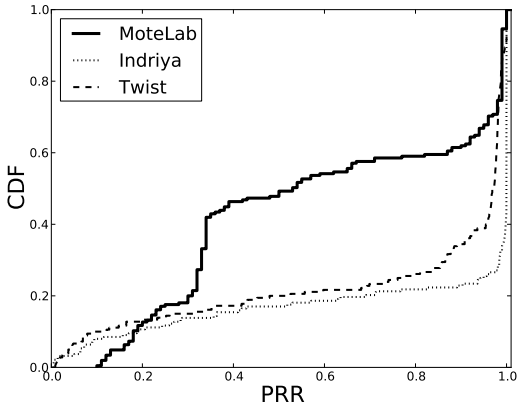


Figure 3: CDF of link qualities measured on the three testbeds. Almost 60% of the links in MoteLab have PRR's below 0.8 compared to just 20% of such links on Indriya and TWIST. We only include links on which at least 10 packets were received.

20% of such links on Indriya and TWIST. The outcome of Figure 3 is essential for understanding the results in Section 6.

4.2 Determining the System Parameters

Before evaluating the stability of addresses in PAD and comparing it to related approaches, we need to calibrate the core parameters of our system: the history size σ and the error probability ϵ. Although both σ and ϵ are user-desired accuracy thresholds, we derive their values here for completeness and for evaluation purposes.

4.2.1 History Size

Our first evaluation factor is to determine the appropriate sample size for the probabilistic addressing, i.e. the coordinate history size σ (see Section 3.1) which shall be used to calculate the PAD addresses. The goal is to strike a suitable tradeoff between the stability and adaptability of PAD. In order to find this we use Pearson's χ^2-Test. It is a test of goodness of fit, which derives how much two distributions differ from one another. Its purpose is to calculate a $p - value$ (error probability) that reflects how likely it is that the differences between two distributions are caused by chance.

To perform this analysis we ran PAD with six landmarks on the TWIST testbed at a transmission power level of −25 dBm. Each node generated a beacon every 10 seconds[5] for a total runtime of 24 hours. Our reference distribution of each node's coordinates for the χ^2-Test is derived from this data set. To determine the history size, we split the 24h data set into smaller segments of 6 to 1800 beacons (see Figure 4). We compare these partial distributions with the reference distribution to find the smallest history size for a node's coordinate distribution that can accurately represent the reference distribution derived from the whole experiment duration.

Figure 4 shows the average $p - value$ for different history sizes. It shows that there is a rapid decrease in the error

[5]The choice of the sending rate over a longer period of time is irrelevant here. We wanted to collect maximum data without saturating the network to derive a stable reference distribution of the coordinates.

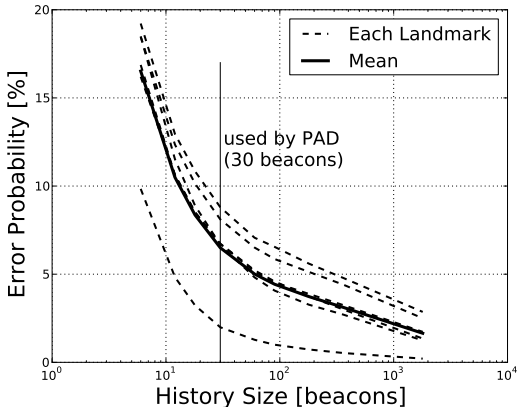


Figure 4: Pearson's χ^2-Test for deriving the history size σ: The graph shows a gradual decrease in the error probability for smaller history sizes. PAD uses the cutoff at $\sigma = 30$ beacons (300 seconds), as beyond that point, trying to improve the error probability even slightly introduces significant memory overhead and impedes the adaptability of addressing. (n.b. log scale on x-axis)

probability for smaller history sizes. For example, when increasing the history size from 6 to 30 beacons, the error probability decreases from 17% to 6.5%. However, later increasing the history size does not substantially impact the error probability anymore: increasing the history size from 30 beacons to 100 beacons only results in a 2% decrease of error while significantly dampening the adaptability of the coordinates and increasing the memory overhead due to the larger history size required to compute the PAD address. Here we can tradeoff a slight inaccuracy for a higher adaptability and smaller memory overhead.

The cutoff used in PAD is therefore at a history size σ of 30 beacons, i.e. 300 seconds in our case. For the remaining evaluation in this paper, we calculate the PAD addresses from a history comprising the last 30 beacons. Our results indicate that even with this history size PAD achieves at least three times more stable addressing than BVR. This result roots in the fact that BVR itself trades stability for adaptability by employing a highly pessimistic and cautious approach for changing the address of a node.

4.2.2 Error Threshold

The error threshold ϵ is the threshold for deciding whether the differences between a newly calculated PAD address and the previous one are significant and hence require an update in the global address database: After calculating its new address $\vec{a}(S)$, a node S compares it to its previous address $\vec{a}'(S)$. If the difference exceeds the threshold ϵ, S needs to update its address in the address database. Hence, ϵ allows the user to tradeoff address updates for routing inaccuracies. We evaluated the stability of PAD with different ϵ values and observed that for smaller values – ranging from 1% to 10% – ϵ does not impact the rate of address updates in the network. In our evaluation we use $\epsilon = 6.5\%$ as a representative value within that range.

Although both ϵ and σ thresholds are empirically derived from testbed results, self-calibration of these thresholds would be a preferred solution such that the network would optimize them according to the observed conditions. Nonetheless, any such self-calibration mechanism requires additional memory and computational overhead (e.g. to store, calculate and compare reference distributions of coordinates), which is not desirable in sensornets.

4.3 Comparison with BVR and S4

Deriving the error threshold ϵ and the history size σ completes all the pieces of our design. Now we thoroughly compare PAD with the addressing mechanism of BVR. We defer the discussion on routing over PAD to Section 5.

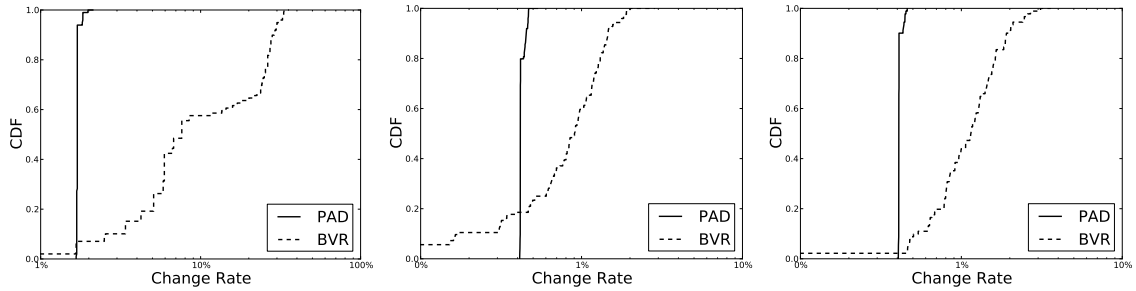
Although S4 is considered state-of-the-art in sensornets, our comparative evaluation in this section only considers PAD with *Beacon Vector Routing (BVR)* [14]. This is because S4 extends BVR with its cluster based routing approach to guarantee reachability at the cost of relatively higher state – maintaining both local-cluster and global states. Whereas, the establishment of *global* coordinates in an S4 network is exactly based on BVR: S4 even uses the code base of BVR. Thus our evaluation in this section accounts for both S4 and BVR with one exception: Routing in S4 is based on the closest landmark to the destination, and hence, updates in the address database are only needed when a node's closest landmark changes. BVR on the other hand requires an address update for every change in any of the coordinate components, because it greedily routes a packet based on the address vector of the k closest landmarks. The rate of coordinate change in S4 would still be the same as for BVR. Moreover, because S4 uses hop-count as its performance metric, later in Section 6 we show that PAD outperforms S4 when the comparison is performed on a testbed and is based on a prevalent metric, i.e. total *number of transmissions* in the network.

We use the latest releases of BVR and S4 from the TinyOS code repository[6]. To ensure that our results are not unjustifiably influenced by the state-of-the-art implementation of the TinyOS 2.1 communication stack, we had to update the APIs of S4 and BVR. However, these changes are only minimal and correspond to slight changes in platform independent APIs, such as *send* and *receive*. We did not alter any other parameter or algorithmic aspect of the protocols itself.

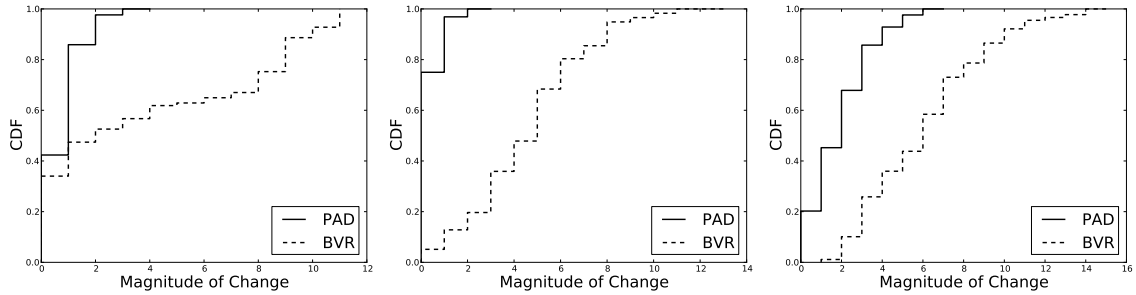
Our comparison with BVR is based on three factors. **Address Stability:** To compare the rate of changes in the addresses in PAD and BVR. It is defined as the share of beacon intervals in which the nodes changed their addresses. This is our key evaluation aspect to show the stability of addresses over time. **Address Monotony:** To measure the difference between hop distances from landmarks over time. This analysis looks at each component of the address vector to analyze the range of change in hop distances from each landmark. **Node Dynamics:** To observe the stability of PAD with regard to node dynamics, i.e. frequent node additions and failures in the network. This analysis will give us hints about how well PAD recovers from such dynamics.

Our experiments for this analysis have the following key characteristics: (1) Each experiment starts with an initial calibration phase of 10 minutes to allow BVR to stabilize its link estimates and virtual coordinate system. However, at any instant, PAD's address distributions are always derived from a history of the last 30 beacons. (2) The calibration

[6]These releases are compatible with TinyOS 2.0, but there are significant differences, e.g. the communication stack, device drivers and interfaces, between the current release TinyOS 2.1 and the first release of TinyOS 2.x.



(a) Change Rate (Share of Intervals with Change, %) in MoteLab (b) Change Rate (Share of Intervals with Change, %) in Indriya (c) Change Rate (Share of Intervals with Change, %) in Twist



(d) Magnitude of Change in MoteLab (e) Magnitude of Change in Indriya (f) Magnitude of Change in Twist

Figure 5: Results from the address stability comparison: The CDFs of our two evaluation factors from three testbeds indicate that PAD reduces the rate of change in addresses and decreases the magnitude of change in addresses on all testbeds.

phase is followed by the evaluation phase in which nodes regularly exchange beacon packets every 30 seconds[7]. (3) On each testbed we preconfigure six well spread nodes to act as landmarks. The landmark selection is a well studied research area [6,14,21] and is not our focus in this paper. (4) Finally, the link estimator in BVR employs passive overhearing of all transmissions in the network. This energy inefficient and computationally expensive mechanism is not employed in PAD. Therefore, as opposed to BVR and S4, there are no link-estimation headers appended with each packet in PAD.

4.3.1 Address Stability

Address stability is an important factor in a virtual coordinate based routing infrastructure, especially for applications that cannot maintain the state of each node in the network and require a lookup mechanism in some location database. First, because routing to outdated addresses leads to routing failures. Second, because rapid changes in the addresses create heavy update and lookup traffic overhead that can be detrimental for network performance, especially close to the nodes responsible for maintaining the address database. Figures 5(a), 5(b), and 5(c) show the cumulative distribution of the nodes' change rate in terms of percentage of the beacon intervals in which the nodes change their addresses. By employing this metric we can assume that the rate of sending beacons does not impact the change rate of addresses, since sending beacons at higher rates increases the chance for changes in addresses in a certain time period but also increases the total number of beacons by the same amount. The CDFs clearly indicates that PAD's addresses are significantly more stable than BVR's.

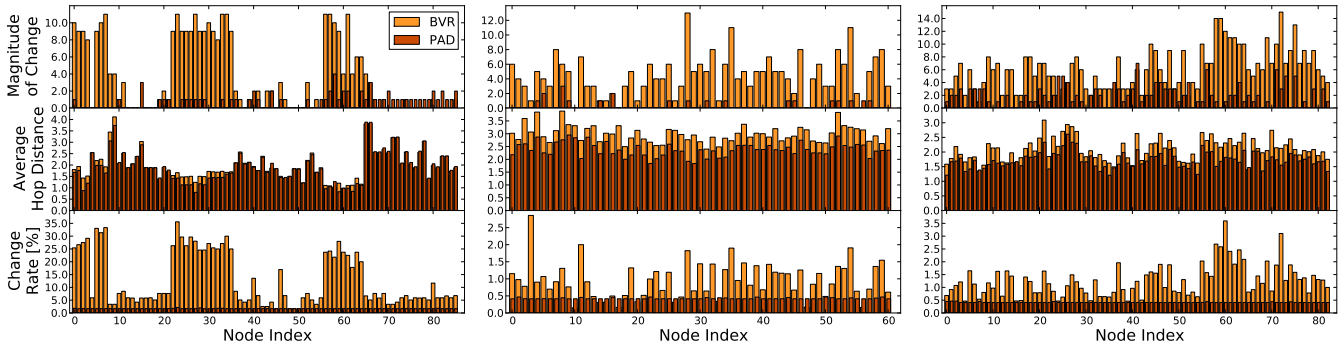
Figure 6 shows address change rate and average hop distance from landmarks for each node in all three testbeds. It can be seen that PAD addresses are quite stable even under challenging conditions (such as in MoteLab), where BVR's addresses have significantly higher change rates. From this we can conclude that instantaneous changes in link conditions may lead to coordinate changes in the addressing mechanisms that assign static virtual coordinates to nodes at any instant. However, the underlying patterns of these instantaneous changes show stable distributions even at small time scales.

4.3.2 Address Monotony: Magnitude of Change

The magnitude of change determines the difference between a node's virtual addresses. For example, if a node changed its hop distance to a landmark from 4 to 6, its magnitude of change (or range) would be 2. The magnitude of change in a node's address is calculated by summing up the magnitude of change in each address-vector. Figures 5(d), 5(e), and 5(f) show that the magnitude of change in addresses is significantly smaller in the case of PAD (see Figure 6 for magnitude of change in each node's addresses).

PAD shows a smaller magnitude of change because it sticks to the minimum hop distance path towards the landmarks. However, in BVR the address changes are influenced more by the ETX metric that favors long term stable links to achieve stable addressing in the network. As a result, the magnitude of change in addresses can be significantly higher. For example, BVR may select a longer but stable path after the previous path became invalid, whereas PAD's distribution will always favor the path with smallest hop count (see Section 3.2).

[7]During the calibration phase the beacon intervals are 10 seconds to increase our data set, as MoteLab and Indriya limit the time period of experimental runs.



(a) MoteLab (b) Indriya (c) Twist

Figure 6: Per-node analysis for address change rate, average hop distance from landmarks, and magnitude of change in addresses for all the three testbeds. The results show significant improvements even under challenging network conditions as experienced in MoteLab. The figures only show the data for the nodes that were available for all our experiments.

The smaller range of addresses implies that the changes in PAD addresses are gradual and a node's virtual location differs only minimally. Therefore, a packet routed towards a certain node is still routed into the vicinity and has a higher probability of reaching the target even if that has changed its original address. However, in BVR routing to outdated addresses may lead to routing failures due to nodes taking significantly different virtual locations.

4.3.3 Node Dynamics

After evaluating the stability of PAD under different network conditions, we now evaluate PAD from another perspective, i.e. by growing and shrinking the size of the network to see how well PAD integrates additional nodes and recovers from node failures. We use the TOSSIM [19] simulator to introduce such node dynamics in a simulated network. We use a 100-node grid-like topology in TOSSIM with 4 nodes configured as landmarks. Our first experiment starts with 50 nodes, and 10 evenly distributed new nodes are added to the topology after every ten minutes. Similarly, our second experiment starts with 100 nodes, and 10 evenly distributed nodes are deleted from the topology after every ten minutes. Figure 7 shows our results where each data point represents the addition (see Figure 7(a)) or deletion (see Figure 7(b)) of 10 nodes. We can clearly see that PAD achieves far less address changes in the network than BVR. This is because the addition of a new node in PAD only affects its address if it offers a smaller hop distance than the ones reflected in the current PAD distribution. However, link estimation based addressing in BVR takes time to incorporate new nodes and stabilize its link-metric and addressing across the network. Hence, these results prove the flexibility of PAD for networks with rapidly growing and shrinking numbers of participants.

4.3.4 Summary

Figure 8 summarizes our results regarding address stability. PAD achieves 3 and 7 times more stable addressing than BVR on Indriya and MoteLab, respectively. An alternative way to formulate these results would be that BVR achieves 89% stability and PAD achieves 98.5% stability on MoteLab: In every 1000 beacon intervals, a node changes its address 110 times in the case of BVR and 15 times in the case of PAD. Similarly, the range of addresses is reduced by 3 to 12 times on different testbeds. Finally, PAD reduces the hop distance from landmarks by 10–25%.

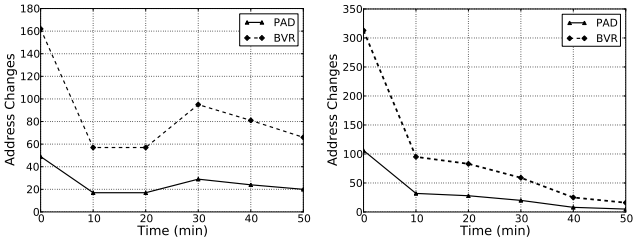
Concluding our comparative evaluation, we have seen that PAD makes significant strides in enhancing the efficiency of tree-construction based virtual addressing in wireless networks. It shows that stable addressing across the network can be achieved without compromising the adaptability of virtual coordinate based routing, which has been the trade of existing routing approaches for a long time.

5. ROUTING ON PAD

After discussing the design and experimental evaluation of PAD, we now present a simple routing strategy that can operate on PAD's addresses. While an advanced routing algorithm is not part of our main research contribution in this paper, we present a simple design here for completeness. So our goal is to merely to provide an address prediction mechanism and an adaptive routing strategy for the purpose of evaluation to see potential benefits and drawbacks of PAD. On the whole, we introduce some new flavors to make routing over PAD more adaptive and borrow some tactics from the existing approaches as well.

PAD's design in principle is independent from any specific routing strategy that operates on virtual coordinate based addressing mechanisms. Therefore, depending upon the application requirements, any routing strategy that leverages specific design objectives shall integrate well with PAD. Such strategies could include energy efficient [13] and adaptive mechanisms [2, 22] to maximize routing throughput or multipath [16] and retransmission [17] mechanisms to achieve reliable communications. The approach presented here is a combination of both, quickly adapting to the underlying link conditions while ensuring reliability by embedding retransmissions and link symmetry tests into the routing decisions.

There are two main elements of our routing approach: (1) We need a mechanism to precisely predict the location of a node in its coordinate distribution, and (2) we need to define a distance function to select the best next hop for forwarding the packet towards a certain destination. While there are well defined distance functions, such as the ones used by BVR [14] and LCR [6], predicting a node's location in its address distribution is a task we need to deal with. Our choices for both these elements are influenced by our primary design objective, i.e. simplicity. In Section 6 we show



(a) Node Addition (b) Node Deletion

Figure 7: Node Dynamics: With PAD node dynamics lead to significantly fewer address changes. Each data point represents the addition or deletion of 10 nodes. In total, PAD results in 154 and 201 address changes compared to BVR's 508 and 593 changes due to node addition and deletion, respectively.

that even this simple routing strategy over PAD's addresses can reduce the number of transmissions necessary for data packets to reach their destination.

5.1 Address Prediction

In a first step we convert PAD's distributions into meaningful addresses that can be used to derive routing decisions. In a first approach, we calculate the *mean* of each coordinate distribution in a node's address. Then, source nodes can use the *mean* address (aggregated PAD address) to forward the packet towards the destination in the virtual coordinate space. In our prototype implementation a node thus advertises these *mean* coordinates in its beacons as address for routing purposes.

As an alternative prediction mechanism, we propose utilizing coordinate *variance* information. The idea behind coordinate variance is to describe a node's location only with respect to those landmarks with stable hop distances over a certain time period. High coordinate variance corresponding to a landmark signifies that a node's hop distance from that landmark varies significantly. Hence, its location with respect to that landmark can be predicted less accurately.

5.2 Distance Function

To route packets, we need a distance function that, at each hop, selects the best next hop for the packet to reach its destination. We use a similar mechanism as BVR [14] for routing except that a data packet now carries the *mean* coordinates (derived from PAD addresses) of the destination instead of BVR's coordinates. At each hop, the destination's *mean* coordinates are compared with the coordinates of all one-hop neighbors. The neighbor, whose coordinates are most similar to the destination, is selected as the next hop for the packet. This process continues until the destination is reached or none of the one-hop neighbors further reduces the remaining distance to the destination, i.e. the current node is closest to the destination in terms of its virtual coordinates. In that case, we use the fallback mode (cf. 6.1 or [14]).

In order to compare coordinates for selecting best next hop, we propose the combination of the *sum distance* and the *sum of the differences*. The *sum distance* metric signifies the distance of the shortest path (number of hops) from a node S to a destination D via the landmark L. So the best next hop T is the one that minimizes $\bar{d}_k^S(T, D)$, the sum distance between T and D, averaged over a set $\mathcal{C}_k(D)$ of the $k = |\mathcal{C}_k(D)|$ landmarks closest to D [14]:

$$\bar{d}_k^S(T, D) = \frac{1}{k} \sum_{L \in \mathcal{C}_k(D)} (\bar{h}(T, L) + \bar{h}(D, L)) \quad (2)$$

with the mean distance of neighbor T to landmark L:

$$\bar{h}(T, L) = \frac{1}{\sigma} \sum_{j=1}^{\sigma} h_j(T, L) \quad (3)$$

where $h_j(T, L_i)$ is the j-th entry in the history of length σ of hop distances from node T for landmark L. Our implementation includes all landmarks of the network, i.e. $k = \lambda$. We propose a combination of $\bar{d}_k^S(T, D)$ and the *absolute component-wise difference* (cf. Section 2.1):

$$\bar{d}_k^B(T, D) = \frac{1}{k} \sum_{L \in \mathcal{C}_k(D)} |\bar{h}(T, L) - \bar{h}(D, L)| \quad (4)$$

The idea is to choose smaller of both distances for routing. Similarly, statistical measures, such as Kullback–Leibler divergence, Hellinger distance, or Total variation distance are possible choices to select a next hop based on its address distribution. However, utilizing these distance measurements for routing over PAD is a research challenge in itself, and a detailed exploration of the design space is beyond the scope of this paper.

To compare our approach with BVR, in our prototype we only use $\bar{d}_k^B(T, D)$ over the mean coordinates to analyze the true impact of PAD on routing without changing the decision process. Moreover, in our implementation of PAD, a neighboring node is only considered a possible next hop if it satisfies the following two conditions. (1) The link with that neighbor has an age of 3, i.e. the last three beacon packets were successfully received over that link. This approach is derived from Alizai et. al [2]. (2) The link is symmetric. Symmetric links are determined by including a list of all current neighbors of a node in its beacon packets (see Section 3.3).

6. ROUTING RESULTS

We compare our simple routing strategy over PAD with S4 and BVR. We use *number of transmissions* as our metric for comparison because it is the prevalent routing metric in energy constrained sensornets [2, 17, 22, 26].

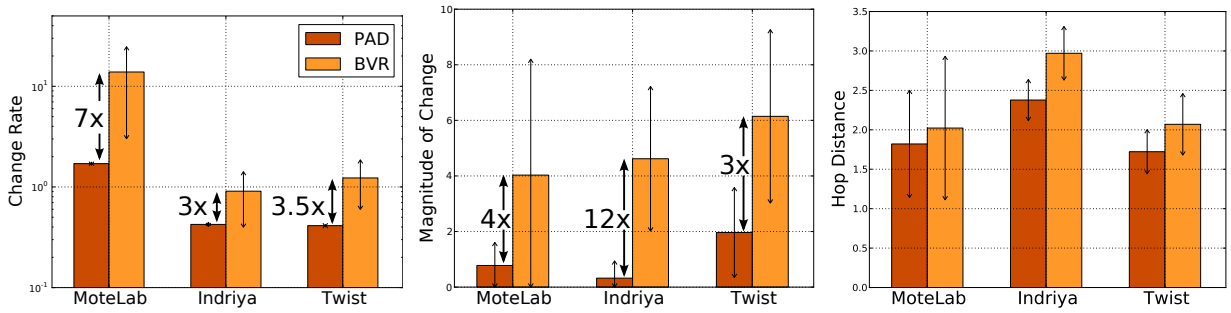
6.1 Experimental Setup

Our experimental setup is similar to the one for PAD evaluation in Section 4, except that now a set[8] of randomly selected sender-receiver pairs is defined, and each sender is to send a minimum of 1000 packets to its destination, one sender at a time. All the data traffic is sent after an initialization period of 15 minutes. The sender nodes are only informed once about the destination's virtual address at the beginning of the packet burst. Moreover, the sender and receiver nodes might be landmarks themselves as well. In the case of packet loss, we allow five routing level retransmissions[9] for each possible next hop, both in BVR and PAD. Each experimental run lasts for 30 minutes on MoteLab and 2 hours on TWIST.

Our prototype implementation of routing on PAD shares another aspect with BVR, i.e. the fallback mode and scoped

[8]The parameters of the routing test, such as sender-receiver pairs and the number of packets, are strongly dependent on the maximum time for experiments allowed on MoteLab.

[9]This is the default retransmission count in the original implementation of BVR.



(a) Average Change Rate (b) Average Magnitude of Change (c) Average Hop Distance from Landmarks

Figure 8: PAD achieves 7 times more stable addressing than BVR under MoteLab's challenging network conditions (notice the logarithmic scale). The error-bars represent the *stdev* of the nodes.

flooding (see [14] for details). The idea of the fallback mode is that in case a packet reaches a dead end, it is forwarded to the landmark closest to the destination. In case of PAD, as it does not maintain any explicit parents, routing towards the closest landmark is performed by selecting a neighbor that offers the minimum hop distance towards the landmark closest to the destination and qualifies the prerequisites such as link age and symmetry discussed in Section 5.2. Each node receiving the packet on the way to the landmark first tries the normal greedy routing mode and continues in fallback mode in case this fails again. If the packet reaches the landmark, this initiates a flooding of the packet with the scope as high as the path length *hop distance* from the landmark to the destination node (revealed by the destinations address). Similar to BVR, this mechanism incorporates the hope that the destination will receive the packet at least after the flooding scope has been reached. The inclusion of scoped flooding in PAD is not a way to enhance performance but to provide a reliable backup path and to complete the implementation for a fair comparison with BVR and S4.

6.2 Number of Transmissions

As our prototype implementation is for sensornets, our key performance metric is the *number of transmissions* required by a packet to reach its destination. Other factors like throughput are not considered here.

Figures 9(a) summarizes our results across the three testbeds. To observe the stability of results over time on MoteLab, we repeated our experiments 5 times for each protocol. The bars in Figure 9(a) show the average of 5 experiments while the error-bars show the highest and the lowest results among these experiments. The results clearly indicate that on MoteLab PAD outperforms both S4 and BVR by decreasing the number of transmissions by at least 26%. However, due to very stable link conditions on TWIST, the margin of improvement is just 7%. Figure 9(b) shows the CDF for the *number of transmissions* and Figure 9(c) details the results for a subset of sender-receiver pairs on MoteLab.

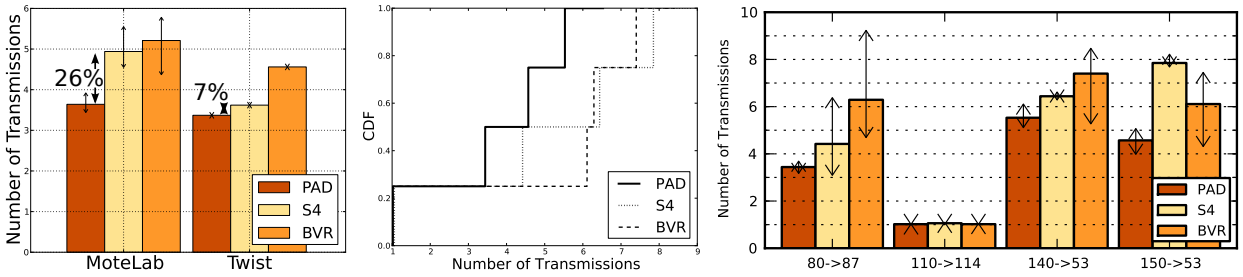
To understand the sanity of these results, we need to revisit a few mechanisms of S4. First, S4 is very conservative in its structure and does not rapidly adapt its topology to the changing underlying conditions in the network. Therefore, it employs retransmission of beacon packets to sustain its hybrid topological structure and maintain a small routing stretch. Second, S4 uses a link quality threshold of $PRR = 30\%$ (calculated using a passive WMEWMA estimator) to accept a link into its routing process. Using such links in a network dominated by low-quality links, without assessing their quality in the short-term, understandably decreases the number of routing choices and increases the number of transmissions in the network. In contrast, PAD incorporates rapidly changing conditions in its fuzzy addresses and assesses links based on very recent transmission conditions. Both these mechanisms of S4 explain the diversity of the results across different testbeds (cf. Figure 3). For example, the margin of improvement is quite high on MoteLab, whereas on TWIST the results are very comparable for all the three protocols.

We also evaluated the impact of landmark failures on transmission characteristics of PAD, such as the fraction of routes that directly arrived at the destination compared to the fraction of routes that required scoped flooding. Our results show similar trends as BVR's results [14]. Moreover, the routing success rate for all three protocols is quite high ($> 95\%$) in static testbed settings and are in alignment with S4's results [21]. This is because PAD, BVR, and S4 are all equipped with strong retransmission (5 retransmissions for each possible next hop) and backup mechanisms to achieve a high level of delivery reliability.

6.3 Communication Overhead

Here we take a closer look at PAD's communication overhead. Against the baseline of BVR, PAD introduces larger node addresses as they contain a node's coordinate history encoded as a probability distribution. The size of PAD's addresses heavily depends on the number of landmarks λ and the distribution size δ (see Section 3.2 for details) . The latter can be measured in terms of the number of hop-count values that have a non-zero frequency and therefore have to be included in the distribution. So λ landmarks and δ non-zero hop counts in the distribution result in an address length of $\lambda \cdot 2 \cdot \delta$ bytes. These larger node addresses in PAD impact the following three communication scenarios:

Local beacon updates: In this case, PAD allows to trade off transmission overhead against memory overhead. (1) Either a node's PAD address is included in its beacons, which increases the transmission overhead. (2) Or only the node's most recent coordinates are transmitted such that the neighbors that receive the beacons can compute the node's PAD address themselves (c.f. Section 3.3). This increases the CPU and memory overhead but does not introduce any transmission overhead against the baseline of BVR's beacon header. Moreover, one has to consider that PAD saves the transmission overhead of all the additional bytes appended



(a) Average Number of Transmissions (b) CDF of the Number of Transmissions in MoteLab (c) Number of Transmissions for a subset of sender-receiver pairs in MoteLab

Figure 9: A simple routing strategy over PAD reduces the number of transmissions in the network when compared with BVR and S4. The bars represent the average of 5 experiments and the error-bars show the highest and the lowest results.

with each data and beacon packet by BVR's link estimator.
Global address update: This update is required in the network's address database whenever a node changes its PAD address. The database interaction is beyond the scope of discussion in this paper. However, to put this overhead estimation into perspective, one has to consider that PAD needs significantly less address updates.
Data transmissions: Finally, each data packet needs to carry the destination address in its header. In our current implementation, we are only using the *mean* for each coordinate distribution in a PAD address. Hence, in its current state, PAD does not introduce additional overhead against the baseline of BVR's data-packet headers.

7. DISCUSSION AND RELATED WORK

The need for location independent addressing and routing schemes has long been realized since the emergence of multihop wireless communication systems such as ad hoc, mesh and sensor networks. These schemes are known for their simplicity, self-configurability, scalability, and for maintaining a constant routing state on each node in the order of the one-hop neighborhood size, making them particularly appropriate for resource-constrained sensornets. PAD has three complementing features that distinguish it from conventional location free addressing and routing approaches: (1) It assigns fuzzy addresses to nodes instead of sharp coordinates by analyzing link variability patterns without link estimation and explicit trees in the network, (2) it decouples addressing from routing allowing for quick adaptation of routing algorithms based on recent network conditions without compromising the stability of addressing, and (3) it embeds the information about all possible paths leading to a node in its address.

For our prototype evaluation we used BVR's greedy routing mechanism. However, we believe that the the same ideas of probabilistic address can be used with S4's intercluster routing approach as well. The functioning and performance of S4 is strongly dependent on a stable topology in which nodes can accurately estimate their distance from the nearest landmarks. To achieve this high level of stability and resilience S4 employs costly mechanisms, such as Resilient Beacon Distance Vector (RBDV), which retransmits a broadcast beacon until a specified number of neighbors have forwarded the same beacon. As a result, as we observed in Section 6, S4 can accomplish its goal – achieving a small routing stretch – without excessively increasing the *number of transmissions* only in very stable network conditions (e.g. TWIST). However, in testing conditions (e.g. MoteLab), S4 has to pay a high price of increased *number of transmissions* in the network for maintaining smaller routing stretches. PAD tolerates the need to maintain such a stable and resilient topology by assigning fuzzy locations to nodes and by allowing to adapt routing to very recent link conditions.

LCR [6] and BVR [14] are two very similar and notable implementations of virtual coordinate based addressing in sensornets. Both provide extensions based on link estimation for stable addressing in the presence of unreliable links in wireless networks. However, in Section 4.3.1 we already observed that long-term link estimation suffers in challenging network conditions experienced on MoteLab, as BVR's addressing showed instability and requires frequent address updates throughout the network. GEM [23] introduces a graph-based scalable addressing scheme. However, it employs a complex recovery process, in which a potentially large number of nodes in the system must recompute their addresses in case of node failure or radio link deterioration. In contrast, PAD provides an elegant solution to maintain address stability even in lossy networks.

Overall, PAD provides a flexible addressing platform that can host different routing strategies depending on application requirements while maintaining the scalability advantages of tree-based routing infrastructures.

8. CONCLUSION AND FUTURE WORK

We presented a robust and scalable addressing mechanism for wireless networks. When compared with other addressing mechanisms, PAD increases the stability and reduces the magnitude of change in addresses. An adaptive routing strategy over PAD allows quick adaptation of the routing paths based on very recent link conditions. Our results from testbed environments demonstrate that even an unoptimized version of routing over PAD can enhance packet delivery over multiple hops. Similarly, our tests under challenging environments such as in MoteLab show that PAD can realize its advantages in real world deployments.

We are still in the early phases of investigating suitable routing algorithms and distance functions, such as Gaussian distance, that can operate on PAD's addresses even more efficiently. Besides link dynamics and node failures, we are also analyzing how well PAD can support other sources of network dynamics such as node mobility. Finally, we want

to extend this work towards 802.11 networks to show that our approach has a broader relevance in the wireless domain.

Acknowledgments

Many thanks to our shepherd, Utz Roedig, and the anonymous reviewers for their feedback and insightful comments. This research was funded in part by the DFG Cluster of Excellence on Ultra High-Speed Mobile Information and Communication (UMIC), German Research Foundation grant DFG EXC 89.

9. REFERENCES

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, 2004.

[2] M. H. Alizai, O. Landsiedel, J. A. Bitsch Link, S. Goetz, and K. Wehrle. Bursty traffic over bursty links. In *SenSys'09*, Nov. 2009.

[3] M. H. Alizai, T. Vaegs, O. Landsiedel, R. Sasnauskas, and K. Wehrle. Poster abstract: Statistical vector based point-to-point routing in wireless networks. In *IPSN'10*, 2010.

[4] G. W. Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *IPSN*, 2005.

[5] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. In *SIGCOMM*, 2005.

[6] Q. Cao and T. Abdelzaher. Scalable logical coordinates framework for routing in wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(4), 2006.

[7] B. Chen and R. Morris. L+: Scalable landmark routing and address lookup for multi-hop wireless networks. *Massachusetts Inst. Technol. (MIT) Tech. Rep.*, 2002.

[8] J. Daintith. *Oxford Dictionary of Chemistry*. Oxford University Press.

[9] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom*, 2003.

[10] M. Demirbas, A. Arora, and M. Gouda. A pursuer-evader game for sensor networks. In *SSS*, 2003.

[11] M. Doddavenkatappa, M. C. Chan, and A. A.L. An experience of building indriya. In *National University of Singapore*, 2009.

[12] C. T. Ee, S. Ratnasamy, and S. Shenker. Practical data-centric storage. In *NSDI*, 2006.

[13] S. C. Ergen and P. Varaiya. Energy efficient routing with delay guarantee for sensor networks. *Wirel. Netw.*, 13(5), 2007.

[14] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *NSDI*, May 2005.

[15] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Comput. Commun. Rev.*, 33(1), 2003.

[16] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4), 2001.

[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *SenSys '09*, 2009.

[18] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *REALMAN*, 2006.

[19] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys*, 2003.

[20] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *SenSys '03*, 2003.

[21] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *NSDI*, 2007.

[22] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. In *IPSN*, 2010.

[23] J. Newsome and D. Song. Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *SenSys '03*, 2003.

[24] J. Ortiz, C. R. Baker, D. Moon, R. Fonseca, and I. Stoica. Beacon location service: a location service for point-to-point routing in wireless sensor networks. In *IPSN*, 2007.

[25] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MobiCom*, 2003.

[26] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: measuring wireless link burstiness. In *SenSys '08*, 2008.

[27] R. Szewczyk, J. Polastre, A. M. Mainwaring, and D. E. Culler. Lessons from a sensor network expedition. In *EWSN*, 2004.

[28] T. Vaegs, M. H. Alizai, and K. Wehrle. Probabilistic addressing in wireless networks. In *IEEE Student Conference, Hamburg, Germany*, 2010.

[29] M. Welsh, G. W. Allen, K. Lorincz, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Sensor networks for high-resolution monitoring of volcanic activity. In *SOSP*, 2005.