Path Reconstruction in Dynamic Wireless Sensor Networks Using Compressive Sensing

Zhidan Liu, Zhenjiang Li, *Member, IEEE*, Mo Li, *Member, IEEE*, Wei Xing, and Dongming Lu

***Abstract*—This paper presents CSPR, a compressive-sensing-based approach for path reconstruction in wireless sensor networks. By viewing the whole network as a path representation space, an arbitrary routing path can be represented by a path vector in the space. As path length is usually much smaller than the network size, such path vectors are sparse, i.e., the majority of elements are zeros. By encoding sparse path representation into packets, the path vector (and thus the represented routing path) can be recovered from a small amount of packets using compressive sensing technique. CSPR formalizes the sparse path representation and enables accurate and efficient per-packet path reconstruction. CSPR is invulnerable to network dynamics and lossy links due to its distinct design. A set of optimization techniques is further proposed to improve the design. We evaluate CSPR in both testbed-based experiments and large-scale trace-driven simulations. Evaluation results show that CSPR achieves high path recovery accuracy (i.e., 100% and 96% in experiments and simulations, respectively) and outperforms the state-of-the-art approaches in various network settings.**

***Index Terms*—Bloom filter, compressive sensing, packet path reconstruction, wireless sensor networks.**

I. Introduction

The PER-PACKET routing path serves as the meta-information for understanding detailed wireless sensor networks (WSNs) behaviors in many network maintenance and diagnosis situations, e.g., routing dynamics [38], detection on routing holes [13] or wormholes [9] or even per-hop per-packet transmission delay [15], network diagnosis [23], [28], [33], etc. Reconstructing per-packet routing path information, however, has been known to be nontrivial. WSNs are self-organized and usually deployed in dynamic environments. The underlying network topology constantly changes, and no fixed routing path can be expected for each node. A straightforward solution to reveal the packet path is to record the complete path during packet forwarding, e.g., storing the ID sequence of all relay nodes, in each packet. The introduced overhead linearly grows with the path length, far from scalable.

There have been many efforts made to address the per-packet path reconstruction problem in WSNs. The method that identifies packet paths via hash values triggers disastrous computation overhead [25]. Some methods reconstruct path information by leveraging interpacket correlation in sufficiently stable and reliable networks [16], [20]. However, according to our investigation on the practical packet trace from CitySee [29], a real-deployed and large-scale WSN, we observe nonnegligible topology variation (e.g., up to 28% packets experienced parent changes) and high packet loss (e.g., up to 55% packets lost for some nodes) all the time. Both topology instability and packet loss significantly deteriorate existing path reconstruction methods in practical WSNs. To cope with above issues, we attack the path reconstruction problem from a new perspective, which requires no interpacket correlations and thus makes the solution insensitive to network dynamics and lossy links.

The key insight of our design is as follows. The length of a routing path is usually much smaller than the network size. As a concrete example, the maximum path length reported in CitySee [29] is only 20 hops in comparison to its network size of 1200 nodes. Therefore, we can construct a *path representation space*, the number of whose dimensions equals the total number of nodes in the network. In such a representation space, an arbitrary routing path can be represented by a *path vector*, where each element corresponds to a node in the network. The path vector sets the hop numbers for nodes on the path and zeros for those not involved in the path. As the path length is much smaller than the network size, such path vectors are thus *sparse*, i.e., the majority of elements are zeros. The path reconstruction becomes a problem of unveiling all existing path vectors hidden in the representation space. If all nonzero elements of a path vector can be encoded (with few bytes) into the packets forwarded along the path, we can recover the path vector (and thus the represented routing path) based on a small amount of packets using compressive sensing technique [5], [12].

In this paper, we propose a *C*ompressive-*S*ensing-based *P*ath *R*econstruction method, CSPR, which formalizes the sparse path representation and leverages compressive sensing to recover routing paths. CSPR lets intermediate nodes briefly annotate the transmitted packets and classifies packets traveling along different paths into different groups. For a particular path, the forwarded packets encode independent observations and CSPR performs compressive sensing to recover the path

Manuscript received June 16, 2014; revised January 13, 2015; accepted May 17, 2015; approved by IEEE/ACM Transactions on Networking Editor A. Markopoulou. Date of publication June 09, 2015; date of current version August 16, 2016. This work was supported in part by Singapore MOE AcRF Tier 2 Grant MOE2012-T2-1-070, NTU Nanyang Assistant Professorship (NAP) Grant M4080738.020, NSFC Grant No. 61303233, the National High Technology Research and Development Program of China (863) under Grant No. 2012AA101701, the National Key Research & Development Program under Grant No. 2013BAK01B04, and the Project of Innovative Team of Digital Cultural Media Technology of Zhejiang Province under Grant No. 2010R50040.

Z. Liu is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore, and also with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China. (e-mail: liuzhidan@ntu.edu.sg).

Z. Li and M. Li are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: lzjiang@ntu.edu.sg; limo@ntu.edu.sg).

W. Xing and D. Lu are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: wxing@zju.edu.cn; ldm@zju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2435805

when a certain amount of packets (and the annotations) is collected. The path reconstruction by CSPR requires no interpacket correlations and utilizes only a small number of received packets. CSPR is thus invulnerable to topology dynamics and lossy links. On the protocol level, CSPR introduces only small and fixed overhead in annotating each packet, which could be optimized accordingly for practical WSNs (e.g., 8 B per packet for a network with 245 nodes). In addition to the basic design, we further propose a set of optimization techniques to gradually shrink the representation space and reduce the sparsity of unrecovered path vectors. The numbers of packets needed for remaining path reconstructions are lowered, and processing is thus accelerated. To examine the performance of CSPR, we first evaluate our method using a 29 TelosB mote testbed. The experiment results validate the feasibility and applicability of CSPR in practice. We further conduct extensive and large-scale trace-driven simulations to examine the efficiency and salability of CSPR. Compared to the state-of-the-art methods, CSPR achieves higher path recovery accuracy (i.e., 100% and 96% for experiments and simulations, respectively) with comparable overhead (8 extra bytes per packet).

The rest of this paper is organized as follows. The path reconstruction problem and the motivation of our design are presented in Section II. The design of CSPR is detailed in Section III. In Section IV, we evaluate CSPR through testbed experiments and trace-driven simulations. We review related works in Section V. Section VI concludes this paper.

II. Problem Statement and Motivation

A. Path Reconstruction Problem

A WSN consists of a number of sensor nodes and a sink. All sensor nodes generate and relay packets to the sink. To collect packets, the sink builds a routing structure, e.g., data collection tree [17], [22], information potential [19], [21], etc., covering all the sensor nodes in the network. After generating one packet, the source node forwards it to its parent node, and the packet will be further forwarded until it reaches the sink. Due to the network dynamics, each node will periodically search for the best parent node. The routing path from each node to the sink thus may change. At the sink, a path reconstruction method is desired to recover the path each packet traveled. One packet path is an ID sequence from the source of the packet to the sink, including IDs of all intermediate nodes relaying this packet and their hop numbers as well.

There have been many efforts made to address the path reconstruction problem (as reviewed in Section V). Two state-of-the-art methods, MNT [20] and Pathfinder [16], have been recently proposed. MNT [20] reconstructs per-packet path by exploiting *interpacket correlation*, i.e., a relayed packet and its adjacent packets locally generated at any node i are usually forwarded to the same next hop. Such local packets serve as *anchors* of the relayed packet at node i. As the first-hop receiver is recorded in packets, the path of a packet can be obtained by concatenating the first-hop receivers of all its anchors. Improving on MNT, Pathfinder [16] tolerates certain inconsistence in interpacket correlation via explicitly recording inconsistence in packets. The reconstruction failure occurs once the inconsistence exceeds the tolerance capacity. To accurately locate anchors, Pathfinder further imposes the packet generation rate of each node to be identical and fixed. Both MNT and Pathfinder require stable network topology such that interpacket correlation can be captured. The practical WSNs, however, behave dynamically and the wireless links are far from stable [11], [18] (as we will demonstrate in Section II-B). Both network dynamic and packet loss have strong impacts on the anchor identifications, and thus deteriorate the performances of MNT and Pathfinder.

B. How Packet Routing Behaves in Practical WSNs

We investigate the packet trace from a real-deployed and large-scale WSN CitySee [29], and discuss how practical routing behaviors impact the path reconstruction performances of the state-of-the-art methods as well. The CitySee, deployed in Wuxi city, China, contains more than 1200 sensor nodes for monitoring urban environmental factors, including carbon dioxide, temperature, humidity, light, etc. Sensor nodes generate data every 10 min, encapsulate data into a single packet, and transmit packets to the sink with CTP [17] in a multihop manner. Each packet possesses a default CTP packet header including following common fields: *source address*, ID of the node generating this packet; *sequence number*, order of the packet generated from the source; *first-hop receiver*, parent node ID of the packet's source; and *hop count*, length of path traveled by the packet. Each packet further records the first 10-hop node IDs for future analysis. The packet trace from a subnetwork of 245 nodes with a collection period of one week is available for our study. Such partial network covers about 16 km^2 area with the average and maximum routing path length as 7 hops and 12 hops, respectively.

With CitySee packet trace, we examine the *topology change* and *packet loss*, which are most relevant to the stability of WSNs [6]. For each node, one topology change is indicated by the first-hop receiver (i.e., parent) difference between two consecutively generated local packets. The topology change rate of a network is defined as the average of ratios $\frac{\text{\# of pkts with topo. changes}}{\text{total \# of pkts received}} \times 100\%$ for each node within certain time duration, where the numerator indicates the total parent changes of each node in the time window. Similarly, for each node, one packet loss is indicated by a sequence number missing. The packet loss rate of a network is further defined as the average of ratios $(1 - \frac{\text{\# of pkts received}}{\text{\# of pkts expected to receive}}) \times 100\%$ for each node within certain time duration.

Topology change rate indicates the topology stability of a network, which reflects the validity of the interpacket correlation assumption made in existing path reconstruction methods [16], [20]. Fig. 1(a) depicts the topology change rate in CitySee with 2-h time window of measurement. From the figure, we observe that the average and maximum topology change rates at all time windows are about 12% and 48%, respectively. In Fig. 2, we summarize the CDF of topology change rates of all nodes during one week. For most nodes, the rate is as high as 10% and the maximum rate reaches up to 28%. Due to the instable topology, each node would transmit packets to the sink via different paths. In Fig. 3(a), we plot the number of routing paths formed in the packet trace as time goes by. The average number of paths starting from a node persistently increases and finally on average each node has 76 paths. All this evidence demonstrate that network topology suffers from severe instability and the interpacket correlation may not be well validated in practice. In Section IV, we have

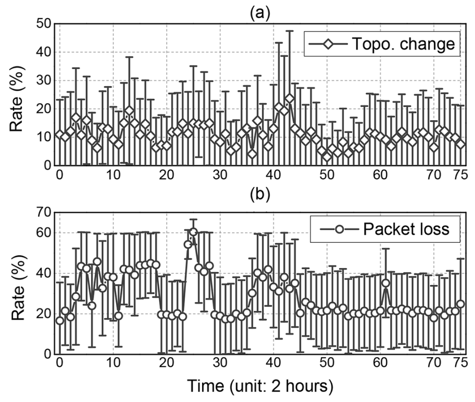


Fig. 1. Average (a) topology change rate and (b) packet loss rate of all nodes at each time window.

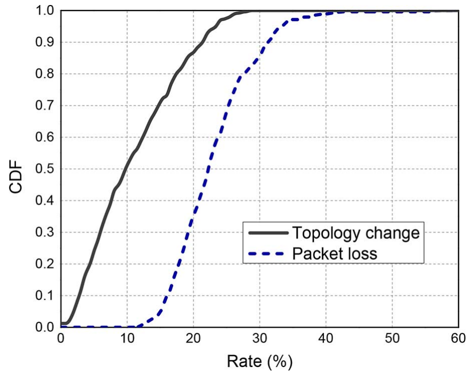


Fig. 2. CDF of topology change rates and packet loss rates of all nodes in the CitySee packet trace.

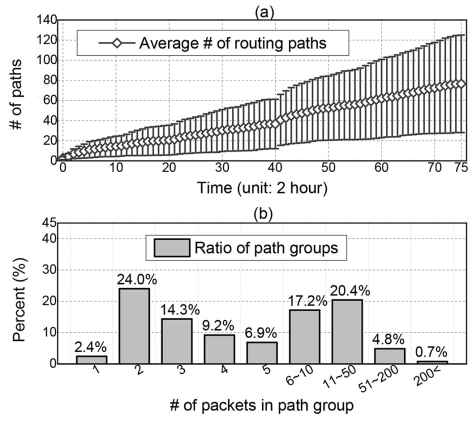
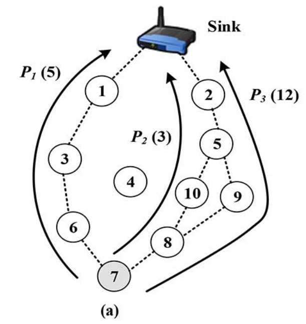


Fig. 3. (a) Average number of paths for each node. (b) Distribution of packet volumes for all groups.

implemented both MNT and Pathfinder. By analyzing their detailed executions, we find that the topology dynamics could solely cause about 10% and 1% of anchor misidentifications, which directly lead to path reconstruction failures. Pathfinder outperforms MNT at the cost of explicitly recording certain topology changes in each packet.

Packet loss rate reflects the reliability of packet receptions. Packet losses will hide the interpacket correlation, e.g., anchor availability, and deteriorate performances of methods relying on such correlation. Fig. 1(b) depicts the packet loss rate in the packet trace, with 2-h time window. From the figure, we see that the packet loss rate changes rapidly with the average ranging from 17% to 69%. In Fig. 2, we summarize the CDF of packet loss rates of all nodes during one week. The distribution mainly



$\mathbb{N}$	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$\mathbf{s}_1$	3	0	2	0	0	1	0	0	0	0
$\mathbf{s}_2$	0	4	0	0	3	0	0	1	0	2
$\mathbf{s}_3$	0	4	0	0	3	0	0	1	2	0

(b)

Fig. 4. (a) Wireless sensor network model. Source node 7 transmits 20 packets to the sink via 3 different routing paths, e.g., 5 packets on path P_1. (b) Sparse representations of the three routing paths in the whole network space.

concentrates in the range between 15% and 35%, with the maximum rate as high as 55%. By analyzing the detailed executions of MNT and Pathfinder, we find that with such a high packet loss rate, about 49% and 35% packets fail to identify anchors, which directly result in path reconstruction failure for those packets.

With the joint impacts of topology dynamics and packet losses, we find that in the large-scale and dynamic WSN CitySee, 59% and 36% packet paths might not be successfully reconstructed by the state-of-the-art methods MNT and Pathfinder, respectively. This motivates us to explore a solution insensitive to both network dynamics and lossy links.

C. Path Reconstruction From Sparse Path Representation

Sparse Representation of Routing Paths: Since sensory data in WSNs are usually collected with a direct acyclic graph (DAG) routing structure (e.g., data collection tree), the path length is thus in the order of $O(\log(N))$, where N is the total number of nodes in the network. According to the statistical results of all paths formed in CitySee packet trace, the path length ranges between 2 and 12 hops, which are much smaller than the network size 245. We can construct a path representation space $\mathbb{N}$. The dimensionality of $\mathbb{N}$ equals the total number of nodes in the network, and each dimension corresponds to one node. In such a representation space, any routing path can be presented by a path vector. According to whether a node is involved in the routing path, the path vector sets either hop number or zero for its corresponding element. As the path length is much smaller than the network size, the path vector is thus sparse, i.e., the majority of elements in the vector are zeros.

Fig. 4 illustrates the sparse path representation with a network containing 10 nodes. We construct the path representation space $\mathbb{N} = [n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}]^T$ including all nodes. An arbitrary in-network path can be represented by a path vector in $\mathbb{N}$. For example, path P_1 is represented as $\mathbf{s_1} = [3, 0, 2, 0, 0, 1, 0, 0, 0, 0]^T$. Node 7 is the source node and could be known from the packet header. Hence, the element n_7 is not considered in $\mathbf{s_1}$ and thus assigned as 0. As nodes 6, 3, and 1 serve as the first, second, and third hop relay, respectively, in path P_1, thus $n_6 = 1, n_3 = 2$ and $n_1 = 3$. All other elements irrelevant to P_1 are zeros in $\mathbf{s_1}$. The sparsity of this path vector is 3 (out of 10). Other path vectors, e.g., $\mathbf{s_2}$ and $\mathbf{s_3}$, can be similarly constructed.

Compressive-Sensing-Based Path Reconstruction: Based on sparse path representation, the path reconstruction thus becomes a problem of unveiling all existing path vectors hidden in the path representation space $\mathbb{N}$. Packets from the same source may travel different paths to the sink, while the paths implicitly classify packets into different *path groups*, i.e., all packets in one group travel exactly the same path. After the paths of all path groups get reconstructed, the path of each packet is obtained as well.

For the path vector of a routing path, if its nonzero elements can be encoded into each packet forwarded along the path, it is viable to recover the vector (and thus the represented path) based on a small amount of packets using compressive sensing [5], [12]. In particular, for any path vector $\mathbf{s}$ in space $\mathbb{N}$ with sparsity k, where $k \ll N = |\mathbb{N}|$, the compressive sensing theory states that M, instead of N, independent equations are sufficient to solve the N unknowns in $\mathbf{s}$, where $M \ll N$. The M independent equations can be acquired by projecting $\mathbf{s}$ to a measurement matrix $\Phi : Y = \Phi\mathbf{s}$, where Y is an M-dimension vector and Φ is an $M \times N$ matrix. If Φ satisfies the *Restricted Isometry Property* [4], $\mathbf{s}$ can be exactly recovered by solving following l_1-minimization problem:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in R^N} \|\mathbf{s}\|_{l_1} \quad \text{s.t. } Y = \Phi\mathbf{s}$$

when $M \geq ck\log(\frac{N}{k})$, where c is a small positive constant [4].

The above compressive-sensing-based approach makes path reconstruction for each path group independent, and can recover the path for a group of packets once sufficient packet are accumulated. As a result, this approach requires no inter-packet correlation, which makes itself inherently invulnerable to network dynamics and lossy links. On the other hand, once a path is recovered, the path for all future packets residing in the same group becomes immediately available, which avoids repeatedly triggering path reconstruction for each received packet and largely reduces the computation overhead.

Design Challenges: Translating the idea of compressive-sensing-based per-packet path reconstruction to a practical system, however, encounters a set of challenges as follows.

1) Accurate Packet Classification: Packet classification into each path group must be accurate. Fig. 3(a) plots the average number of paths (each path one path group) formed in the packet trace as more and more packets are received by the sink. According to the statistics, we observe significant differences existing in the number of paths for each node, e.g., the maximum number of paths of some node could even reach 199. It is vital to distinguish each individual path and classify the packets into the right group. Packet misclassification will lead to path reconstruction error for one group. Therefore, packet classification must have high accuracy.

2) Lightweight Per-Packet Annotation: To enable the compressive-sensing-based path reconstruction, packets need to carry encoded information for all nonzero elements of the path vectors representing their traveling paths. In particular, when one packet is relayed by an intermediate node, the node needs to annotate its hop number along the path into the packet header. As a result, packet header is updated hop by hop. The annotation overhead must be small. In particular, the annotation field in the packet header should be small in size and remain constant (not increase with the path length). In addition, updating should be performed in a distributed manner without introducing any centralized control.

3) Short Per-Packet Path Recovery Delay: From Fig. 3(b), we observe that the packet volumes for all path groups within one week are highly heterogenous. About 50% path groups include fewer than 5 packets, while some path groups (about 5.5%) contain more than 50 packets. To perform compressive-sensing-based path reconstruction for one path group, the number of packets (one annotation as one measurement for each packet) accumulated in this group should be at least M to ensure a good recovery quality. Some path groups, however, cannot accumulate sufficient packets even after a long time. As practical requirement, per-packet path recovery delay should not be excessively long.

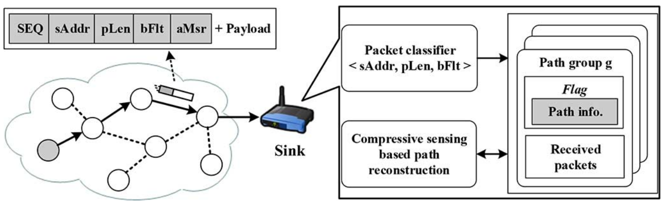


Fig. 5. System architecture of CSPR.

III. Design of CSPR

CSPR consists of two parts, the in-network part for path information encoding and the server part for per-packet path reconstruction. The system architecture of CSPR is depicted in Fig. 5. We will present the system overview first and then detail each component.

A. CSPR Overview

Several fields in the packet header are used by CSPR to carry packet information, as depicted by Fig. 5. SEQ is the packet sequence number. sAddr is source address of the packet. pLen records the path length. bFlt is a Bloom filter to space-efficiently record the IDs and corresponding hop count information of all relay nodes. aMsr stores the encoded measurement along the path. All the five fields are initialized at the source node. In particular, SEQ and sAddr keep unchanged after initialization, whereas pLen, bFlt, and aMsr are updated at each intermediate hop. Note that SEQ, sAddr, and pLen can be found in the default packet header, e.g., CTP packet header, and only two fields, bFlt and aMsr, are additionally introduced by CSPR. The extra overhead to each packet is thus slight, e.g., 6 B of bFlt and 2 B of aMsr for a network with 245 nodes. We detail the in-network updating of packet header in Section III-B.

CSPR adopts a 3-tuple key, $\langle$sAddr, pLen, bFlt$\rangle$, to identify the path of a packet. For all received packets, CSPR first distinguishes their paths according to sAddr, and then differentiates those from the same source based on pLen. Finally, bFlt is used to distinguish paths owning the same sAddr and pLen. If two packets have the same 3-tuple key, they are considered to travel the same path and will be classified into the same path group. At the sink, CSPR maintains a database, where each entry is indexed via the 3-tuple key and corresponds to a unique path group. When a packet is received, CSPR extracts the 3-tuple key from the packet header and looks for a matched entry in the database. If the matched entry has already recovered the path, the path for the packet becomes immediately available. If an entry is matched yet the path is not ready, CSPR launches path reconstruction when sufficient packets are accumulated. If no entry matched, CSPR creates an entry for the new path group indexed by the 3-tuple key of the packet. We detail the compressive-sensing-based path reconstruction component in Section III-C.

As improvements on the basic design, a set of optimization techniques is proposed to gradually shrink the path representation space and reduce the sparsity of unrecovered path vectors. The number of packets needed by compressive sensing is accordingly lowered such that the remaining path reconstructions are accelerated. In addition, CSPR can launch a remedy scheme if some path groups fail to recover their paths after an excessive long delay. We detail those components in Section III-D.

B. In-Network Path Information Encoding

In this section, we introduce the in-network updating of the last three fields pLen, bFlt, and aMsr in turn.

Updating of pLen*:* The pLen field of each packet is initialized to 0 by the source and increased by one at each intermediate hop along the path. At each intermediate hop, pLen is updated prior to both bFlt and aMsr as the updating of the latter two fields relies on the new pLen value. When a packet arrives at the sink, we can know the packet path length through the bFlt field, while we cannot infer the hop count information for each intermediate node along the path relying on this filed at the server side.

Updating of bFlt*:* Bloom filter is an L-bit array associated with H independent hash functions, where L and H are two parameters to be determined (detailed in Section III-C). The bFlt field of each packet accommodates an L-bit array, and sensor nodes use the same set of H independent hash functions $f_i(\cdot), i = 1, 2, \ldots, H$, to update bFlt. Different arrays represent different path information. Initially, all L bits in the bFlt field of a packet header are set to 0. At each intermediate hop, the node compresses its existence into the bFlt field as follows. H hash values $v_i = f_i(d \times h) \in \{0, 1, \ldots, L-1\}, i = 1, \ldots, H$, are first obtained by feeding the product of node ID d and hop count h in the pLen field to the H hash functions. Then, the v_i bits of the bFlt field are set to 1 by that relay node, $i = 1, 2, \ldots, H$.

Updating of aMsr*:* The aMsr field in each packet is also initialized as 0 by the source and updated along the path. At each intermediate hop, the node encodes its hop number along the path in aMsr. In particular, the node multiplies the updated pLen value with a random coefficient and adds the product with current aMsr value. We design such a field for the purpose of recovering sparse path vector via compressive sensing technique.

The reason that CSPR adopts both aMsr and bFlt to encode the path information is because: if CSPR only uses aMsr, it cannot group packets from the same routing path, and if CSPR encodes the path information with bFlt merely (even with a larger size), it needs the topology information to enumerate all possible intermediate nodes, which results in enormous computation overhead and plenty of false positive results. By combining aMsr and bFlt, CSPR can accurately classify packets and reconstruct their corresponding paths.

When CSPR later recovers the path for a set of received packets in one path group, it solves equation $Y = \Phi \mathbf{s}$ to obtain $\mathbf{s}$ using compressive sensing technique. The path vector $\mathbf{s}$ is a column vector with N elements. Each element represents one node in the path representation space $\mathbb{N}$. If a node is included in the path represented by $\mathbf{s}$, the corresponding element indicates its hop number; otherwise, element is zero. In the equation, each element in Y is the final aMsr value of one packet and the corresponding row in Φ can be represented by $\phi = [\alpha_1, \alpha_2, \ldots, \alpha_N]$, where α_j means that if node j relays the packet, α_j will be the coefficient multiplied with pLen value, $j = 1, 2, \ldots, N$. The product of Φ and $\mathbf{s}$ essentially replays the updating process of aMsr along the path.

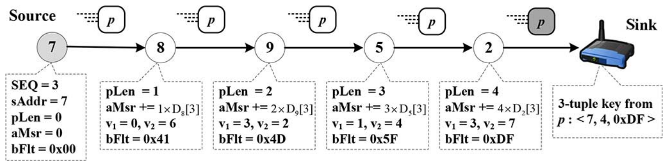


Fig. 6. Path information encoding process for a packet p on path P_3 in Fig. 4. CSPR finally can extract a 3-tuple key from packet p at the sink.

Before the reconstruction of $\mathbf{s}$, we are not aware of which α_j finally participates in the aMsr updating. We have to provide complete α_j in ϕ for the compressive sensing recovery. To avoid explicitly acquiring such information from each node in the network, we introduce a dictionary-based strategy working in a fully distributed manner. Each node i is configured with a coefficient dictionary $\mathcal{D}_i$, stored in the RAM. To update the aMsr field for a packet, node i multiplies the updated pLen value with the nth coefficient in $\mathcal{D}_i$, where $n = (\mathsf{SEQ}\,mod|\mathcal{D}_i|)$ and $|\mathcal{D}_i|$ is the size of $\mathcal{D}_i$. Each $\mathcal{D}_i$ is made up by Gaussian random numbers, and the CSPR server is aware of the dictionaries of all nodes in the network. Coefficients in all dictionaries are generated independently. As a result, different nodes have different dictionary elements. In our current design, each coefficient occupies 2 B and each dictionary contains 100 independent coefficients. The 200-B storage overhead accounts for only 2% storage occupancy to commercial sensor motes, e.g., TelosB with 10 kB RAM [1]. We could synchronize the generation of random numbers between nodes and server via global seed [26], while at the cost of reducing coefficient randomness.

Fig. 6 illustrates the path information encoding for a packet p on path P_3 in Fig. 4. For ease of illustration, we simply set $L = 8$ and $H = 2$ for the Bloom filter. Source node 7 generates packet p with SEQ equaling 3 and initializes sAddr, pLen, bFlt, and aMsr to 7, 0, 0x00, and 0, respectively. At each hop, pLen, aMsr and bFlt are updated. For instance, at node 5, pLen is increased from 2 to 3. Two hash values are $v_1 = 1$ and $v_2 = 4 (v_1, v_2 \in \{0, 1, \ldots, 7\})$. The bFlt is thus updated by setting the first and fourth bits to 1. The aMsr is updated by adding the product of the updated pLen (i.e., 3) and the third coefficient in $\mathcal{D}_5$ to current aMsr value.

C. Compressive-Sensing-Based Path Reconstruction

In this section, we first present the packet classification mechanism in CSPR, and then detail the compressive-sensing-based path reconstruction with path verification scheme to ensure the reconstruction correctness.

Packet Classification: For each received packet, CSPR extracts the 3-tuple key, $\langle$sAddr, pLen, bFlt$\rangle$, from the packet header and then classifies it into a path group. A path group is designed to contain packets traveling the same path. At the sink, CSPR manages all path groups with a database. One database entry is indexed via the 3-tuple key and corresponds to a unique path group. Each entry is further allocated with a piece of buffer to accommodate the packets belonging to this group. An entry also has an indicator *Flag* to tell whether the path gets recovered. When a packet is received by sink, CSPR

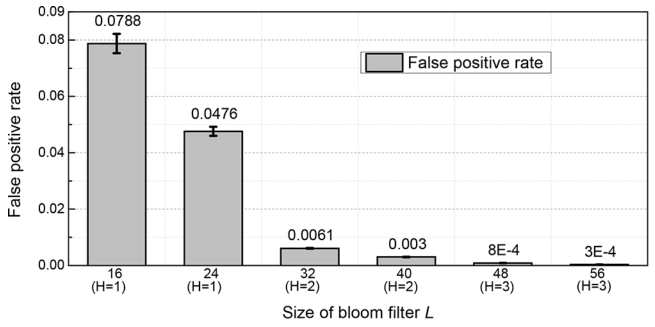


Fig. 7. False positive rates under various combination setting of L and H.

extracts the key from the packet header and looks for a matched entry. If a matched entry exists, the packet is inserted into the allocated buffer. Furthermore, in case the indicator *Flag* is *true*, CSPR checks whether the recovered path of this group is valid for the packet through the path verification component (as described later). When the *Flag* is *false*, however, CSPR will recover the path if the amount of packets accumulated in the group is sufficient (the number will be given later). On the other hand, if no entry matched, CSPR creates an entry for this new path group, initializes *Flag* as *false*, and inserts the packet into the buffer.

The packet classification using the 3-tuple key might lead to misclassification since the comparison result between two Bloom filers could be false positive, i.e., different paths possess the same L-bit array. However, with a proper parameters L and H settings of the Bloom filter, misclassification rate could be low. The setting of L trades off between packet overhead and classification accuracy. A lager L leads to a lower false positive probability yet more overhead to each packet. According to Bloom filter theories [3], [36], given the false positive rate p to tolerate and the routing path length k, the optimal Bloom filter size L will be about $O(-\frac{k\ln(p)}{(\ln 2)^2})$, with the corresponding number of hash functions H being $\lfloor \frac{L}{k}\ln 2 \rfloor$. Therefore, the key of the Bloom filter configuration is to set the parameter k. The optimal setting can achieve the best tradeoff between the overhead and the false positive rate. In principle, we prefer to set k as the maximum routing path length, which can achieve a low false positive rate. For the CitySee packet trace, the maximum path length (excluding the source and destination) observed in the trace is 10. Fig. 7 depicts the false positive rate when L varies from 16 to 56, where we set $k = 10$ to determine H, for the CitySee packet trace. From the figure, we see that the false positive rate is generally not high, i.e., <0.09. In particular, when L is larger than 40, the false positive rate is smaller than 0.001. Although the above setting can achieve a low false positive rate using a small number of hash functions, the maximum path length information is usually unknown in a practical system before the deployment. To address this issue, in CSPR, we propose to set k as $\lceil \log(N) \rceil$, where N is the total number of nodes in the network. The network size N usually can be determined before the deployment. We further test the performance of this practical approach. Since the CitySee packet trace contains 245 nodes in total, we set k as $\lceil \log(245) \rceil \approx 8$, and thus the optimal H is $\lfloor \frac{L}{8}\ln 2 \rfloor$. From the result, we find that the false positive rates are small in general as well. The packet misclassification cases can be negligible as long as we configure the Bloom filter carefully. With a balance between the classification accuracy and packet overhead, we configure Bloom filter using $k = \lceil \log(N) \rceil$ and $p = 0.15\%$ in CSPR, e.g., $L = 48$ for the CitySee trace.

As long as sufficient packets, truly belonging to the same path group, are received, our path reconstruction method can still recover the path even if certain misclassified packets are mixed in the reconstruction. Moreover, after a routing path is recovered, a path verification component is used to further verify the reconstruction correctness, which can also eliminate all misclassified packets from the group as well. As a result, CSPR is not vulnerable to packet misclassification, while accurate classification is still preferred as high accuracy ensures that more paths could be recovered with shorter latency.

Path Reconstruction: Based on the encoded measurements in received packets, CSPR recovers the path for a path group using compressive sensing technique. Concretely, for one path group, CSPR solves $Y = \Phi \mathsf{s}$ to obtain the path vector s for path recovery. Each element in Y, denoted as y_i, is the aMsr value of a received packet i. The corresponding row in Φ is represented by $\phi_i = [\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,N}]$, where $\alpha_{i,j}$ is the $(\mathsf{SEQ}_i mod|\mathcal{D}_j|)$th coefficient in the coefficient dictionary $\mathcal{D}_j$ of node j and SEQ_i is the sequence number of packet i. If the elements in s are known, the product of ϕ_i and s replays the updating process of the aMsr field along the path of packet i. Therefore, $y_i = \phi_i \mathsf{s}$. For the path reconstruction problem, both y_i and ϕ_i are known while the N elements in s are unknowns. In principle, N independent equations are needed to obtain s, and each equation $y_i = \phi_i \mathsf{s}$ corresponds to one received packet.

Since path vector s is sparse, it can be recovered using M rather than N equations by leveraging compressive sensing. Therefore, as long as M packets are accumulated by one path group, the vector s can be recovered. We use the aMsr values from M packets to form an M-dimension column vector Y, i.e., $Y = [y_1, y_2, \ldots, y_M]^T$. For each packet i out of M packets, we further use its sequence number SEQ_i to select coefficients from coefficient dictionaries of all nodes in the path representation space $\mathbb{N}$ to construct $\phi_i = [\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,N}]$. All ϕ_i together form an $M \times N$ matrix

$$\Phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_M \end{bmatrix} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,N} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M,1} & \alpha_{M,2} & \cdots & \alpha_{M,N} \end{bmatrix}. \quad (1)$$

Since all coefficients in Φ follow the Gaussian distribution, the measurement matrix Φ satisfies the RIP condition [26]. In addition, if the number of packets in the path group is sufficient, at least $ck\log(\frac{N}{k})$, we can apply the compressive sensing technique to recover the path vector s. A variety of compressive sensing solvers can be used to obtain s. However, most of them provide no confidence for the reconstruction quality. As a result, s might not be correctly reconstructed. In CSPR, since the sparsity of s, i.e., path length indicated by the pLen field, is known prior to the path reconstruction, we adopt a more advanced solver CoSaMP [31] that requires the vector sparsity as input. If CoSaMP outputs a recovered result, the result is correct with a high probability. Otherwise, CoSaMP outputs "Fail" instead. Furthermore, CoSaMP can tolerate certain noises mixed in Y, and the path vector can be recovered even if some misclassified packets are included. In particular, CoSaMP could recover

s if the number of correctly classified packets is greater than $M = ck\log(\frac{N}{k})$ when $c = 1.5$ according to recent study [5].

In CSPR, after a packet is received, if the path group this packet belongs to has not recovered its path yet, the server checks whether the number of packets in the group exceeds M. The path reconstruction will be performed if the threshold is reached. If CoSaMP returns "Fail," more packets are expected. If CoSaMP returns a valid result, CSPR executes path verification component to further ensure its correctness.

Path Verification: Given a recovered path and a packet, the path verification component verifies whether the recovered path is valid for the packet via path vector s of the recovered path and aMsr value of the packet. More precisely, for a packet with sequence number SEQ, we calculate the product of ϕ and s, where $\phi = [\alpha_1, \alpha_2, \ldots, \alpha_N]$ and α_j in ϕ is the $(\mathsf{SEQ}mod|\mathcal{D}_j|)$th coefficient in $\mathcal{D}_j, j = 1, 2, \ldots, N$. As coefficients are randomly Gaussian, it is highly improbable for two packets traveling two different paths yet leading to the same aMsr value. Therefore, if $\phi \cdot \mathsf{s}$ is equal to the aMsr value of the packet, the recovered path is valid for this packet.

When a path p is newly reconstructed, this component is executed to verify the correctness of path p for a path group. It is also used to eliminate all misclassified packets due to Bloom filter collision. If path p is only valid for the minority of packets in the group (e.g., $\leq$ 20%), the path recovery is considered to be failed, e.g., an incorrect output from CoSaMP. The path reconstruction will be performed after more packets are received. Otherwise, path p is viewed as the correct path, and the corresponding *Flag* of this group is changed to be *true*. However, we exclude all packets failed in the aMsr check from current group and form a new group, with *Flag* as *false*, for those packets. Since the new group has the same 3-tuple key with current group, we introduce gIdx as the secondary key, an auto-increment key, to further distinguish entries in database with the same 3-tuple key.

The recovered paths will benefit all future packets traveling on them. When a group with recovered path receives a packet, CSPR just simply invokes the path verification component to check whether the packet truly traveled the recovered path. If yes, this packet obtains its path immediately. In CSPR, for a given 3-tuple key, path groups are matched with the packet following the ascending order of gIdx keys. At worst, the packet might be assigned to a newly formed path group. Thanks to identifying each individual path, a large number of packets could have their path with no recovery delay and meanwhile CSPR avoids huge computation overhead, which will be demonstrated in our evaluations.

D. Optimization

In this section, we propose a set of optimization techniques to improve the performance of our basic design.

Reduction of Path Representation Space: This optimization aims to reduce the number of elements in a representation space by continuously monitoring the network topology. The minimum number of packets required to recover a length of k path is $ck\log(\frac{N}{k})$, which is proportional to the space size N. The basic design utilizes all N nodes in the network to form the space. This component tries to reduce the space size for each path group and thus reduces its needed packets for path reconstruction. For any node i, CSPR maintains a first-hop receiver set $\mathbb{S}_i$, which is learned from received packets (via the *first-hop receiver* field in packet header) and recovered paths (the next hop of node i along a path). All elements in $\mathbb{S}_i$ have ever received packets from node i. The reduced representation spaces for all groups with the sAddr attribute as node i are the same, denoted as $\overline{\mathbb{N}}_i$. Elements of $\overline{\mathbb{N}}_i$ are added in an iterative manner: 1) elements in $\mathbb{S}_i$ belong to $\overline{\mathbb{N}}_i$; 2) elements in $\mathbb{S}_j$, where $j \in \overline{\mathbb{N}}_i$, belong to $\overline{\mathbb{N}}_i$.

We take node 9 in Fig. 4 as an example to illustrate the formation of $\overline{\mathbb{N}}_9$. The first-hop receiver set $\mathbb{S}_9$ of node 9 contains 5 and 8. By iteratively including all first-hop receivers of nodes in $\overline{\mathbb{N}}_9$, $\overline{\mathbb{N}}_9$ finally contains 2, 5, 8, and 10. Compared to original space $\mathbb{N}$, the size of $\overline{\mathbb{N}}_9$ is reduced from 10 to 4. From our investigation on the CitySee packet trace, we find the average and maximum reduced representation space sizes are only 39 and 104 respectively, which are much smaller than the total number of nodes 245 in the network.

Even with sufficient packets, path reconstruction based on the reduced representation space might be unsuccessful unless all intermediate nodes of the path are included in the reduced space. As it is hard for CSPR to explicitly determine whether all intermediate nodes are included, we thus use the reduced representation space as a backup scheme. For each path group, CSPR always use the original N-dimension representation space for the reconstruction. If the reconstruction fails, CSPR recovers one more time using the reduced space. If the second reconstruction succeeds, CSPR will launch the path verification component to verify the correctness of the recovered path; Otherwise, CSPR waits for more packets and performs the next round reconstruction, still starting from the N-dimension representation space.

Reduction of Path Vector Sparsity: This optimization tries to reduce the sparsity of unknown path vectors by inferring from recovered path vectors. The intuition behind it is that if we know the existence of some intermediate nodes prior to path reconstruction, we can treat their elements in the path vector as zeros such that reduces the vector sparsity. As a result, fewer packets are needed to reconstruct the path.

After a path p gets recovered, CSPR may potentially reduce the path vector sparsity of another path q. To control the computation overhead, in current CSPR, path vector sparsity reduction is only applied for a path q having the same source and path length as p. For an intermediate node i at the hth hop along path p, H hash values $v_j = f(i \times h) \in \{0, 1, \ldots, L - 1\}, j = 1, 2, \ldots, H$, are obtained by feeding the product of $i \times h$ to the H hash functions. If all v_j bits in the Bloom filter bFlt of path q are 1's, we denote the input $i \times h$ is contained by bFlt of path q. Node i is thus considered to be included in path q as well at the same hop h with a high probability. Therefore, CSPR can opportunistically omit the element for node i in the path vector of path q and the vector sparsity is reduced by one. By excluding all intermediate nodes (of path p) contained in path q's Bloom filter, the original formulation transforms to a new version $Y' = \Phi'\mathsf{s}'$. In particular, by removing the annotations from nodes passed the bFlt testing of path q, each element in Y' only preserves the summation of node ID and coefficient products for the remaining nodes in path q. Similarly, the recovered s' only contains nonzero values for those preserved nodes. A reduced space $\overline{\mathbb{N}}$ only including nodes for the remaining hops can be constructed as well. Finally, the number of needed packets for the path reconstruction is largely reduced.

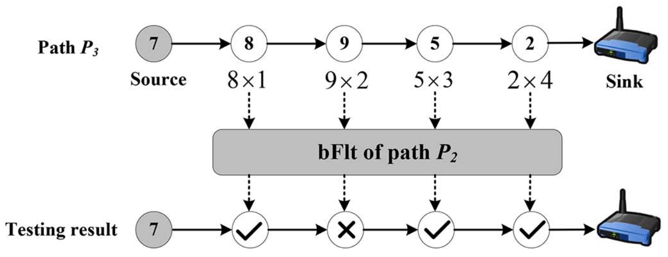


Fig. 8. Example for reduction of path vector sparsity.

For example, in Fig. 8, CSPR has recovered path P_3 in Fig. 4 and tries to reduce the path vector sparsity of P_2. Among four intermediate nodes (8, 9, 5, and 2) in P_3, the nodes 8, 5, and 2 are tested to be contained by bFlt of P_2. The path vector sparsity of P_2 is thus reduced to 1. The products of their node ID and coefficients are deducted from Y for P_2 to form Y', in which only the product of the node ID and the coefficient for the second hop node in P_2 remains. The reduced space $\overline{\mathbb{N}}$ for the new equation is constructed by candidate nodes at that hop, i.e., first-hop receivers of 8, $\mathbb{S}_8$.

Path vector sparsity reduction is launched when a new path is correctly reconstructed and there exists unrecovered paths having the same source and path length as this one. Similar to packet misclassification, a node might be false positively removed from the path vector in the Bloom filter testing phase. In this case, the inferred path is incorrect and cannot pass the path verification. Fortunately, with a proper Bloom filter setting, false positive scenario is rare in CSPR.

Heuristic Path Scanning: For those path groups with insufficient accumulated packets even after a long time, this component is designed to scan possible paths for them based on the learned network topology. It is triggered when the path reconstruction deadline of a path group is approaching. As this scheme is relatively computation intensive, it recovers path not only for the group that triggers its execution, but also for other unrecovered groups whose paths have the same source, sAddr, as this group at the same time.

Starting from the source node i of the path group that triggers heuristic path scanning, CSPR builds a directed graph covering all nodes in the reduced representation space $\overline{\mathbb{N}}_i$. For any two nodes a and b in $\overline{\mathbb{N}}_i$, if node b is in the first-hop receiver set of node a, i.e., $b \in \mathbb{S}_a$, CSPR adds an arrow from a to b. This component enumerates all possible paths from source node i to the sink in the directed graph. For each enumerated path, CSPR treats it as a newly reconstructed path and applies the path verification procedure to check whether it is a valid path for some unrecovered group whose path is originated from node i. As a result, heuristic path scanning can recover paths for multiple path groups.

IV. Performance Evaluation

In this section, we evaluate the performance of CSPR with comparisons to two state-of-the-art approaches based on a 29 TelosB mote testbed and the CitySee packet trace.

A. Evaluation Setup

Compared Approaches: We compare CSPR to the two state-of-the-art approaches, Pathfinder [16] and MNT [20]. For fairness, we implement the path speculation in Pathfinder [16] as a remedy method for both Pathfinder and MNT to improve their performances when anchors are lost. We add 1-B XOR field into packet header for MNT as checksum, similar with Pathfinder, to verify the reconstructed paths. For all the three approaches, we define the same path recovery delay bound δ to meet practical requirements. In general, the three approaches will reconstruct packet paths following their core principles (referred to as *core method*) and employ the remedy methods (heuristic path scanning for CSPR and path speculation for Pathfinder and MNT, referred to as *remedy method*) to reconstruct those failed packets when path recovery delay bound is approaching. All approaches are running on a desktop PC with dual-core 3.16 GHz CPU and 4 GB RAM. The execution of remedy methods have a time limit of 1 s to avoid excessive computation overhead.

Performance Metrics: The *recovery accuracy* for each node is calculated as $\frac{\text{\# of correct recovered pkts}}{\text{total \# of pkts received}} \times 100\%$. Similarly, the recovery *false positive* for each node is computed as $\frac{\text{\# of false positive recovered pkts}}{\text{total \# of pkts received}} \times 100\%$. The path *recovery delay* for packet i is defined as sequence number offset $\mathsf{SEQ}_j - \mathsf{SEQ}_i$ (with no reordered packets). The path of packet i is successfully recovered after packet j, generated by the same source as packet i, is received. We set the bound $\delta = 50$, which means a remedy method will be employed to search possible paths when another 50 packets of the same source are received. Small δ will reduce the overall recovery delay but trigger much more computation overhead. Intuitively, the approach with higher recovery accuracy yet lower false positives and smaller path recovery delays is expected for per-packet path reconstruction in WSNs.

B. Testbed Experiments

We implement CSPR on TelosB mote and use a 29-node testbed to validate its feasibility and applicability. Twenty-nine TelosB motes are uniformly distributed in a square area. One node acts as the sink and is placed at the top left corner. Due to the limitation of experimental space, we configure the transmission power of each TelosB mote to the minimum level, and thus the communication range of each sensor node would be about 15 cm. To include the aMsr and bFlt two fields into the packet, we preserve the default packet header format and append them at the end of the payload. It provides the most flexibility to avoid additional communication overhead for the packets that do not need the path reconstruction, e.g., control traffics, which are usually exchanged among neighbors. In the experiments, each node generates packets randomly with an average interpacket interval of 1 s. Within the network, each intermediate node along a path updates the packet header following the requirements of each approach. The sink receives packets from network and separately executes the three approaches to reconstruct packet paths. The actual path is recorded in packets as ground truths. The experiment lasts for about 50 min and collects 60 000 packets.

The sink records the path reconstruction progress of each approach every 30 s. Concretely, for each approach, the portions of packets accurately recovered with only the core method and the complete approach are both computed. The evolution of path reconstruction progress of each approach is plotted in Fig. 9. From the figure, we find that Pathfinder and MNT can only reconstruct 47% and 36% packet paths based on their core methods, respectively. Most reconstruction failures are caused by packet

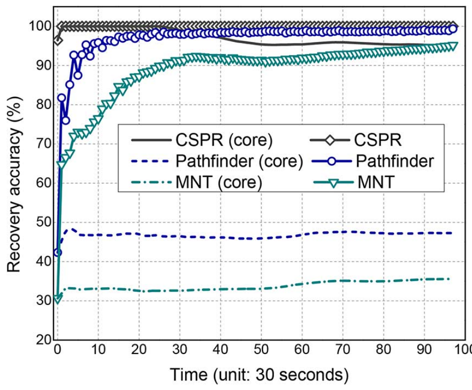


Fig. 9. Path reconstruction progresses with the core and complete methods of each approach.

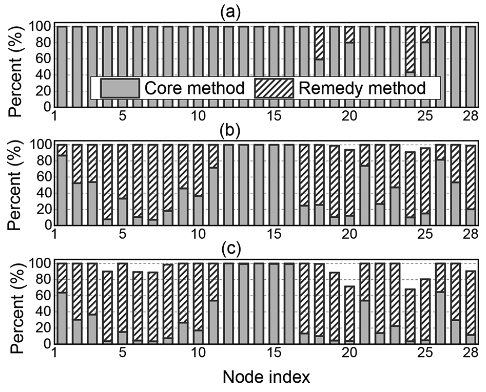


Fig. 10. Portion of packets recovered by core and remedy methods of each approach. (a) CSPR. (b) Pathfinder. (c) MNT.

losses (about 38% packet loss rate during experiment). With no fixed packet generation interval, Pathfinder loses its accuracy in locating anchors, which further harms the performance. Assisted by the remedy method (i.e., path speculation), they can achieve similar accuracy as the core method of CSPR, i.e., >95%. With heuristics path scanning, CSPR achieves accuracy of 100%. Fig. 10 further presents the detailed per-node path reconstruction performance. Twenty-four nodes can recover their packet paths just by the core method of CSPR. With respect to the other two approaches, however, most nodes achieve a good recovery accuracy via remedy method.

Rather than reconstructing path for each individual packet, CSPR distinguishes each path and reconstructs a routing path for a group of packets. As a result, once a path group has recovered its path, the future packets belonging to this group can easily obtain their paths through path verification. The already-recovered paths benefit all subsequent packets and can reduce the overall path recovery delay. In Fig. 11, we present the packet path recovery delay distribution of each approach. According to the statistics, for such a small network, CSPR reconstructs all paths based on a very small portion of packets ($<1\%$), and then obtains the paths of subsequent packet ($>99\%$) with no recovery delay. When the core methods of Pathfinder and MNT fail to reconstruct the paths, the remedy method would be used before the expiring of bound δ. We see that 31.5% and 40.8% packets have great path recovery delay (≥ 10) for Pathfinder and MNT, respectively. Though the remedy method can be immediately used once the core method fails, it triggers tremendous computation overhead.

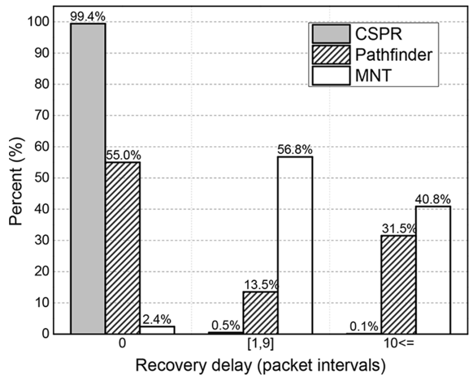


Fig. 11. Distributions of packet path recovery delay for each approach.

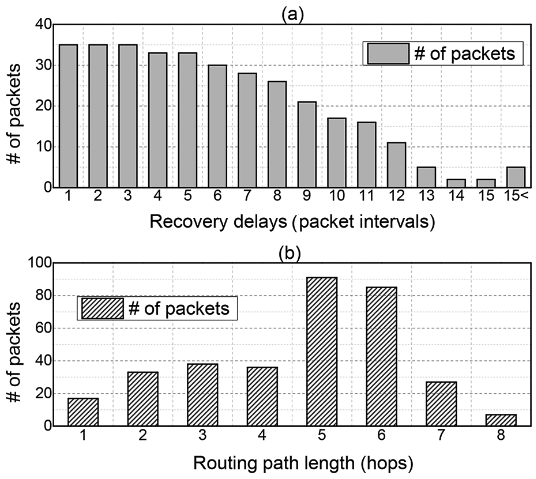


Fig. 12. (a) Path recovery delay distribution of delayed packets. (b) Routing path length distribution of delayed packets.

We further analyze the detailed recovery delay performances for those delayed packets by CSPR. From Fig. 12(a), we find that only a few packets are really recovered with great path recovery delay (e.g., ≥ 10). For most delayed packets, CSPR can still reconstruct their actual routing paths with a short waiting time, which is less than 10 packet intervals (corresponding to about 10 s in our testbed experiments). For those delayed packets, we also study the distribution of their routing path lengths and plot the results in Fig. 12(b). The packets traveling longer routing paths tend to be recovered later, which accords with intuitions. This is because the packets needed for successful path reconstruction by CSPR is about $M = ck \log(\frac{N}{k})$, which is proportional to the routing path length k. Note that in the small testbed network with 29 nodes, only a small portion of packets travel with long paths, e.g., $\geq$7 hops. Thus, the number of delayed packets in Fig. 12(b) becomes smaller when the path length is greater than 7.

C. Trace-Driven Simulations

To evaluate the scalability and efficiency of CSPR in practical large-scale and more dynamic networks, we conduct extensive trace-driven simulations by leveraging the practical packet trace from the real-deployed and large-scale WSN CitySee [29], as introduced in Section II-B. The packet trace includes packets from a network of 245 nodes with 174 829 packets in total.

CSPR includes three components to make per-packet path reconstruction a success, i.e., the compressive-sensing-based path

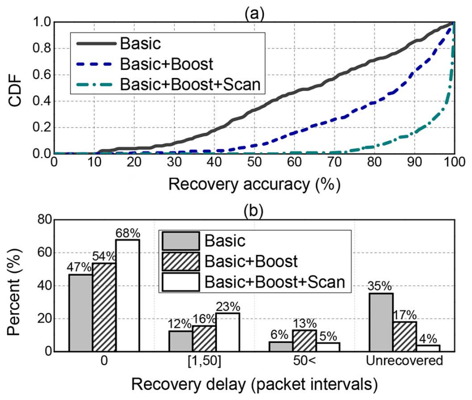


Fig. 13. (a) CDF of path recovery accuracy. (b) Distribution of path recovery delay.

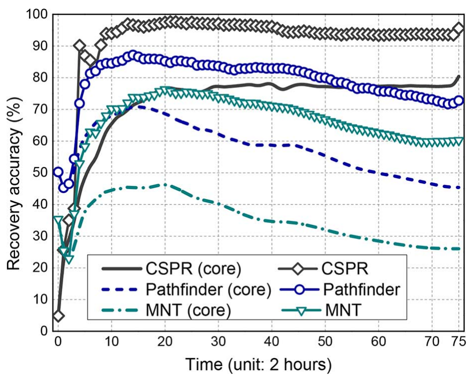


Fig. 14. Path reconstruction progresses with the core and complete methods of each approach.

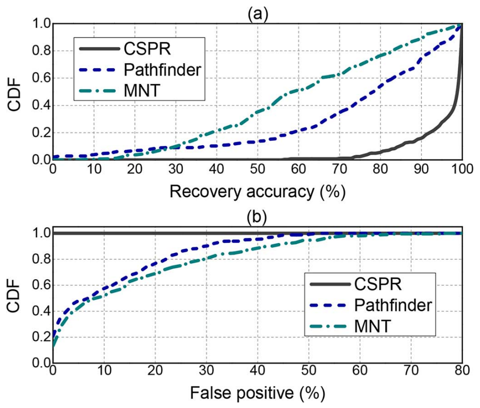


Fig. 15. (a) CDF of path recovery accuracy. (b) CDF of false positive for each approach.

reconstruction component (referred to as *Basic*), the path reconstruction optimization component based on path representation space shrink and path vector sparsity reduction (referred to as *Boost*), and the heuristic path scanning component (referred as *Scan*). To examine their efficiencies, we separately perform path reconstruction based on the *Basic* component, the *Basic* and *Boost* components, and all of the three components, and measure the recovery accuracy and recovery delay in the three execution cases. We present the CDF of recovery accuracy in Fig. 13(a) for the three execution cases. For most nodes, the *Basic* reconstructs paths for the greatest portion of packets, i.e., about 63.0%, which demonstrates its efficiency. Benefiting from the learned network topology and recovered paths, the *Boost* recovers paths for 17.4% more packets. The *Scan* contributes a considerable portion as well, which brings 15.2% accuracy promotion further. On the other hand, both *Boost* and *Scan* reduce the overall path recovery delay. In Fig. 13(b), we plot the statistical results of path recovery delay for all packets according to whether a packet path is recovered with zero delay, within the recovery delay bound δ, beyond the bound δ or not recovered at all. Thanks to *Boost* and *Scan*, many more packet paths can be recovered with much less delay. Overall, about 32% packets benefit from such optimizations and can recover their paths within the bound δ.

We apply the three approaches, i.e., CSPR, Pathfinder, and MNT, to reconstruct packet paths in the CitySee packet trace. In Fig. 14, we plot the path reconstruction progress for each approach with its core method and the complete method. Both Pathfinder and MNT perform poorly when only executing their core methods with recovery accuracy no more than 70% and 50%, respectively, at all time. Due to influences of topology dynamics and packet losses, the core methods of Pathfinder and MNT frequently fail to identify anchors for packets, and thus use the remedy method to recovery those packet paths. With the remedy methods, they achieve a higher final recovery accuracy 74% and 62%, respectively. The curve sharps of Pathfinder and MNT in Fig. 14 implicitly reflect the continuous influences of packet losses. One lost packet may be the anchor for many packets. On the contrary, CSPR achieves an accuracy 80.4% even only with its core method. With the remedy method, CSPR performs better and has a stable accuracy around 95% with the final accuracy 96%.

We further summarize the CDFs of path recovery accuracy and false positive, for each approach, in Fig. 15. From Fig. 15(a), we find that CSPR outperforms Pathfinder and MNT with significant advantages. Pathfinder and MNT perform poorly in the practical packet trace, which experiences severe topology dynamics and packet losses as revealed in Section II-B, with recovery accuracies of most nodes falling in the range between 50% and 80%. Only 23% and 45% nodes for MNT and Pathfinder, respectively, can achieve an accuracy $\geq$ 80%, while CSPR makes 95% nodes reach such level. Furthermore, about 85% nodes achieve a high accuracy $\geq$ 90%. Overall, the average recovery accuracies for MNT, Pathfinder, and CSPR are 62%, 74%, and 96%, respectively. In Fig. 15(b), we see nonnegligible false positives in Pathfinder and MNT, both of which verify the recovered paths mainly based on the XOR field in packets. Pathfinder checks the result with more information recorded in packet, and thus has fewer false positives than MNT. Such path verification manners result in on average 11% and 15% false positive for MNT and Pathfinder, respectively. Thanks to the path verification component enabled by the aMsr field, CSPR almost has no false positive.

We present the distribution of path recovery delay in Fig. 16(a), which also classify packets into four categories according to the path recovery delay of each packet. Note that the packets with false positive recovered paths are viewed as *unrecovered*. Overall, the timely recovered packet portions, i.e., packets in the first two categories, for MNT, Pathfinder, and CSPR are 60%, 73%, and 91%, respectively. It is worthy to note that CSPR recovers paths for the most packets with zero delay. Because of the packet classification mechanism

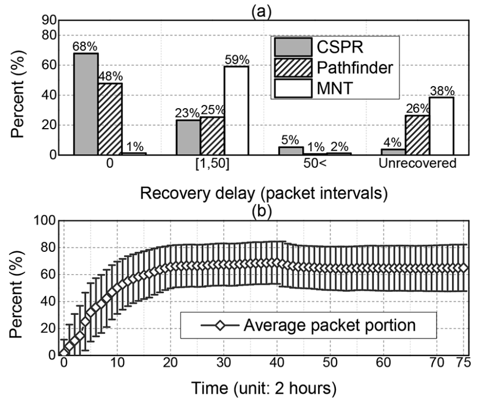


Fig. 16. (a) Distribution of path recovery delay. (b) Portion of packets benefited from the already-recovered paths.

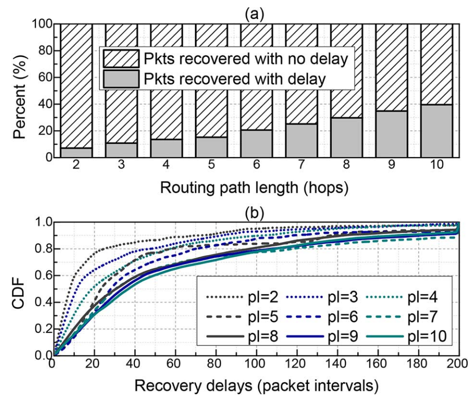


Fig. 17. (a) Percentages of paths reconstructed on time and with delays. (b) CDF of path recovery delays with different path lengths (i.e., *pl*).

in CSPR, a great portion of packets can recover their paths at a negligible cost of only executing path verification with the recovered paths in path groups. In Fig. 16(b), we further plot the portion of packets that benefit from such mechanism. After an accumulation phase of recovered paths, CSPR can find correct paths for about 65% packets. As a result, CSPR avoids repeating reconstruct those frequent traveled paths.

To understand the relation between the routing path length and path recovery delay in large-scale sensor networks, we plot the percentages of paths recovered on time and with delays with respect to the routing path length in Fig. 17. Concretely, we classify all packets according to their routing path lengths, and for each specific routing path length we calculate the ratios for both the packets that are timely reconstructed and the packets that are recovered with delays. According to Fig. 17(a), the longer routing paths packets have traveled, the more likely their paths will be reconstructed with delays, which coincides with the conclusion of Fig. 12(b). This is because CSPR needs to accumulate more packets to enable compressive-sensing-based path reconstruction for the longer routing paths. To further understand the distribution of path recovery delays, we plot Fig. 17(b) to show the CDF of the recovery delays for each specific path length. From Fig. 17(b), we find that in general the longer a path the packet travels, the larger recovery delay it will suffer. For example, about 58% packets traveling on paths with length of 7 hops can be reconstructed within 40 packet intervals, while about 80% packets traveling on paths with length of 3 hops can be recovered.

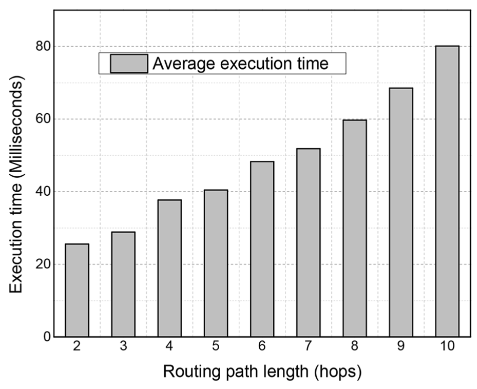


Fig. 18. Program execution time of compressive-sensing-based path reconstruction by CSPR versus the routing path length.

CSPR is a lightweight path reconstruction approach, and the overhead of the compressive sensing recovery, e.g., CoSaMP, is negligible on a server. In Fig. 18, we measure the program execution time as the performance metric to further understand the computation overhead of our design. For different routing path lengths, we plot the reconstruction delays, which are mainly caused by the compressive sensing recovery. Fig. 18 shows that the overhead due to compressive sensing recovery is negligible, which is less than 100 ms for all routing path lengths observed in the experiments.

In Section II-B, we have already observed about 20% natural packet losses. To examine the impacts of severe packet losses to the path reconstruction performance of each approach, we fabricate extra packet losses by randomly removing packets from the original trace. We present the recovery accuracy and false positive of each approach with various extra packet loss rates in Fig. 19. The extra packet loss rate labeled *"0"* corresponds to the original trace. With more packets lost, the performance of MNT drops quickly. MNT becomes to be not applicable, with accuracy $<50\%$, when the extra packet loss rate beyond 15%. The performance of Pathfinder is also affected, with accuracy dropping from 74% to 66%. These results illustrate that packet losses indeed have a strong impact on the performances of approaches relying on interpacket correlation. When anchors are lost, Pathfinder and MNT use the remedy method to search possible paths for packets and thus introduce false positives, due to their relaxed path verification manners. From Fig. 19(a), we observe nonnegligible false positives for Pathfinder and MNT with averages ranging from 4% to 15%. Contrastively, CSPR is insensitive to packet losses and achieves stable accuracy $>92\%$ with no false positive.

Overhead Comparison: Finally, we investigate the design overhead among our method, MNT, Pathfinder, and the straightforward method (named as *DirRec*) that directly records the relay node ID sequence. In Table I, we compare the four approaches from the following six aspects: the packet overhead, the in-sensor storage, the in-sensor computation overhead, the time complexity for each path reconstruction operation, the number of packets required in each path reconstruction operation, and the frequency of path reconstructions.

In summary, CSPR introduces comparable packet overhead, in-sensor storage, and computation overhead with other three approaches, but higher computation overhead at the server side for each run of the path reconstruction. However, in CSPR, after

TABLE I
COMPARISON AMONG DIFFERENT PATH RECONSTRUCTION APPROACHES[1]

	Pkt overhead	**Storage**	**Computation**	**Time complexity**	**# of pkts required**	**Reconstruction frequency**
CSPR	8 bytes	200 bytes	Path info. encoding	$O(\lambda MN)$	$1.5k\log(\frac{N}{k})$	Once for each path group
MNT	5 bytes	none	negligible	$O(WP)$	$2k$	Once for each packet
Pathfinder	7 bytes	100 bytes	Huffman encoding	$O(FP)$	k	Once for each packet
DirRec	$2k$ bytes	none	negligible	$O(1)$	1	Once for each packet

[1] M is the number of packets needed for path reconstruction, N is the network size. λ, the maximum iterations of CoSaMP [31], is set as 100. k is the routing path length. P is the maximum routing path length in the network. W is the size of potential anchor packet set in MNT. F is the size of packet set for offset estimation in Pathfinder.

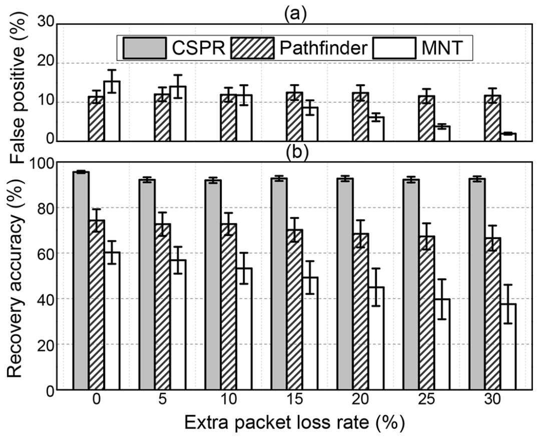


Fig. 19. (a) False positive and (b) recovery accuracy of the three approaches with various extra packet loss rates. Each value is an average of 10 runs.

one routing path was recovered, all the following packets traveling the same routing path do not need explicit path recovery. Therefore, the computation overhead amortized to each individual packet is negligible. As we have investigated in this section, the routing paths for about 99% and 65% of packets do not need to be explicitly reconstructed on our testbed and the CitySee packet trace, respectively.

V. RELATED WORK

Routing Path Reconstruction: CAPTRA [35] identifies a packet path through the coordinations among nodes in a network-wide scale. Alam *et al.* [2] adopt a probabilistic packet marking technique to trace the provenance of a packet. Both of the works cannot realize per-packet path reconstruction as CSPR. PathZip [25] compresses path information of a packet into a hash value and obtains the packet path by matching all possible paths with the hash value. As the computation complexity grows exponentially with the network size, PathZip may not scale to work with large networks. Pathfinder [16] and MNT [20], the two state-of-the-art approaches, reconstruct packet path relying on interpacket correlation. Their performances, however, are severely influenced by topology dynamics and packet losses, just as demonstrated in Section II-B. Different from them, CSPR is insensitive to network dynamics and lossy links due to its distinct design. In [24], we have reported the preliminary design of CSPR. In this journal version, we supplement the path vector sparsity reduction technique, perform more solid testbed and trace-driven evaluations for both the original design and the new technique, and provide detailed comparison between CSPR and the state-of-the-art approaches.

Network Tomography: Network tomography in wired networks has been well studied, and tremendous approaches have been proposed to investigate the internal behaviors [8], [14]. By actively generating probe packets from network nodes, network tomography mainly aims to recover the network topology or infer some link-level characteristics [27], e.g., delay or packet loss [34]. Restricted by the available resources, extensive probes are prohibited for network topography in WSNs. Recently, many works have been proposed to achieve network tomography in WSNs by leveraging statistical methods [32], group testing [7], compressive sensing [37], and optimization methods [15]. Compared to those works, CSPR reconstructs routing path for each individual packet without triggering extra probe packets.

Network Diagnosis: Network diagnosis aims at inferring the root cause for abnormal networking symptoms and maintaining the health of deployed WSNs. Relying on the collected system metrics from network, Sympathy [33] pinpoints the root cause of network failures through a decision tree. PAD [23] captures abnormal events and infers the root cause for the observed abnormity in a probabilistic manner. The active packet marking scheme in PAD can only recover one routing path for each source node, while the underlying network topology is required to be relatively stable. PD2 [6] is a data-centric approach that locates performance problems based on data flows. D2 [10] detects and diagnoses anomaly by mining network symptoms. AD [30] exploits the correlation among different system metrics to discover silent failures. Existing works in this category are orthogonal to CSPR and are potentially benefitted from the outputs of CSPR for more accurate and fine-grained diagnostic results.

VI. CONCLUSION

In this paper, we present the CSPR, a compressive-sensing-based path reconstruction approach. Different from the state-of-the-art approaches, CSPR is inherently insensitive to network dynamics and lossy links. Extensive evaluations through both testbed-based experiments and trace-driven simulations show that CSPR outperforms the state-of-the-art approaches in various network settings.

ACKNOWLEDGMENT

The authors acknowledge the support of practical packet trace from the CitySee project [29].

REFERENCES

[1] “TelosB mote datasheet,” [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf
[2] S. Alam and S. Fahmy, “A practical approach for provenance transmission in wireless sensor networks,” *Ad Hoc Netw.*, pp. 28–45, 2013.

[3] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.

[4] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[5] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[6] Z. Chen and K. G. Shin, "Post-deployment performance debugging in wireless sensor networks," in *Proc. IEEE RTSS*, 2009, pp. 313–322.

[7] M. Cheraghchi, A. Karbasi, S. Mohajer, and V. Saligrama, "Graph-constrained group testing," *IEEE Trans. Inf. Theory*, vol. 58, no. 1, pp. 248–262, Jan. 2012.

[8] A. Coates, A. O. Hero, III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Process. Mag.*, vol. 19, no. 3, pp. 47–65, May 2002.

[9] D. Dong, M. Li, Y. Liu, X.-Y. Li, and X. Liao, "Topological detection on wormholes in wireless ad hoc and sensor networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1787–1796, Dec. 2011.

[10] W. Dong, C. Chen, J. Bu, X. Liu, and Y. Liu, "D2: Anomaly detection and diagnosis in networked embedded systems by program profiling and symptom mining," in *Proc. IEEE RTSS*, 2013, pp. 202–211.

[11] W. Dong, Y. Liu, Y. He, and T. Zhu, "Measurement and analysis on the packet delivery performance in a large scale sensor network," in *Proc. IEEE INFOCOM*, 2013, pp. 2679–2687.

[12] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[13] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing routing holes in sensor networks," in *Proc. IEEE INFOCOM*, 2004, pp. 187–200.

[14] C. Fragouli, A. Markopoulou, and S. Diggavi, "Topology inference using network coding," in *Proc. 44th Annu. Allerton Conf. Commun., Control, Comput.*, 2006.

[15] Y. Gao *et al.*, "Domo: Passive per-packet delay tomography in wireless ad-hoc networks," in *Proc. IEEE ICDCS*, 2014, pp. 419–428.

[16] Y. Gao *et al.*, "Pathfinder: Robust path reconstruction in large scale sensor networks with lossy links," in *Proc. IEEE ICNP*, 2013, pp. 1–10.

[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. ACM SenSys*, 2009, pp. 1–14.

[18] L. He, J. Pan, and J. Xu, "A progressive approach to reducing data collection latency in wireless sensor networks with mobile elements," *IEEE Trans. Mobile Comput.*, vol. 12, no. 7, pp. 1308–1320, Jul. 2013.

[19] L. He *et al.*, "Evaluating service disciplines for on-demand mobile data collection in sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 797–810, Apr. 2014.

[20] M. Keller, J. Beutel, and L. Thiele, "How was your journey?: Uncovering routing dynamics in deployed sensor networks with multi-hop network tomography," in *Proc. ACM SenSys*, 2012, pp. 15–28.

[21] Z. Li, M. Li, and Y. Liu, "Towards energy-fairness in asynchronous duty-cycling sensor networks," *Trans. Sensor Netw.*, vol. 10, no. 3, p. 38, 2014.

[22] Z. Li, J. Wang, and Z. Cao, "Ubiquitous data collection for mobile users in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2246–2254.

[23] Y. Liu, K. Liu, and M. Li, "Passive diagnosis for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1132–1144, Aug. 2010.

[24] Z. Liu, Z. Li, M. Li, W. Xing, and D. Lu, "Path reconstruction in dynamic wireless sensor networks using compressive sensing," in *Proc. ACM MobiHoc*, 2014, pp. 297–306.

[25] X. Lu, D. Dong, X. Liao, and S. Li, "PathZip: Packet path tracing in wireless sensor networks," in *Proc. IEEE MASS*, 2012, pp. 380–388.

[26] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *Proc. ACM MobiCom*, 2009, pp. 145–156.

[27] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Efficient identification of additive link metrics via network tomography," in *Proc. IEEE ICDCS*, 2013, pp. 581–590.

[28] Q. Ma, K. Liu, X. Miao, and Y. Liu, "Sherlock is around: Detecting network failures with local evidence fusion," in *Proc. IEEE INFOCOM*, 2012, pp. 792–800.

[29] X. Mao, X. Miao, Y. He, X. Li, and Y. Liu, "CitySee: Urban CO_2 monitoring with sensors," in *Proc. IEEE INFOCOM*, 2012, pp. 1611–1619.

[30] X. Miao, K. Liu, Y. He, Y. Liu, and D. Papadias, "Agnostic diagnosis: Discovering silent failures in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2011, pp. 1548–1556.

[31] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmonic Anal.*, vol. 26, no. 3, pp. 301–321, 2009.

[32] H. X. Nguyen and P. Thiran, "Using end-to-end data to infer lossy links in sensor networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.

[33] N. Ramanathan *et al.*, "Sympathy for the sensor network debugger," in *Proc. ACM SenSys*, 2005, pp. 255–267.

[34] P. Sattari, A. Markopoulou, C. Fragouli, and M. Gjoka, "A network coding approach to loss tomography," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1532–1562, Mar. 2013.

[35] D. Sy and L. Bao, "CAPTRA: Coordinated packet traceback," in *Proc. ACM/IEEE IPSN*, 2006, pp. 152–159.

[36] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 131–155, 1st Quart., 2012.

[37] W. Xu, E. Mallada, and A. Tang, "Compressive sensing over graphs," in *Proc. IEEE INFOCOM*, 2011, pp. 2087–2095.

[38] T. Zhu *et al.*, "Understanding routing dynamics in a large-scale wireless sensor network," in *Proc. IEEE MASS*, 2013, pp. 574–582.

Zhidan Liu received the B.E. degree from Northeastern University, Shenyang, China, in 2009, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2014, both in computer science and technology.

He is currently a Research Fellow with Nanyang Technological University, Singapore. His research interests include wireless sensor networks, mobile computing, and data analytics.

Zhenjiang Li (M'12) received the B.E. degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2007, and the M.Phil. degree in electronic and computer engineering and Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2009 and 2012, respectively.

He is currently a Research Fellow with Nanyang Technological University, Singapore. His research interests include distributed networking systems, cyber-physical systems, and mobile computing.

Mo Li (M'06) received the B.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 2004, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2009.

He is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include wireless sensor networking, pervasive computing, and mobile and wireless computing.

Wei Xing received the B.E., M.E., and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China, in 1989, 1992, and 2009, respectively.

He joined the Department of Control in College of Information Technology, Zhejiang University, in 1992. Since 2002, he has been with the College of Computer Science and Technology, Zhejiang University, where he is currently an Associate Professor. His research interests include multimedia technology and Internet of Things.

Dongming Lu received the B.E., M.E., and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China, in 1989, 1991, and 1994, respectively.

He is currently a Professor with the College of Computer Science and Technology, Zhejiang University. His research interests include Internet of Things, multimedia technology, and digital preservation of cultural heritage.