Pathfinder: Robust Path Reconstruction in Large Scale Sensor Networks with Lossy Links

Yi Gao[†], Wei Dong[*][†], Chun Chen[†], Jiajun Bu[†], Gaoyang Guan[†], Xuefeng Zhang[†], Xue Liu[‡]
[†]Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science, Zhejiang University
[‡]School of Computer Science, McGill University
{gaoyi, dongw, chenc, bjj, zhangxuefeng}@zju.edu.cn, guangaoyang@gmail.com, xueliu@cs.mcgill.ca

***Abstract*—In wireless sensor networks, sensor nodes are usually self-organized, delivering data to a central sink in a multi-hop manner. Reconstructing the per-packet routing path enables fine-grained diagnostic analysis and performance optimizations of the network. The performances of existing path reconstruction approaches, however, degrade rapidly in large scale networks with lossy links. In this paper, we propose Pathfinder, a robust path reconstruction method against packet losses as well as routing dynamics. At the node side, Pathfinder exploits temporal correlation between a set of packet paths and efficiently compresses the path information using path difference. At the PC side, Pathfinder infers packet paths from the compressed information and employs intelligent path speculation to reconstruct the packet paths with high reconstruction ratio. We evaluate several variations of Pathfinder as well as two most related approaches using traces from a large scale deployment and extensive simulations. Results show that Pathfinder outperforms existing approaches, achieving both high reconstruction ratio and low transmission overhead.**

I. Introduction

Recent years have witnessed the wide proliferation of wireless sensor networks (WSNs) with increasing scale in various applications [1], [2], [3]. In these networks, sensor nodes are usually self-organized, delivering data to a central sink in a multi-hop manner.

The routing path of each packet is useful for understanding the network performance [4], [5], [6]. Measurements on routing dynamics need per-packet routing path to analyze how frequently path changes occur. Measurements on packet losses need the path information to infer where packets are lost [6]. Further, many network diagnostic tools need path information for root cause inference [7], [8], [9]. With packet path information, network visibility can be greatly improved, enabling many effective network optimizations such as flow control and hot-spot detection.

The difficulty in obtaining per-packet routing path is due to two reasons. First, sensor networks are usually self-organized and dynamically changing. There is usually no prior knowledge about the underlying routing topology. Second, it is costly to directly attach path information in each packet since the overhead increases as the network scales. Different approaches have been proposed in the past years.

PAD [8] only attaches in each packet a two-byte field which is used to store one forwarder along the routing path. For an n hop path, PAD needs n packets to assemble the entire path. Although PAD is very lightweight, it cannot accurately infer the routing path with frequent path changes.

PathZip [10] attaches a 64-bit hash value in each packet, containing compressed path information. PathZip assumes the availability of each node's neighbor table. In order to reconstruct the routing path for each packet, the PC-side algorithm performs exhaustive search over neighboring nodes of all forwarders along the path for a match with the hash value. The computational overhead grows exponentially with increasing path length. In other words, given a limited computation time for each packet, the reconstruction ratio of per-packet path will be low for large scale networks.

MNT [4] exploits existing information in the packet header (e.g., the first-hop forwarder or parent for short) to reconstruct the routing path with the assumption that all nodes in the network generate local traffic. The key insight of MNT is to utilize the parent information in local packets originated from a given node (referred to as *helper packets*) to infer routing paths of packets passing the node. MNT has a low message overhead and is scalable to large scale networks. It, however, cannot work well in networks with high loss rate and high routing dynamics (see Section II.B).

In order to reconstruct per-packet path in large scale sensor networks with modest packet delivery ratio, relatively high routing dynamics, we propose Pathfinder, a novel path reconstruction method exploiting temporal correlation between packet paths to efficiently compress the path information.

Pathfinder also uses local packets from forwarders for inferring routing path for a forwarded packet. Different from MNT which enforces strict requirements on a set of helper packets on each forwarder along the path (see Section II.B), Pathfinder takes a much more optimistic way: it infers the packet path using parent information in only one packet in each forwarder based on the path difference information contained in each packet. Therefore, Pathfinder achieves better path reconstruction ratio with higher packet losses and routing changes. Pathfinder includes a protection byte in order to verify the correctness of the reconstructed path with a very high probability.

[*]Corresponding author. We gratefully acknowledge our shepherd Wenjing Lou and the ICNP reviewers. This work is supported by the National Science Foundation of China Grants No. 61202402 and 61070155, the National Key Technology R&D Program (2012BAI34B01), the Fundamental Research Funds for the Central Universities (2012QNA5007) and the Research Fund for the Doctoral Program of Higher Education of China.

Pathfinder includes a limited number of bytes to record the changed part of a routing path (i.e., the next hop of the forwarded packet does not equal to the parent of the local packet from each forwarder). A key observation from our measurement study on a real-world sensor network shows that for most packets, the changed part of a routing path keeps small even for a large network scale, e.g., ≤ 2 for 12-hop paths. Therefore, recording several hops additional information is useful for effective path reconstruction. Compared with PathZip, which compresses a whole routing path into a 64-bit hash value, Pathfinder only records the changed part in a way that the PC-side algorithm can reconstruct the entire path in an efficient manner, making Pathfinder computationally more scalable to large scale networks.

We implement Pathfinder in TinyOS 2.1.1 and evaluate its performance using traces from a large scale sensor network as well as extensive TOSSIM [11] simulations. We also implement two most related approaches, MNT [4] and PathZip [10], for comparison. Results show that Pathfinder significantly outperforms MNT and PathZip in various network settings.

Major contributions of this work are as follows.

- We identify the inefficiency of existing path reconstruction approaches in large scale sensor networks with lossy links and high routing dynamics.
- We propose Pathfinder, a novel path reconstruction approach: at the node side, Pathfinder exploits temporal correlation between a set of packet paths and efficiently compresses the path information using path difference; at the PC side, Pathfinder infers packet paths from the compressed information and employs intelligent path speculation to reconstruct the packet paths with high reconstruction ratio.
- We implement Pathfinder and compare its performance with two most related approaches using traces from a large scale real-world sensor network as well as extensive simulations. Results show that Pathfinder significantly outperforms MNT and PathZip in various network settings.

The rest of this paper is organized as follows: Section II describes the motivation of this work. Section III presents the design of Pathfinder. Section IV shows the evaluation of Pathfinder using traces from a large scale sensor network as well as extensive simulations. Section V discusses related work, and finally, Section VI concludes the paper.

II. Motivation

In this section, we first introduce the assumptions and network model used in our paper. We then illustrate our motivation via a small example. Finally, using measurement results from a large scale sensor network, we give system insights that Pathfinder builds upon.

A. Network Model

We assume a multi-hop data collection network with a single sink. The network employs a data collection protocol (e.g., CTP [12]) to collect data. Each sensor node generates and sends packets to the sink via multihop wireless. All nodes have a common packet generation period.

Table 1: Data fields in each packet

Field	Description
origin	The original node where the packet is generated.
parent	The next hop of the original node.
seqno	The sequence number which increases each time a data packet is transmitted.

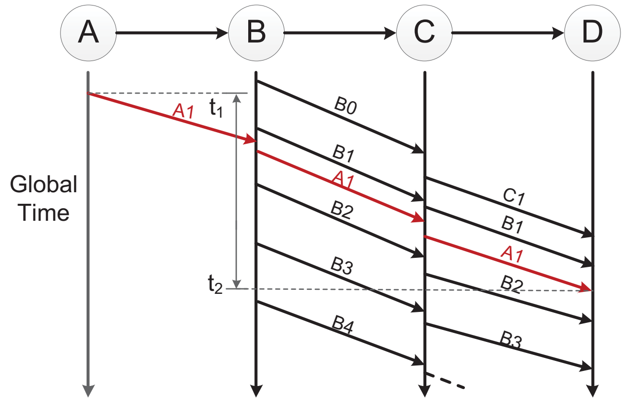


Fig. 1: A motivating example illustrating how MNT reconstructs the routing path of packet (A, B, 1). The packet is also denoted as A1 in the figure for short.

As shown in Table 1, each data packet includes the *origin*, *parent* and *seqno*. In the following sections, we will use a tuple (origin, parent, seqno) to represent a packet. We would like to reconstruct the entire routing path of the data packet.

B. Motivating Example

We will use an illustrative example to show the inefficiency of MNT and motivate the design of Pathfinder.

Figure 1 shows a routing path consisting of nodes A, B, C, D where D is the sink node. Each node (except the sink) transmits packets periodically. For example, A transmits packet (A, B, 1) and B transmits (B, C, 1), (B, C, 2), (B, C, 3), (B, C, 4). Consider how we reconstruct the entire routing path for packet (A, B, 1).

MNT adopts the following approach:

1) It locates (A, B, 1) in the packet trace seen at sink D. The arrival time is t_2 in sink's clock.
2) It infers the generation time of (A, B, 1) as t_1 in sink's clock. It is feasible as each packet can be attached with a network sojourn time which accumulates hop by hop [13].
3) In order to infer B's next hop for packet (A, B, 1), MNT locates all packets originated from B, with generation time residing in (t_1, t_2) and two adjacent packets, e.g., (B, C, 0), (B, C, 1), ..., (B, C, 4) for this example. These packets are referred to as *helper packets*.
4) If all three helper packets originated from B (a) are received at the sink (i.e., no loss), and, (b) have the same next hop, say node C, it can be inferred that C is on the routing path of (A, B, 1).

5) In order to infer C's next hop for packet (A, B, 1), it performs similar steps as in (3)–(4), i.e., locating the helper packets at C and use these packets for reconstruction if the two requirements described in step (4) are satisfied at C. In this example, it can be inferred that D is the next hop after C. Hence the entire path for (A, B, 1) is (A, B, C, D).

We can see that MNT can successfully reconstruct the routing path for (A, B, 1) in the above example. However, it fails to reconstruct the path in the following two scenarios.

1) **Packet Losses.** Suppose that the second packet originated from B is lost. In this case, C is not ensured on the routing path since the next hop of the lost packet cannot be determined. There is ambiguity in determining B's next hop for packet (A, B, 1).
2) **Routing dynamics.** Suppose the second packet originated from B now takes a different next hop, i.e., we can represent the packet as (B, C', 2) where C' is the parent. In this case, there is also ambiguity in determine B's next hop for packet (A, B, 1) since B's next hop could be either C or C'.

MNT takes a conservative approach: if either of the above two scenarios occur, packet (A, B, 1)'s routing path cannot be reconstructed in MNT.

We can see that MNT requires that all helper packets at all forwarders successfully arrive at the sink with the same next hop. With increasing packet losses and routing dynamics, the reconstruction ratio of MNT decreases.

Instead of using a set of helper packets to infer the routing path, Pathfinder uses only one previous packet from B, e.g., packet (B, C, 1) which is also referred to as *reference packet* in this paper. Pathfinder is able to reconstruct the routing path in the above two scenarios.

1) **Packet losses.** If packets originated from B after (A, B, 1) (i.e., (B, C, 2) or (B, C, 3)) are lost, the path reconstruction in Pathfinder is also not affected since Pathfinder relies on the reference packet (B, C, 1). If the reference packet (B, C, 1) is lost, Pathfinder can also perform opportunistic inference based on other packets from B at the sink.
2) **Routing dynamics.** If packets originated from B after (A, B, 1) change parent, the path reconstruction in Pathfinder is also not affected. If packet (A, B, 1)'s next hop at node B is C' instead of C, Pathfinder will include limited fields in packet (A, B, 1) to record C'. Pathfinder fails to reconstruct the path if the path difference (i.e., the number of nodes which need to be recorded) exceeds a threshold, say 2. Our measurement from a real-world large scale sensor network reveals that the actual path difference is usually small. Hence Pathfinder can reconstruct most of the packet paths in practice.

Using one reference packet instead of a set of helper packets is the key difference between Pathfinder and MNT. We need to address two specific challenges in realizing the idea. First, Pathfinder needs to efficiently compress the information whether the next hop of a forwarded packet is the same as the parent of a reference packet at the node side. Second, Pathfinder needs to accurately localize the reference packet for effective reconstruction of the routing path at the PC side. Finally, Pathfinder uses an intelligent path speculation method to further reconstruct more routing paths.

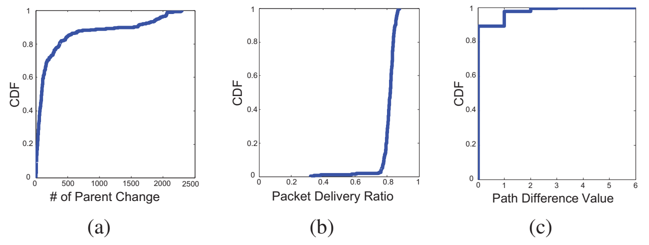


Fig. 2: Measurement results. (a) The number of parent changes during the whole measurement study. (b) The CDF figure of PDR of all nodes. (c) The CDF figure of path difference value of all nodes.

C. Measurement Study

To understand the routing behaviors in a real-world large scale sensor network, we conduct a measurement study on CitySee [1]. CitySee includes 1200 wireless sensor nodes deployed in an urban area. Each node transmits multi-dimensional sensing data (e.g., CO_2, temperature, humidity, light, etc.) with a period of 10 minutes for further analysis. Collection Tree Protocol (CTP [12]) is used for multi-hop data collection. The first 10 hops are attached in each packet for further analysis. In this paper, we analyze packet trace from a subnet with 270 nodes in a measurement period of one week. In total, there are 865919 packets in the trace.

Figure 2(a) shows the CDF of the number of parent changes during the measurement period. On average, there is a parent change every 9.6 packets. As a comparison, there is a parent change every 88.2$\sim$793.3 packets in traces used by MNT [4]. High routing dynamics can degrade MNT's performance (as illustrated in the previous subsection) and call for a robust path reconstruction approach against routing dynamics.

Figure 2(b) shows the CDF of packet delivery ratios (PDRs) of all the nodes in the network. On average, the PDR is about 80% as opposed to about 99% in traces used by MNT. High loss rate can degrade MNT's performance. As reported in the MNT paper [4], the path reconstruction ratio decreases to $<$50% when PDR equals to 80%. Hence, we need a robust path reconstruction approach against packet losses.

A key observation from our measurement study is that the path difference is usually small albeit relatively high loss rate and high routing dynamics. Intuitively, *path difference value* denotes the number of actual forwarders which differ from the parents in reference packets. Those forwarders need to be explicitly recorded in data packets for effective path reconstruction. Formally, we define path difference value of a packet as follows.

Definition 1 (Path difference value): Let $path(pkt_A) = (A, f_1, ..., f_n, sink)$ be the actual path of a packet pkt_A = (A, f_1, 1). At each forwarder f_i ($1 \le i \le n$), we denote the parent of pkt_A's reference packet as par_i. Let I_i be an indicator

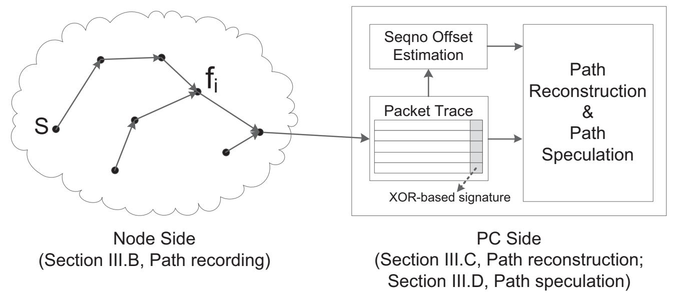

S
f_i
Seqno Offset Estimation
Packet Trace
XOR-based signature
Path Reconstruction & Path Speculation
Node Side (Section III.B, Path recording)
PC Side (Section III.C, Path reconstruction; Section III.D, Path speculation)

Fig. 3: Two components of Pathfinder.

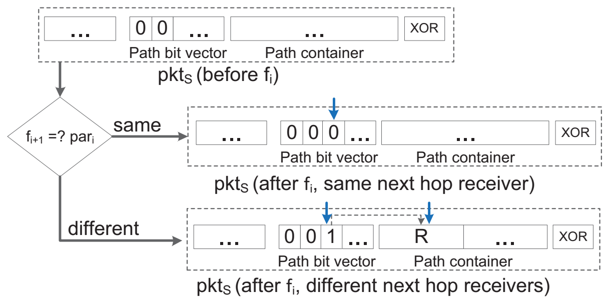


Fig. 4: pkt_S is updated at the forwarding node f_i. According to whether pkt_S's next hop receiver is the same as the parent of the reference packet, pkt_S will be updated differently.

variable representing whether packet pkt_A's next hop at f_i is the same as par_i, i.e.,

$$I_i = \begin{cases} 0; & \text{if } f_{i+1} \equiv par_i \\ 1; & \text{otherwise} \end{cases} \tag{1}$$

We define *path differenc value* of a packet to be the sum of I_i along the routing path, i.e., $\sum_{i=1}^{n} I_i$.

If path difference value equals to zero, we can use parent information in reference packets for path reconstruction. Otherwise, we need additional overhead in recording the actual forwarders. To be useful, a small path difference value is desirable to make the message overhead small.

Figure 2(c) shows the CDF of path difference values of all packets. More than 89% of the path differences are equal to zero and about 98% of them are smaller than two. This observation motivates us to reconstruct the packet paths based on efficiently compressed path difference information.

III. Design

In this section, we introduce the main design of Pathfinder, a robust path reconstruction approach for large scale WSNs with lossy links.

A. Overview

Pathfinder consists of two main components for path reconstruction as shown in Figure 3. At the node side, the path recording component exploits temporal correlation among a set of packet paths and efficiently compresses the path information using path difference. At the PC side, the path reconstruction component infers packet paths from the compressed information and employs intelligent path speculation to reconstruct the packet paths with high reconstruction ratio.

In order to record the path difference efficiently, Pathfinder includes a path bit vector and a path container in each data packet. One bit in the path bit vector indicates whether the next hop of a forwarded packet at a particular forwarder is the same as the parent of its reference packet (i.e., $I_i \equiv 0$ in Equation 1). If yes, no information needs to be recorded in the path container. Else, Pathfinder records the actual next hop of the forwarded packet. Section III.B introduces the details of path recording. Pathfinder additionally adds an XOR-byte for path verification. In order to reduce the message overhead, Pathfinder exposes a limit on the size of the path container, e.g., ≤ 2 in our current implementation. Besides, Pathfinder employs Huffman encoding to compress the bit vector.

When data packets are received at the sink, Pathfinder uses information contained in data packets for path reconstruction. Section III.C introduces the basic path reconstruction process with an emphasis on how to accurately localize the reference packets. There are possibilities that the above path reconstruction fails due to multiple reasons (e.g., limited path container size, loss of the reference packet, and etc). Section III.D introduces an intelligent path speculation method to further enhance the path reconstruction capability.

It is worth noting that our current implementation of Pathfinder only reconstructs the paths of packets which are successfully received at the sink. For those packets lost in the network, reconstructing their (partial) paths need extra information. A possible way to capture their (partial) paths is to deploy multiple sniffers in the network. Packet traces captured by those sniffers can be fed into the Pathfinder algorithm for path reconstruction.

B. Path Recording

At the node side, path difference of each packet is recorded in three data structures, bit vector, container and XOR-byte which are updated hop-by-hop.

When a packet pkt_S originated from node S is delivered to a forwarding node f_i, f_i compares the next hop of pkt_S after f_i (f_{i+1}) with the parent of pkt_S's reference packet (par_i). If they are the same, f_i appends a 0 to pkt_S's path bit vector. Otherwise, f_i appends a 1 to the bit vector, and, at the same, records the actual next hop after f_i to pkt_S's path container if the size of the container does not exceed its limit. In cases that the path difference is larger than the container's limit due to severe routing dynamics, the path difference cannot be recorded completely. In practice, the path difference is usually very small as shown in the measurement study. Even when the path difference is not recorded completely, an intelligent path speculation (Section III.D) is further used to reconstruct the routing path.

Figure 4 illustrates how path information of pkt_S is updated at node f_i.

The XOR-byte XORs all the least significant bytes of the travelled nodes IDs together. It is used to verify whether a reconstructed path is correct or not.

A key observation revealed in Section II shows that although there exists relatively high loss rate and routing dynamics in real-world large scale sensor networks, the path difference, however, is usually small, e.g., ≤ 2. Based on this observation, we optimize the message overhead in Pathfinder as follows.

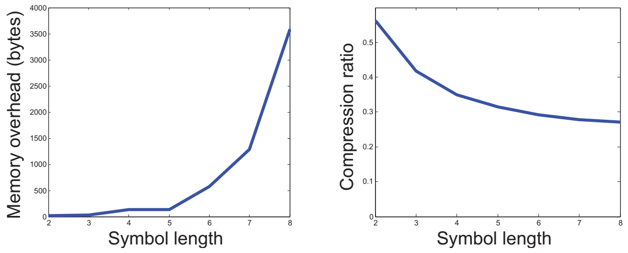
First, Pathfinder enforces size limits for both the bit vector and container. The maximum bit vector size is 2 bytes which corresponds to a maximum hop count of 16 without compression. The maximum container size is 4 bytes which corresponds to a maximum of two node IDs. We see from Figure 2(c) that the path difference of 98% packet paths are less than or equal to two. Considering the additional 1-byte overhead of XOR, the maximum message overhead of Pathfinder is 9 bytes which is similar with the 8 bytes hash value overhead in PathZip. The average message overhead of Pathfinder, however, is significantly smaller than PathZip since path difference is usually small (Section IV).

Second, there should be many 0s in the bit vector because of a small path difference. Hence, the bit vector can further be compressed. We employ Huffman encoding which is an entropy encoding algorithm used for lossless data compression. The Huffman encoding algorithm encodes a stream of symbols according to their frequencies of occurrences. A symbol frequently occurs will be assigned to a short code, hence the total size of the symbol stream can be greatly reduced on average. In our case, the input of the algorithm is a *uncompressed* bit vector. Pathfinder treats a fixed length of bits as symbols. We call the length as symbol length. For example, when the symbol length is 3, a bit vector of 000001000010 consists of 4 symbols, i.e., 000, 001, 000, 010. Pathfinder performs Huffman encoding according to a locally stored encoding table which is built a priori based on our measurement data. For example, symbol 000 can be encoded into 0, symbol 001 can be encoded into 10 and symbol 010 can be encoded into 11. Therefore, the original bit vector is encoded into 010011 which is smaller than the original bit vector. Pathfinder further optimizes the case when path difference is 0: no bit vector or container information needs to be recorded. Since the bit vector is updated on each hop, each forwarder first decodes the bit vector, then updates it and encodes the updated bit vector again.

There is a key parameter in Huffman coding, the symbol length. A longer symbol length can achieve higher compression ratio but needs more memory to store the encoding stable. Based on our measurement data, Figure 5 shows the memory overhead and compression ratio under different symbol lengths. Considering both the memory consumption and the achievable compression radio, we configure a symbol to be five bits. This configuration translates to about 100 bytes memory overhead (which is small compared to a total of 10KB RAM on TelosB [14] nodes) and 1/3 compression ratio.

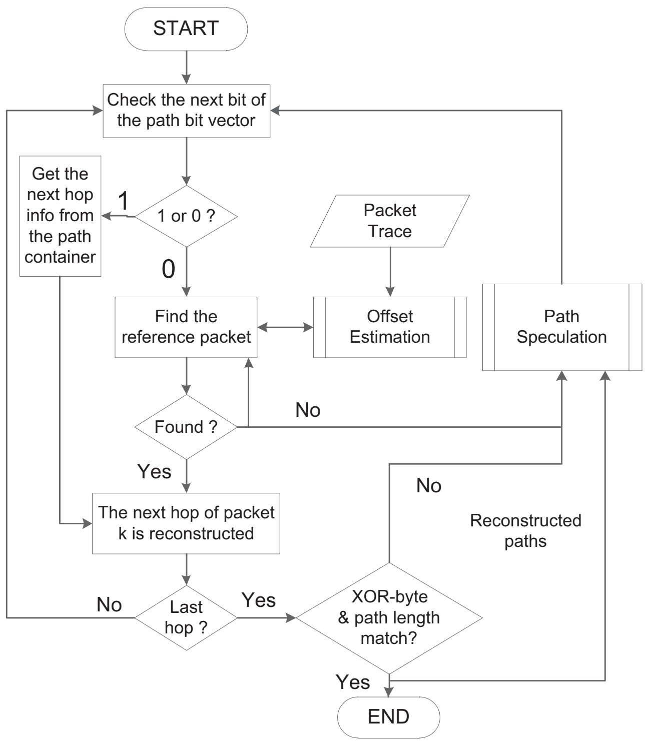
C. Path Reconstruction

After obtaining the packet trace, Pathfinder reconstructs the packet path at the PC side. The left part of Figure 6 shows the basic reconstruction procedure. For a received packet pkt_S originated from node S, the sink scans the uncompressed bit vector from the left to the right in each iteration. In the i-th iteration, Pathfinder tries to reconstruct the i-th hop forwarder. If Pathfinder finds a zero in the bit vector, it needs to locate pkt_S's reference packet. The parent of the reference packet is the i-th hop forwarder of pkt_S. If Pathfinder finds a one in the path bit vector, it gets the i-th hop forwarder from the path container if it is recorded.



(a) Memory overhead (b) Compression ratio

Fig. 5: Memory overhead and compression ratio under different symbol lengths.



START
Check the next bit of the path bit vector
1 or 0 ?
1
Get the next hop info from the path container
0
Packet Trace
Find the reference packet
Offset Estimation
Path Speculation
Found ?
No
Yes
The next hop of packet k is reconstructed
No
Reconstructed paths
No
Last hop ?
Yes
XOR-byte & path length match?
Yes
END

Fig. 6: The complete workflow of reconstructing a path.

A key problem in Pathfinder is how to accurately localize reference packets for a given packet. If all reference packets are accurately identified, the reconstructed path is guaranteed to be correct. However, it is challenging due to several practical reasons.

First, the reference packet may even be lost. In this case, path reconstruction fails since where is no parent information in the reference packet. We tackle this problem by path speculation described in the next subsection (Section III.D).

Second, the arriving order of a forwarded packet and its

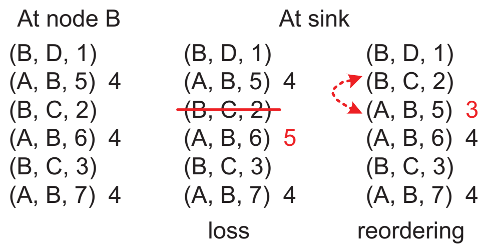


Fig. 7: An example illustrating why the simple method cannot locate the reference packet accurately.

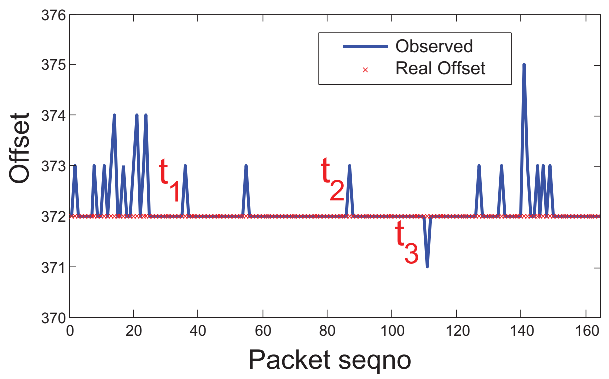


Fig. 8: An observed offset sequence of two nodes in a deployed network.

reference packet may be different at the forwarder and the sink. Figure 7 shows an example, showing the arriving order of packets at a forwarder B and the sink. There are three packets originated from A, forwarded at node B. In addition, there are three packet originated from B. From B's viewpoint, the reference packet for a packet originated from A can be easily identified as the latest local packet from B before the packet originated from A (since B determines the reference packet). For example, the reference packet of (A, B, 5) is (B, C, 1), the reference packet of (A, B, 6) is (B, C, 2), and the reference packet of (A, B, 7) is (B, C, 3). The viewpoint of the sink might be different from B's viewpoint due to packet loss or out-of-order arrival. If we still identify reference packet as the latest arriving packet from B, there are likely false identifications. In the case of packet loss shown in Figure 7, (B, D, 1) will be misidentified as (A, B, 6)'s reference packet (while no reference packet is actually received at the sink). In the case of packet reordering shown in Figure 7, (B, C, 2) will be misidentified as both (A, B, 5) and (A, B, 6)'s reference packet.

The misidentification of a reference packet will likely cause path reconstruction failures (when the XOR values do not match) or false path reconstruction (when the XOR values occasionally match). However, there will be no negative impact when the misidentified packet occasionally has the same parent as the reference packet (or even benefit when the reference packet is lost). We implement Pathfinder-simple, a simple version which identifies the reference packet as the latest local packet before a forwarder packet and evaluate its performance in Section IV.

In general, reference packet identification is a difficult problem because of the discrepancy between a forwarder's view and the sink's view. However, under certain conditions, it can be identified very accurately.

Notice that in each packet there is a sequence number field which increments at the original node each time a packet is generated and transmitted. When all network nodes transmit packets with a common period which is sufficiently larger than normal packet delays, the sequence number offset between a forwarded packet and its reference packet usually keeps stable. For the example shown in Figure 7, if we can accurately estimate the sequence number offset between A's packet and its reference packet from B as 4, the reference packet can still be be correctly identified.

The observed sequence number offset at the sink differs from the ground truth because of packet loss or packet re-

Algorithm 1 Algorithm for sequence number offset estimation

```
Input: A recent offset sequence
Output: $v_i$: the estimated offset of the source and forwarder
1: procedure OFFSET-ESTIMATOR
2:     Let $\omega = (s_1, \ldots, s_n)$ be the offset sequence
3:     $c(v_i) = 0$
4:     for $i$ = 2 to length($\omega$) do
5:         if $s_i \equiv s_{i-1}$ then
6:             increase $c(v_i)$ where $v_i = s_i$
7:     return $v_i$ where $c(v_i)$ is the maximum
```

ordering. However, the sequence of offsets exhibits easy-to-recognize statistical features.

Figure 8 shows an offset sequence in the CitySee network. The red line shows the ground-truth offsets and the blue line shows the observed offsets at the sink. The peaks are caused by packet reordering or packet losses. For example, at time instants t_1 and t_2, there are packet losses, making observed offsets larger. At time instant t_3, there is a packet reordering, making the observed offset smaller. We can see that the number of real offset values is statistically much larger than other values caused by packet loss or packet reordering.

This observation motivates us to devise Algorithm 1 for accurately estimating the real offset. Pathfinder first calculates a *offset sequence* at the sink. Each offset in the offset sequence is calculated by the sequence numbers of a packet and its reference packet located by the method in Pathfinder-simple. As shown in Figure 7, the offset sequences in case of packet loss and reordering are (4, 5, 4) and (3, 4, 4), respectively. We introduce a confidence value which is increased whenever two consecutive offsets are the same (line 5, 6). Then the estimated offset v_i is the one with the maximum confidence value $c(v_i)$.

We employ the above algorithm and implement Pathfinder-accurate for accurately identifying the reference packets. The accuracy of the algorithm will be affected by several factors.

First, it is affected by packet loss and packet reordering if they frequently occur. However, in practice, the estimated offset is highly accurate. In particular, we can prove the following theorem.

Theorem 1: Considering packet loss alone, Algorithm 1

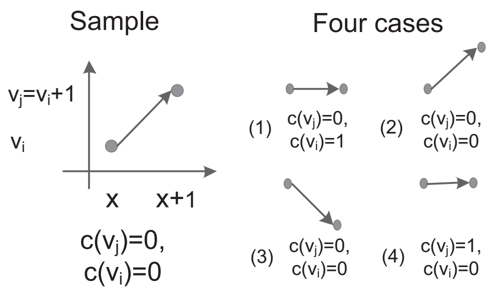


Fig. 9: Four cases of the two consecutive offsets.

can estimate the offset with guaranteed correctness when the packet loss rate is below 25%.

Proof: In order to prove the theorem, we calculate the *lowest* loss rate required to make the estimated offset be incorrect. Assume v_i is the correct offset and the loss rate reaches minimum, $v_j = v_i + 1$ must be the wrong one. The reason is that for any $v_{j'} > v_j$, more packet losses are needed and the lowest loss rate cannot be reached. In order to make v_j be the estimated offset, its confidence $c(v_j)$ needs to be larger than $c(v_i)$.

Consider two consecutive offsets (s_x, s_{x+1}) and each offset is either v_i or v_j, then there are four different cases shown in Figure 9. Case (2) and case (3) have $c(v_j) = c(v_i)$ and case(4) has $c(v_j) > c(v_i)$. At least four packets are needed to form two consecutive offsets, including two packets from the source node and two packets from the forwarder. Assume these four packets are $(B1, A1, B2, A2)$. When $B2$ is lost, the offset sequence is $(seq_{A1-B1}, seq_{A2-B1})$ which matches case (2). When $B1$ is lost, the offset sequence matches case (3). Case (4) needs more packet losses. The lowest loss rate can be achieved when the offset sequence includes many case (2) or (3) and only one case (4). Since $c(v_j) = c(v_i)$ in case (2) or (3) and $c(v_j) > c(v_i)$ in case (4), $c(v_j) > c(v_i)$ holds for the whole sequence. Therefore, at least 25% loss rate is required to make the estimated offset be incorrect. That means when the loss rate is below 25%, Algorithm 1 can estimate the offset with guaranteed correctness. ■

This means our algorithm is resilient to packet loss.

Second, the algorithm imposes requirements on the upper-layer application. It requires all network nodes transmit local packets with a fixed inter packet interval (IPI) which is sufficiently larger than the normal network delay. Otherwise, even the ground-truth offsets are unstable, impairing the effectiveness of our algorithm. Hopefully, many sensor networks, e.g., GreenOrbs [2], CitySee [1], Clinic monitoring [15], PermaSense [16], SensorScope [17], satisfy the above requirements, making our algorithm practically usable for these applications.

There are still chances that we incorrectly estimate the offset. We address this problem in two ways. First, the XOR byte provides a mean for verification, which clearly identifies wrong reconstructed path with a high probability. Second, the path speculation algorithm (described in the next subsection) can further enhance the path reconstruction capability at the cost of higher computation overhead.

Algorithm 2 Path speculation

```
Input: $P$: received packets set; $R$: all reconstructed paths; $k$:
    the packet whose path is being reconstructed
Output: $x$: packet $k$'s reconstructed path
 1: procedure PATH-SPECULATION
 2:     for each reconstructed path $r_i$ with origin node $i$ do
 3:         if $f \in r_i$ and $f \notin F(i)$ then
 4:             insert $f$ to $F(i)$
 5:     $G(i)$ = CONSTRUCT-G($P$, $R$, $F(i)$)
 6:     for each path $x$ from $i$ to sink in $G(i)$ do
 7:         if length, XOR-byte, bit vector or
 8:            container of $x$ does not match $k$ then
 9:             continue
10:         if maximum time limit exceeds then
11:             break
12:         return $x$

13: procedure CONSTRUCT-G($P$, $R$, $F(i)$)
14:     let $G(i)$ be a graph
15:     insert each node in $F(i)$ to $G(i)$
16:     for $p$ in $P$ do
17:         if source($p$) $\in F(i)$ and parent($p$) $\in F(i)$) then
18:             add an edge from source($p$) to parent($p$)
19:     for $r$ in $R$ do
20:         for each two consecutive hops $u$, $v$ in $r$ do
21:             if $u \in F(i)$ and $v \in F(i)$) then
22:                 add an edge from $u$ to $v$
23:     return $G(i)$
```

D. Path Speculation

When path reconstruction fails (i.e., the XOR values do not match), Pathfinder employs path speculation to further enhance the reconstruction capability.

There are several challenges to speculate the path. First, if the reconstructed path is not the actual one (since XOR values do not match), we do not get fine-grained knowledge which nodes match the actual ones. Therefore, we cannot enumerate neighbors of the matched nodes for efficient speculation. Second, enumerating all possible neighbors from the original node like PathZip causes a significant computation overhead, making it less scalable to large scale networks.

The key insight of our approach is to exploit the already reconstructed paths. Due to temporal correlation, packets tend to follow links consisting of already seen forwarders towards the sink. Based on this insight, Pathfinder performs path speculation as follows.

Within a time window, we use $F(i)$ to denote the set of nodes that have forwarded packets for node i. After we have the $F(.)$ for each node, we try to add directed edges in each $F(.)$ according to the received packets. For two nodes a and b in $F(i)$, we add a directed edge from a to b if there exists at least one packet from a to b. We then have a directed graph $G(.)$ for each node. Finally, we try to enumerate all possible paths in $G(i)$.

Algorithm 2 describes the above process. First, $F(i)$ is constructed according to the reconstructed paths (line 2, 3, 4). Then the Construct-G subroutine constructs the directed graph

$G(i)$ (line 13∼23). Specifically, an edge is added to the graph whenever there exists one packet, local packet (line 16∼18) or forwarded packet (line 19∼22), has been delivered on that edge. Finally all possible paths for node i are enumerated (line 6) considering the constraints on path length, XOR-byte, bit vector and container (line 7, 8, 9). The bit vector and container in the packet can be used to ensure the correctness of the reconstructed path. For example, a bit vector $\{0010000\}$ and a container $\{20\}$ indicate the third hop (after the parent) of the packet is node 20.

In practice, Pathfinder employs a batch approach to speed up the path speculation. For multiple packets from the same node, Pathfinder only enumerates the possible paths once and cross validates the enumerate paths and the paths being reconstructed. It further improves the efficiency of path speculation as well as reconstructs more paths in the time limit.

PathZip has to enumerate about d^h possible paths where d is the node degree and h is the hop count of the received packet. The number of already seen forwarders for a given packet within a time window, however, is much smaller. Therefore, our path speculation algorithm can enumerate possible path more efficiently.

IV. Evaluation

In this section, we evaluate the performance of Pathfinder using traces from a large scale deployment and extensive simulations. In Section IV.B, we perform trace-driven study to evaluate the performances of Pathfinder and existing approaches in a real-world deployed network. In Section IV.C, we perform extensive simulations to reveal more system insights.

A. Evaluation Setup

We implement Pathfinder on TinyOS 2.1.1 with CTP [12] as the data collection protocol.

To evaluate Pathfinder's performance, we perform a trace-driven study using traces from the CitySee [1] project. The CitySee application uses the TinyOS LPL MAC protocol [18], the CTP routing protocol [12] for multihop routing, and the Drip dissemination protocol for disseminating and configuring key system parameters. The trace includes all received packets from a subset of more than 270 nodes. We use the collected information to reproduce each node's local events which are further used to help path recording. The MNT approach relies on the timestamping mechanism [19] to obtain the global packet generation time. In addition to the natural packet losses, we further inject more packet losses in order to investigate how Pathfinder performs in networks with higher loss rate.

In order to reveal more system insights, we perform extensive simulations in multiple networks with 255∼900 nodes. We use TOSSIM [11] as the simulator. The routing path is recorded in each data packet for validation. We place nodes uniformly distributed in a square area with the sink at the center. The transmission range is configured to generate different network settings without breaking network connectivity.

The path reconstruction algorithm is executed on a desktop PC with a dual-core 3.1GHz CPU and 4GB RAM. The path reconstruction of each packet has a time limit of one second for Pathfinder as well as PathZip.

Table 2: Performance comparison of different methods

Methods	Reconstruction Ratio	False Reconstruction Ratio
MNT	77.1%	0
PathZip	61.0%	0
Pathfinder-simple	34.6%	11%
Pathfinder-accurate	98.8%	1.4%

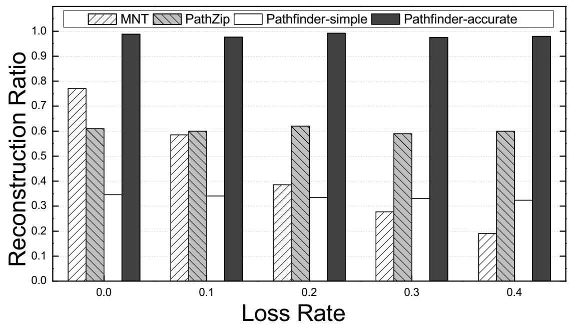


Fig. 10: The reconstruction ratios of four different methods in the trace-driven study.

B. Trace-driven Study

We evaluate MNT [4], PathZip [10] and two versions of Pathfinder (i.e., Pathfinder-simple, and Pathfinder-accurate) in the trace-driven study. As described in Section III.C, Pathfinder-simple is a variation of Pathfinder in which reference packets are identified as the latest packets before the forwarded packets while Pathfinder-accurate uses offset estimation to accurately identify the reference packets. In Pathfinder-simple, the path speculation is not used for simplicity.

Reconstruction ratio. Table II shows the evaluation results of four different methods for the CitySee trace. We can see that Pathfinder-accurate achieves the highest reconstruction ratio with a low false reconstruction ratio. MNT cannot achieve high reconstruction ratio in the trace-driven study since there are severe packet losses and routing dynamics in the trace. As described in Section II.B, MNT cannot successfully reconstruct many packet paths in case of severe packet losses and routing dynamics. The performance of PathZip is limited by the network scale and neighbor size of each node. In the trace-driven study, the search space of PathZip is too large to be enumerated in limited time. Pathfinder-simple suffers from the severe packet losses and routing dynamics, resulting in poor performances.

Message overhead. The message overhead is the number of bytes added to each packet on average. We assume each packet already includes the CTP header (including origin, sequence number, hop count) We assume two bytes are used to represent a node ID. The message overhead of MNT is six bytes, including the parent ID, and the 4-byte timestamp. PathZip has a overhead of 8-byte MD5-like hash value. Pathfinder has a variable overhead with a maximum overhead of 9 bytes including parent node ID (2 bytes), XOR-byte (1 byte), bit vector ($\leq$2 bytes) and container ($\leq$4 bytes). However, the average overhead in Pathfinder is significantly smaller. In the CitySee trace, the average message overhead of Pathfinder is 3.9 bytes, much smaller than the fixed message overhead of both MNT and PathZip.

Impact of loss rate. We inject packet losses into the trace and evaluate the performances of the four methods. As shown in Figure 10, the reconstruction ratio of Pathfinder is still very high when there are severe packet losses. The performance of MNT drops rapidly when packet loss increases. PathZip has a stable performance when loss rate increases. The performance of Pathfinder-accurate performs significant better than Pathfinder-simple. The reason is that a simple approach to identifying reference packets cannot yield accurate results, especially when there are high routing dynamics.

C. Simulations

Figure 11 illustrates our simulation results. Figure 11(a) shows the path lengths of all received packets. On average, the path length is about 10 hops. Based on the recovered paths, Figure 11(b) shows a contour figure of the number of forwarders for each source node. Since the sink is at the center, nodes far away from the sink have more forwarders to help forward their packets. Figure 11(c) and Figure 11(d) show the recovered paths of several packets and all paths of a node at the right bottom area. Figure 11(d) reveals the severe routing dynamics in the network.

Next, we mainly investigate the path reconstruction performance with varying network scale and packet loss ratio.

Impact of network scale. Figure 12 shows the reconstruction ratios and the false reconstruction ratios of four different approaches when the network size varies from 225 nodes to 900 nodes. All approaches perform well when the network size is 225 nodes. When the network scales up, MNT and PathZip's performance degrade and Pathfinder still has high reconstruction ratio.

Impact of packet losses. Figure 13 shows the total reconstruction ratios and the false reconstruction ratios of four different approaches when the loss rate varies from zero to 40 percent. Pathfinder achieves the highest reconstruction ratio and a low false reconstruction ratio under all loss rates. On average, Pathfinder has a reconstruction ratio of 97.1% and a false reconstruction ratio 2.3%. Pathfinder-simple also achieves high reconstruction ratio, but the false reconstruction ratio increases rapidly when loss rate increases. Pathfinder-simple's performance in the simulations is much better than in the trace-driven study. The reason is there is low routing dynamics in the simulation. Although there are also packet losses, Pathfinder-simple may find a wrong reference packet with the same parent as in the reference packet. MNT cannot achieve high reconstruction ratio even when the loss rate is very low. We looked into the trace and found the reason was due to the parent changes. When the loss rate increases, MNT's performance degrades. As described in MNT's paper, MNT does not introduce any false reconstructed paths. PathZip also does not introduce any false reconstructed paths due to a large hash size(64 bits), but has low reconstruction ratios under all loss rates. The reason is there are many long paths which are difficult to be enumerated in a limited time in the network with 900 nodes.

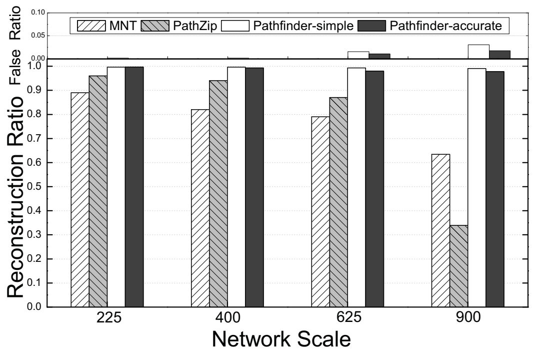


Fig. 12: The reconstruction ratios and false reconstruction ratios of four different methods with different network scales in the simulations.

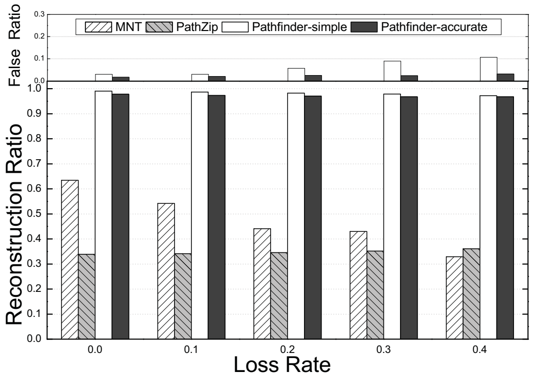


Fig. 13: The reconstruction ratios and false reconstruction ratios of four different methods with different loss rates in the simulations.

V. Related Work

The related works of Pathfinder can be grouped into three categories: network debugging, wired network tomography and path reconstruction in WSNs.

A. Network Debugging and Diagnosis

In order to detect the root causes of network failure, many approaches have been proposed to debug and diagnose the network. PAD [8] uses a packet marking method to obtain the travelled path which is used to diagnose network anomalies. Since multiple packets are needed to reconstruct a path, the network is required to be static. Sympathy [9] selects network metrics that enable efficient failure detection on PC side. Path information reconstructed by Pathfinder could potentially benefit those approaches in terms of detection accuracy and granularity.

B. Wired Network Tomography

In IP wired network, network tomography techniques are used to measure the network internal behaviors. This problem has been well-studied and many approaches [20], [21], [22] have been proposed in the literature. Different from wireless networks, these wired network tomography mainly focus on using probes to measure the underlying network behaviors. In the scenario of WSNs, using probes is usually not desirable

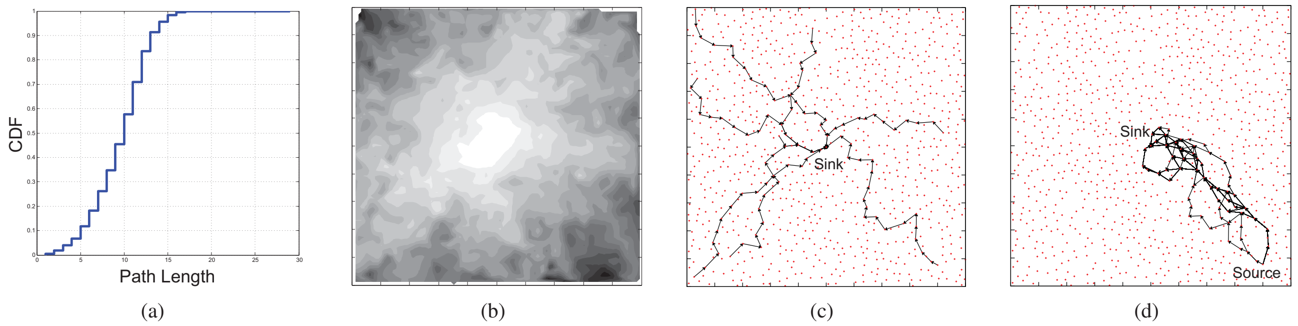


Fig. 11: Some reconstruction results of the 900 nodes simulations. (a) The CDF figure of the lengths of all reconstructed paths. (b) The contour figure of the number of forwarders of all nodes. Nodes in the darker area have more forwarders to help deliver packets. (c) The reconstructed paths of several packets. (d) All reconstructed paths of packets from one same source node.

mainly because of two reasons. First, the wireless dynamics in WSNs are hard to be captured by limited number of probes. Second, using frequent probes may affect the regular data transmission in WSNs.

C. Path Reconstruction in WSNs

MNT [4] and PathZip [10] are two recently proposed approaches for path reconstruction in WSNs. In order to reconstruct each packet's path, MNT first calculates a reliable packet set for each packet. The reliable packet set includes no packet loss, no packet reordering or parent change. Based on this set, the routing path can be safely reconstructed at PC side. PathZip tries to reconstruct the travelled path by compressing the path information into a hash value which is attached to each packet. The sink does an exhaustive search over neighboring nodes for all node along the path for a match with the hash value. When the loss rate is low and the routing dynamics are not severe, MNT and PathZip can reconstruct paths effectively. Different from these approaches, Pathfinder is more robust against packet losses and routing dynamics.

VI. Conclusion

This paper presents Pathfinder, a robust path reconstruction method, for large scale wireless sensor networks with lossy links. By recording the path difference at the node side, Pathfinder reconstructs the routing path of each packet effectively at the PC side by accurate reference packet locating. Further, intelligent path speculation is used to reconstruct more paths. A trace-driven study and extensive simulations are used to verify the effectiveness of Pathfinder. Evaluation results show that Pathfinder significantly outperforms two existing approaches in different network settings.

References

[1] X. Mao, X. Miao, Y. He, T. Zhu, J. Wang, W. Dong, X. Li, and Y. Liu, "Citysee: Urban CO2 monitoring with sensors," in *Proceedings of INFOCOM*, 2012.

[2] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai, "Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest," in *Proceedings of SenSys*, 2009.

[3] J. Zhou, Y. Chen, B. Leong, and B. Feng, "Practical virtual coordinates for large wireless sensor networks," in *Proceedings of ICNP*, 2010.

[4] M. Keller, J. Beutel, and L. Thiele, "How was your journey?: uncovering routing dynamics in deployed sensor networks with multi-hop network tomography," in *Proceedings of SenSys*, 2012.

[5] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of Sensys*, 2003.

[6] Y. Yang, Y. Xu, X. Li, and C. Chen, "A loss inference algorithm for wireless sensor networks to improve data reliability of digital ecosystems." *IEEE Transactions on Industrial Electronics*, vol. 58, no. 6, pp. 2126–2137, 2011.

[7] K. Liu, Q. Ma, X. Zhao, and Y. Liu, "Self-diagnosis for large scale wireless sensor networks," in *Proceedings of INFOCOM*, 2011.

[8] K. Liu, M. Li, Y. Liu, M. Li, Z. Guo, and F. Hong, "Passive diagnosis for wireless sensor networks," in *Proceedings of SenSys*, 2008.

[9] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of SenSys*, 2005.

[10] X. Lu, D. Dong, Y. Liu, X. Liao, and L. Shanshan, "Pathzip: Packet path tracing in wireless sensor networks," in *Proceedings of MASS*, 2012.

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

[12] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of SenSys*, 2009.

[13] M. Keller, L. Thiele, and J. Beutel, "Reconstruction of the correct temporal order of sensor network data," in *Proceedings of IPSN*, 2011.

[14] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *Proceedings of IPSN*, 2005.

[15] O. Chipara, C. Lu, T. C. Bailey, and G. catalin Roman, "Reliable clinical monitoring using wireless sensor networks: Experiences in a step-down hospital unit," in *Proceedings of SenSys*, 2010.

[16] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "Permasense: investigating permafrost with a wsn in the swiss alps," in *Proceedings of EmNets*, 2007.

[17] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Out-of-the-box environmental monitoring," in *Proceedings of IPSN*, 2008.

[18] D. Moss and P. Levis, "BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking," Stanford, Tech. Rep., 2008.

[19] J. Wang, W. Dong, Z. Cao, and Y. Liu, "On the delay performance analysis in a large-scale wireless sensor network," in *Proceedings of RTSS*, 2012.

[20] M. Rabbat and R. Nowak, "Multiple source, multiple destination network tomography," in *Proceedings of INFOCOM*, 2004.

[21] M.-F. Shih and A. Hero, "Hierarchical inference of unicast network topologies based on end-to-end measurements," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 1708–1718, 2007.

[22] M. G. Rabbat, M. J. Coates, and R. D. Nowak, "Multiple-source internet tomography," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2221–2234, 2006.