This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Measurement and Analysis on the Packet Delivery Performance in a Large-Scale Sensor Network

Wei Dong, *Member, IEEE*, Yunhao Liu, *Senior Member, IEEE*, Yuan He, *Member, IEEE*, Tong Zhu, *Member, IEEE*, and Chun Chen, *Member, IEEE*

***Abstract*—Understanding the packet delivery performance of a wireless sensor network (WSN) is critical for improving system performance and exploring future developments and applications of WSN techniques. In spite of many empirical measurements in the literature, we still lack in-depth understanding on how and to what extent different factors contribute to the overall packet losses for a complete stack of protocols at large scale. Specifically, very little is known about: 1) when, where, and under what kind of circumstances packet losses occur; 2) why packets are lost. As a step toward addressing those issues, we deploy a large-scale WSN and design a measurement system for retrieving important system metrics. We propose MAP, a step-by-step methodology to identify the losses, extract system events, and perform spatial-temporal correlation analysis by employing a carefully examined causal graph. MAP enables us to get a closer look at the root causes of packet losses in a low-power ad hoc network. This study validates some earlier conjectures on WSNs and reveals some new findings. The quantitative results also shed lights for future large-scale WSN deployments.**

***Index Terms*—Packet delivery performance, measurement, packet losses, wireless sensor networks.**

I. Introduction

AS AN emerging technology that bridges cyber systems and the physical world, wireless sensor networks (WSNs) are envisioned to support numerous unprecedented applications. We have witnessed many research studies, deployments of real systems, and substantive practical applications in recent years.

In the past years, many WSN protocols have been reported and shown to be effective in testbed or small-scale networks. On the other hand, it is not uncommon to see that many real-world deployments often adopt a set of tailored protocols to fulfil the application's requirements. We believe that it is important to understand the performance of some *well-principled protocols in combination at large scale*.

From a networking perspective, the most basic aspect of wireless communication is the packet delivery performance: the spatial-temporal characteristics of packet loss and its environmental dependence [1]. 1) The packet delivery performance deeply impacts the performance of application built upon the network. For example, data loss in a sensor network for environmental surveillance may mislead the users to incorrect knowledge of the environment and make inappropriate decisions. Data loss in a sensor network for traffic monitoring may lead to incorrect traffic management and cause serious problems like traffic jams. Data loss in a sensor network for structure health monitoring will probably cause the miss of critical problems in the monitored object and potentially lead to unpredictable accidents. Data loss in a sensor network for industrial control will probably cause the inability to control the industrial process, making the whole system simply fail to work. 2) For energy-constrained sensor networks, packet delivery performance is important since that translates to network lifetime [1]. Sensor networks typically use retransmissions upon loss detection. If there are severe packet losses, a node will repeatedly retransmit the packet, resulting in much more energy consumptions. Hence, poor cumulative packet delivery performance across multiple hops may expend significant energy, impairing the network lifetime.

Many deployments have reported the overall packet delivery performance [2], [3]. Also, many empirical measurement studies show how some specific factors impact the packet delivery performance via controlled experiments [1], [4]. However, we usually do not know how and to what extent different factors contribute to the overall packet losses for a complete stack of protocols at large scale. Specifically, very little is known about: 1) when, where, and under what kind of circumstances packet losses occur; 2) why packets are lost. Answers to the above questions are critical for improving system performance and exploring future developments and applications of WSN techniques.

Understanding the packet delivery performance in an operating WSN is challenging due to the following facts. First, data packets might be lost during multihop transmissions, and thus data collection is incomplete by nature. It is very difficult to acquire the complete information of the internal status of the network. Second, operational efforts to disclose the root causes

Manuscript received December 30, 2012; revised September 06, 2013; accepted October 19, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Mahajan. This work was supported by the National Basic Research Program (973 Program) under Grant 2014CB347800; the National Science Foundation of China under Grants No. 61202402, No. 61070155, and No. 61170213; the National Key Technology R&D Program (2012BAI34B01); the Fundamental Research Funds for the Central Universities (2012QNA5007); the Research Fund for the Doctoral Program of Higher Education of China (20120101120179), and the NSFC Distinguished Young Scholars Programunder Grant No. 61125202. A preliminary version of this work was published in the Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Turin, Italy, April 14–19, 2013.

W. Dong and C. Chen are with the College of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: dongw@zju.edu.cn; chenc@zju.edu.cn).

Y. Liu, Y. He, and T. Zhu are with the School of Software and Tsinghua National Lab for Information Science and Technology (TNLIST), Tsinghua University, Beijing 100084, China (e-mail: yunhao@greenorbs.com; he@greenorbs.com; zhutong@greenorbs.com).

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.2013.2288646

1063-6692 © 2013 IEEE

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

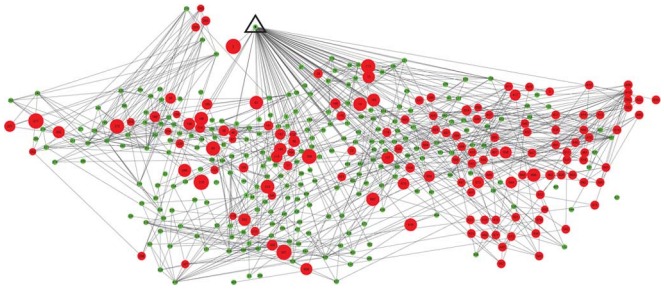


Fig. 1. Network topology where the node with a triangle is the sink node. The green nodes represent nodes with PDR $\geq$ 90%. The red nodes represent nodes with PDR $<$ 90%, and the length of the radius indicates the number of lost packets.

behind packet losses are insufficient, and few efforts have been validated to be effective at large scale.

As a step toward addressing those challenges, we deploy GreenOrbs, a large-scale and long-term WSN system in the wild. The network we measure has been in continuous operation since December 2010 with nearly 400 nodes. For the sensor node hardware, we use the commonly used TelosB nodes. For the software, we use TinyOS and its radio stack, including the LPL MAC, the CTP collection protocol [5], and the Drip dissemination protocol [6].

Based on GreenOrbs, we propose MAP, a practical methodology for Measuring and Analyzing the Performance of a large operating WSN. MAP incorporates a carefully designed measurement module for retrieving networking and system metrics. MAP includes three steps for analyzing the packet losses. First, it identifies the loss events as well as other important system events called triggers. Second, it carefully tracks the interactions inside the WSN system by means of a causal graph. Third, it examines the spatial-temporal correlations among system events having causal relationships.

Using this methodology, we are able to conduct a deep examination of packet losses. For example, for a recent deployment of our system, Fig. 1 shows the geographic distribution of packet losses. The red nodes are those with packet delivery ratio (PDR) less than 90%, and the length of the radius indicates the number of lost packets. We note that while most previous deployment studies report the delivery performance as in Fig. 1, they usually lack detailed analysis for classifying the losses into smaller categories that can be more useful for gaining system insights. The combination of multiple factors makes the overall packet losses exhibit complex patterns that are extremely difficult to reason about. Compared to previous deployment studies, MAP takes a further step to decompose the overall losses into smaller categories that can be related closer to the underlying causes. It can also reveal the spatial-temporal distributions of different losses.

The contributions of this paper are summarized as follows.

1) We examine the packet delivery performance for a complete stack of TinyOS protocols in a large-scale operating WSN.
2) We develop a methodology to investigate the underlying causes for those losses. We quantify to what extent each individual cause contributes to the overall identified losses.
3) We give implications and lessons learned to guide future WSN designs and deployments.

The rest of this paper is structured as follows. Section II describes the related work. Section III introduces the network and the particular datasets we use in our study. Section IV shows basic statistics of the network. Section V presents the loss identification algorithm and the spatial-temporal distribution of packet losses. Section VI describes the methodology for revealing the root causes. Section VII summarizes implications and lessons we have learned. Section VIII presents a discussion of our approach before we conclude in Section IX with an outlook on future works.

II. Related Work

This section discusses related works including existing WSN deployments, measurement studies in wireless networks, and the Internet.

Sensor Network Deployments: Many WSN prototypes have been deployed in the recent years. Table I summarizes representative sensor network deployments with some key network metrics. Compared to other deployments, we can see that our GreenOrbs network: 1) employs the latest TinyOS protocol stack; 2) uses the largest number of sensor nodes per subnet.

During the year 2002–2003, a WSN for habitat monitoring at Great Duck Island [7] is deployed. Tolle *et al.* [8] report a sensor network consisting of 33 nodes to monitor the microclimate of a redwood tree, covering an area of about 50 square meters. Werner-Allen *et al.* [9] have deployed a WSN of 16 nodes to monitor an active volcano. VigilNet [10] includes 200 nodes to support military surveillance, covering an area of 100 $\times$ 100 square meters. ExScal [11], [2] attempts to deploy a WSN at a large scale. The system consists of over 1000 sensor nodes and 200 backbone nodes, while it fails to keep in continuous operation for long. Bapat *et al.* [2] analyze the yield of ExScal. SensorScope [12] is a real-world WSN system on a rock glacier, of which the largest deployment consists of nearly 100 sensor nodes. The PermaSense project [13] collects long-term measurements of diverse physical phenomena for geophysical research. It employs a customized protocol, Dozer, for reliable data collection. A wireless clinical monitoring system is introduced in [14]. The system collects pulse and oxygen saturation readings from patients with the TinyOS protocol stack. Liu *et al.* [3] present measurement results of a large-scale sensor network in the forest.

We can see from the above deployments that data collection from multiple sensor nodes to a sink is a common application scenario for WSNs. TinyOS and its protocol stack are widely used in real sensor networks. A key difference between our current study and previous studies is that we examine the correlations among different events and quantitatively analyze the causes for packet losses, while most of the previous works only report the overall network performance without deep analysis on the underlying causes. With the analysis results from our study, developers and network operators can focus on the bottleneck factors to improve the network performance.

Wireless Measurements: There are some dedicated measurement studies in wireless and sensor networks. 1) Aguayo *et al.* [15] present a link-level measurement study on an 802.11b mesh network. Our measurements are based on different hardware and software. Moreover, sensor

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
SENSOR NETWORK DEPLOYMENTS

Project	# of nodes	Duration (days)	Hops	PDR	H/W	S/W
Great Duck Island [7]	49 (single hop subnet) + 98 (multihop subnet)	115	<=6.3	28%-58%	mica2dot	TinyOS 1.x, LPL, Woo's protocol
Redwood [8]	33	44	N/A	49%	mica2dot	TinyOS 1.x, MintRoute, TinyDB
Volcanno [9]	16	19	N/A	19%-83%	Tmote Sky	TinyOS 1.x, MintRoute, Deluge, FTSP, Fetch
VigilNet [10]	200	N/A	N/A	~80%	XSM+Mica2	TinyOS 1.x, customized protocol
ExScal [11]	1200 (20-50 per subnet)	14	N/A	72.83%	XSM+XSS	customized
SensorScope [12]	~100	60	N/A	54%-99%	TinyNode (w. solar)	TinyOS 2.x sync duty cycling protocol
PermaSense [13]	24	~365	<=7	99%	TinyNode	TinyOS 2.x Dozer
Clinic monitoring (indoor) [14]	~60	41	N/A	99%	TelosB	TinyOS 2.x
GreenOrbs [3]	~400 (in one subnet)	~365	<=12	81.40%	TelosB	TinyOS 2.x

networks employ low-power radios and dynamic routing protocols. 2) Zhao *et al.* [1] report a measurement study on packet delivery performance using 60 Mica nodes. Srinivasan *et al.* [4] present measurement of packet delivery performance of the Telos and MicaZ platforms. Natarajan *et al.* [16] measure and analyze the link-layer behavior of a body area network at 2.4 GHz. Maheshwasi *et al.* [17] conduct a measurement study on interference modeling and scheduling on two 20-node TelosB testbeds. These measurement studies focus on impact of each individual factor using controlled experiments, while we present measurement results in an operating WSN. Those measurements are important for understanding the impacts of particular factors at different layers. The adopted measurement approaches, however, are targeted at individual aspect and cannot be easily integrated for a synthetical study in an operating WSN since they do not consider the joint impact of many aspects together, such as PHY, LPL MAC, and multihop routing.

There are also some passive measurement tools using sniffers. Mahajan *et al.* [18] propose Wit, a tool that builds on passive monitoring to analyze the MAC-level behaviors of operational wireless networks. Chen *et al.* present LiveNet [19], a system for investigating the behaviors of a deployed WSN via passive monitoring. The sniffer infrastructure in LiveNet is able to capture detailed wireless behaviors, but at the same time incurs excessive extra overhead due to the need of deploying additional sniffer nodes and collecting the logs back for analysis.

Internet Measurements: For years, network measurement has been a hot topic in the field of Internet, which attracts many research efforts. The Internet employs a layered network architecture consisting of end systems, routers, and wired links. The available data sources for analysis and the measurement approaches are thus different from WSNs consisting of resource-constrained sensor nodes and wireless links. Wang *et al.* [20] conduct a measurement study on the impact of routing events on the end-to-end path performance. They show that end-to-end Internet path performance degradation is correlated with routing dynamics and analyze the root cause of the correlation between routing dynamics and such performance degradation. Turner *et al.* [21] present a methodology for understanding the causes and impacts of link failures. They opportunistically mine data sources that are already available in modern network environments and analyze over five years of failure events in a large regional network. Mahimkar *et al.* [22] study the impact of upgrades on network key performance indicators in a large operational network.

The viewpoint of existing Internet measurement studies can be regarded as important references for our work in the WSN context. Nevertheless, understanding the behavior and performance of a WSN is an even more complex and challenging task due to: 1) the complex behaviors of the network and its nodes; 2) the lack of common infrastructure for the retrieval of system events; and 3) the insufficient operational efforts for categorizing the losses. Hence, we need a new measurement and analysis approach to understanding the packet delivery performance of a WSN.

III. DATA SOURCES

In order to set the context for our analysis, we briefly describe our system first and then detail the particular data sources available.

A. GreenOrbs Network

Our ongoing research project aims at building a long-term and large-scale WSN system in the forest. It employs the TelosB mote [23] with msp430f1611 processor and CC2420 radio. The project was started from April 2009. From August 2010, we rebuilt the software based on TinyOS 2.1.1 [24], with an improved architecture and implementation of the measurement module.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Basically, each sensor node reads the sensor data, measures and records system status, and delivers three kinds of packets to the sink with a period of 10 min. The application uses TinyOS and its network protocols, including the TinyOS LPL MAC protocol (with a sleep interval of 500 ms) for achieving low+duty cycling, the CTP routing protocol [5] for multihop routing, and the Drip dissemination protocol [6] for disseminating and configuring key system parameters.

We perform careful accounting in our measurement module. For example, we instrument the LPL MAC protocol to precisely track the radio duty cycle. We also implement methods for obtaining per-packet routing path and per-packet generation time, which are essential for performing spatial-temporal correlation analysis. Our current measurement module is not general enough to be incorporated into other protocol suites without any modifications, and we plan to make it more general in the future.

On December 10, 2010, we started a new deployment of the system with nearly 400 nodes in the campus woodlands (with the power level of 31), covering an area of about 60 000 m^2. A single TelosB sink node was used for collecting data. As can be seen from Table I, the GreenOrbs network has the largest number of sensor nodes per subnet.

In this paper, we analyze the collected packets of 10 days starting from December 19, 2010. The trace contains 1 137 430 packets in total.

B. Collected Packets

The sink node collects three kinds of packets with CTP collection types C1, C2, and C3, respectively. Our measurement and analysis in the subsequent sections are based on the data fields in these packets.

The C1 packet contains two kinds of information: 1) sensor data, including temperature, humidity, light, and voltage; 2) routing information, including path-ETX [26] from the source node to the sink node, and node IDs along the path (with a maximum number of 10). record paths explicitly

The C2 packet contains the routing table with a maximum neighbor number of 10. Each routing table entry contains: 1) the neighbor node ID; 2) the received signal strength indicator (RSSI) value from the neighbor; 3) the link-ETX estimate to the neighbor; 4) the path-ETX estimate to the sink.

The C3 packet contains various counters. For example: 1) the `CPU` counter records the accumulated task execution time in unit of $\frac{1}{32}$ms; 2) the `radio` counter records the accumulated radio-on time in milliseconds; 3) the `transmit` counter records the accumulated number of transmitted packets.

The above-mentioned three kinds of packets also share a common packet header including: 1) the `source` field, indicating which node the packet originates from; 2) the `seqno` field that increments when CTP sends a packet; 3) the `thl` field that indicates the hop count of the arriving packet; 4) the `source_time` field that is the time instant when the source node transmits the packet in its local time; (5) the `sink_time` field that is the packet reception time in the local clock of the sink node.

IV. Basic Statistics

With collected packets of 10 days, we are able to extract some basic statistics about the working system. During the measurement period, there are 343 nodes with PDR $\geq 10\%$. We detect that the sink was down from 14:40 p.m. December 24, 2010, to 8:20 a.m. December 25, 2010. In our analysis, we exclude the sink-down time and nodes with PDR $< 10\%$ in order not to bias our analysis.

As each node sends three packets every 10 min, we know the number of packets that should be received during the measurement period when the delivery from the source node is fully reliable. Without considering packet losses during the sink-down time, the system achieves an average PDR of 81.3%. We believe the packet delivery performance of our network is a typical representative for TinyOS protocols in real-world deployments (as can be seen from Table I). Compared to tailored protocols that are reported to achieve 99.9% reliability at permilli in small-scale networks [5], [27], the current ready-to-use TinyOS protocols are not good enough in large networks.

Please refer to our conference paper for more detailed system statistics [28].

V. Loss Identification: A First Step

Basically, we identify packet losses by observing gaps in sequence numbers (`seqno`). In our study, we identify loss event of the form $\langle$`ID, stime, etime, size`$\rangle$ where `ID` is the source node ID, `stime` and `etime` denote the start time and end time, and `size` denotes the loss size that is the maximum number of consecutive packets lost by the same source node. Two types of packet losses can be identified by observing the `drop_no_ack` and `drop_overflow` counters contained in C3 packets.

We also want to identify other interesting system events, called *triggers* (e.g., packet corruption, loop, reboot, etc.), which can explain the loss events. A trigger is in the form of $\langle$`ID, stime, etime, scope`$\rangle$ where `ID` is the source node ID, `stime` and `etime` denote the start time and end time, and `scope` contains a list of nodes that are likely to be impacted by the trigger.

A. Algorithm

Now we turn our attention to the loss identification algorithm. We use the 1-B `seqno` field in the CTP header for loss identification. A practical challenge in loss identification is that the 1-B seqno can easily overflow. Using a seqno field with more bits (e.g., 16 bits) will simplify the task. However, the GreenOrbs implementation keeps the header unmodified at that time, which complicates our algorithm at the PC side.

Basically, we identify the loss events using the following steps.

1) We validate packet fields for filtering out corrupted packets.
2) We assign each packet a virtual seqno (vseqno) that never overflows. how to assign this?

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

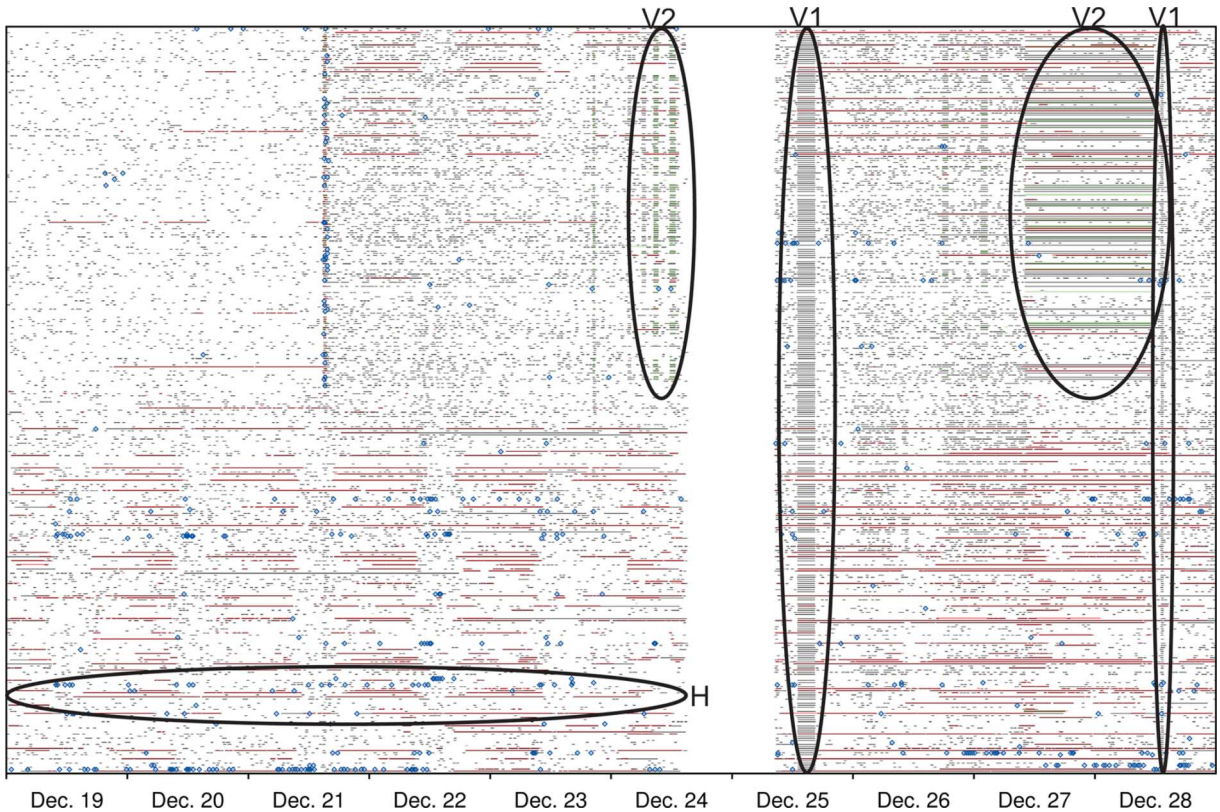


Fig. 2. Packet losses at a glance. The y-axis denotes the node IDs in ascending order. A black line indicates a loss event. A diamond indicates a reboot event. Vertical bands in Dec. 25 and Dec. 28 labeled V1 correspond to loss events happening at all nodes. Vertical bands in Dec. 24 and Dec. 27 and 28 labeled V2 correspond to loss events happening at a subset of nodes. The horizontal bands in the bottom left corner of Fig. 2 labeled H correspond to loss events happening at particular nodes. The sink is down during 14:40 p.m. Dec. 24, 2010, to 8:20 a.m. Dec. 25, 2010.

3) We sort the packets for each node according to the virtual seqno and identify the loss events by looking for gaps in the virtual seqno.

We also use two counters in C3 packets, i.e., `drop_no_ack` and `drop_overflow`, to identify two particular kinds of packet losses. We call these two types of packet losses no-ack drops and overflow drops, respectively.

B. Results

We have identified a total of 181 862 losses. We have also identified 5930 no-ack loss events (by examining the `drop_no_ack` counter), containing 84 030 losses, and 347 overflow loss events (by examining the `drop_overflow` counter), containing 5219 packet losses. They contribute to nearly 50% of the identified losses.

Fig. 2 shows the loss events during the measurement period. The x-axis denotes the time, and y-axis denotes the source node IDs in ascending order. Each loss event is represented by a black line located according to its start time and end time. We also plot the no-ack loss event by a red line and the overflow loss event by a green line. A diamond indicates a reboot event. We can make several observations from Fig. 2.

Vertical Banding: We can see two types of vertical bands here. 1) Vertical bands in Dec. 25 and Dec. 28 labeled V1 in Fig. 2 correspond to loss events happening at all nodes. The root cause should be at the sink side. 2) Vertical bands in Dec. 24 and Dec. 27 and 28 labeled V2 in Fig. 2 correspond to loss events happening at a subset of nodes. This is mostly caused by routing loops. An evidence is that we also observe overflow losses at a subset of nodes (green lines).

Horizontal Banding: The horizontal band in the bottom left corner of Fig. 2 labeled H corresponds to loss events happening at particular nodes. We observe that those nodes experience heavy packet losses during the measurement period. Most of them also experience no-ack loss events, suggesting that those nodes may have very poor link qualities to neighboring nodes, causing the retransmission threshold to be exceeded. Interestingly, we observe that most of these nodes experience recoveries in the midday. We will further look into such a phenomenon in the following section.

VI. Root Causes: A Closer Look

To investigate the root causes of packet losses, we would like to perform correlation analysis between the loss events and the triggers identified in Section V.

A. Spatial-Temporal Correlation Analysis

Intuitively, if a loss event is highly correlated with a trigger, it is caused by the trigger with a high probability. Fig. 3 shows the causal relationships we will investigate in this paper.

We investigate the following categories of causes:

A) Sink-side failures that are further classified into:
   A1) sink node failures;
   A2) PC-end failures.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

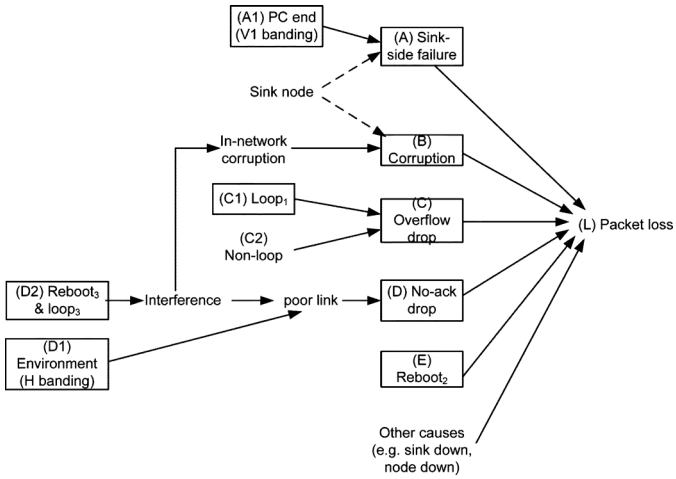


Fig. 3. Causal relationships of system events to loss.

B) Corruptions that are further classified into:
   B1) in-network corruptions;
   B2) corruptions at the sink node.
C) Overflow drops that are further classified into:
   C1) loop-induced overflow drops;
   C2) non-loop overflow drops.
D) No-ack drops that are further classified into:
   D1) no-ack drops due to environment-induced link degradation (env-no-ack drops);
   D2) no-ack drops due to in-network interference (interference-no-ack drops). Those losses can be induced by node reboot or routing loops. Both node reboot and routing loops will cause a very high beaconing rate in the current implementation of CTP/LPL, causing severe interference to neighboring nodes.
E) Node reboot that can directly cause queued packets of the downstream nodes to be dropped at an intermediate node.

There are other causes such as sink down or node down. We note that a single trigger may have different impacts. For example, a reboot may cause the queued packets to be dropped directly or cause interference to all neighboring nodes, resulting in no-ack drops at those nodes. A routing loop may cause overflow drops directly or cause interference to all neighboring nodes, resulting in no-ack drops. The multiple impacts of a single event and the complex interactions among triggers and the loss events make the delivery performance of a sensor network extremely difficult to reason about.

As each trigger is annotated with a start time and an end time, we use temporal correlation to match it with loss events. To find matches, we widen the start time and end time of a trigger by a time lag to compensate for factors like delayed reporting. The setting of the time lag, however, depends on the scenario under consideration. To minimize the false positives, we also use spatial correlation, i.e., we consider the impact scope of a trigger. As mentioned in Section V, the `scope` field in a trigger contains a list of nodes that are likely to be impacted by the trigger.

We consider three kinds of impact scopes in this study:
1) trigger.`scope` := trigger.`ID`;
2) trigger.`scope` := downstream nodes of trigger.`ID`;
3) trigger.`scope` := neighboring nodes of trigger.`ID`.

The downstream nodes of trigger.`ID` are found by inspecting all C1 packets from trigger.`ID` received in the time window [trigger.`stime`$-$w, trigger.`etime`] (w $= 10$ min in this study). As each C1 packet contains the forwarding path toward the sink, the downstream nodes for a particular node can be found.

The neighboring nodes of trigger.`ID` are found by inspecting C2 packets in the time window [trigger.`stime`$-$w, trigger.`etime`]. The neighboring nodes are those that either appear in the routing table of trigger.`ID` (upstream nodes) or whose routing tables contain trigger.`ID` (downstream nodes).

We will use a subscript to differentiate triggers with different scopes: trigger_1 has impacts on trigger.`ID`, trigger_2 has impacts on all downstream nodes of trigger.`ID`, and trigger_3 has impacts on all neighboring nodes of trigger.`ID`. In correlating triggers to loss events (or other triggers), we ensure that the `ID` in the loss event is contained in the impact scope of the trigger, i.e., event.`ID` $\in$ trigger.`scope`.

Validation: For deployed systems, it is difficult to obtain all levels of ground truth. We employ a variety of mechanisms to provide certain levels of ground truth in order to verify our results. For example, we use sniffer nodes to verify that the channel of our sensor network (i.e., channel 15) is not interfered by external noises. We also maintain a ticket system to record important system events such as weather changes, sink down, etc.

In order to quantitatively verify the accuracy of our correlation analysis algorithm, we perform controlled experiments using the same GreenOrbs program in a 5×10-grid testbed with a power level of 1, manually injecting different causes of packet losses. We inject the following kinds of events for introducing packet losses:

- move a node away from other nodes and move it back after a duration;
- reboot a node by pressing its reset button;
- manually assign parents in order to create routing loops;
- increase the traffic rate in order to cause queue overflows.

Each node locally logs the packet transmission and reception events, and we collect those events via serial lines in the testbed. The collected local logs are used for obtaining the ground truth of packet losses. Each experiment lasts for 5 h. The above-mentioned losses are randomly injected into the first n nodes ($n \leq 50$) in the duration of the first m hours ($m \leq 5$). We have examined the total number of triggers for the GreenOrbs network during 10 days to be approximately 22 000. Considering that our testbed experiment consists of 50 nodes for 5 h, we set the number of manually injected events to be $2\ 2000 \times \frac{50}{343} \times \frac{5}{10 \times 24} = 66$.

The accuracy of our algorithm is defined to be the percentage of loss events having a correct correlation with its cause. We would expect that with smaller n and m, the accuracy of our algorithm will decrease because there will be multiple causes correlating to a single loss event, resulting in inablity to differentiate the real cause. Fig. 4 shows the accuracy of our algorithm with varying n and m. We have two basic observations from the figure: 1) We indeed see that the accuracy of our algorithm decreases with smaller n and m. 2) If the events are injected in the entire network of 50 nodes for the entire duration of 5 h (i.e.,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

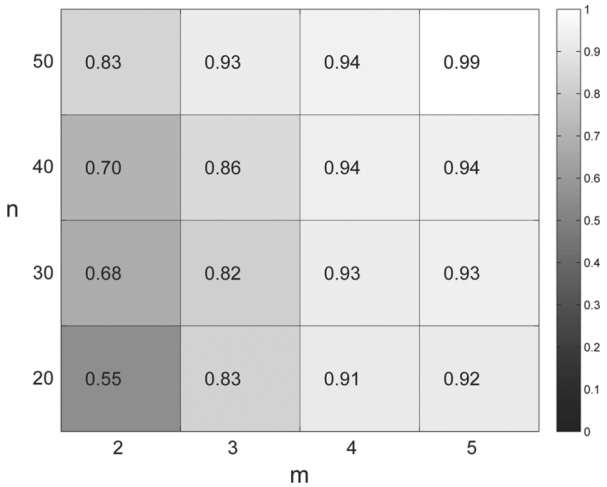


Fig. 4. Accuracy of our correlation algorithm.

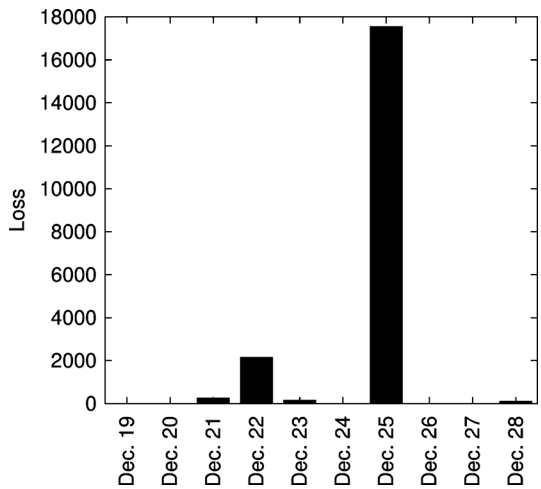


Fig. 5. Sink-side losses for each day.

$n = 50$, $m = 5$), the accuracy of our algorithm is as high as 0.99.

In Sections VI-B–VI-E, we will use our correlation analysis algorithm to investigate different causes of packet losses in the GreenOrbs network.

B. Sink-Side Failures

The sink node receives packets via the wireless radio, and then forwards the packets via the serial port to the PC where a java tool records the collected packets.

By inspecting the `receive` counter in packets originated from the sink node, we are able to detect that 22 873 packets are dropped at the sink side, i.e., either at the sink node or at the PC.

What causes sink-side failures? Fig. 5 plots the number of packet losses at sink side for each day. We observe that a high number of packet losses happened on December 25. Interestingly, we can see that one vertical band covering all nodes on December 25 from Fig. 2.

Checking $A1 \rightarrow A$*:* (Note that the labels A and A1 correspond to triggers/events shown in Fig. 3.) The number of packet losses corresponding to the vertical bands (i.e., V2 in Fig. 2) is 22 638, which means the vertical bands contribute to 99.6% of the sink-side losses. To further investigate the causes of the vertical bands, we examine the status of the sink node. We find that the sink node does not reboot across the event since the packet sequence numbers from the sink node continue to increase. After many rounds of detections during testbed experiments, we find that several possible causes exist: 1) the serial line connecting the sink node and PC is too long and thus is unreliable in delivering packets; 2) the serial port number on the PC unexpectedly changes, causing failure of the java tool; 3) the java tool seems blocked (a restart of the java tool will solve the problem). The above causes are outside the sink node. We collectively call them PC-end failures.

The above result also implies that the amount of packet losses inside the sink node (e.g., due to queue overflow) appears to be small.

C. Corruptions

Corrupted packets are difficult to identify in the first place. We only check a limited packet field (e.g., the ID field, the routing table entries) to validate the correctness.

We are able to detect a total of 9511 corrupted packets that correspond to the same number of corruption triggers. This does not necessarily indicate that 9511 packets are lost because of corruptions. We find that there are 222 corruption triggers that are guaranteed not to cause losses because the subsequent correct packet in the collected packet trace has exactly the same `source` and `seqno` fields with the previous corrupted packet.

Checking $B \rightarrow L$*:* We try to match the corruption triggers to the loss events to find the actual losses caused by corruption. We do not match for the 127 corrupted triggers with broken `source` fields because those triggers cannot be used for spatial correlation and thus may cause a large false positive rate. We set the time lag as 10 min since a smaller time lag will inevitably cause false negatives because our loss detection latency can reach 10 min (i.e., one transmission period). We have detected a total of 9037 corruption-induced losses. This contributes to $9037/181\,862 = 5\%$ of the identified loss. In order to get an estimate of the false positive rate, we consider a sample of the 222 corrupted packets that are guaranteed not to cause losses. The correlation algorithm finds 19 of those packets matched with losses, implying an false positive rate of $19/222 = 8.5\%$.

Checking $D2 \rightarrow B$*:* How do corruptions occur? Does in-network interference cause packet corruptions? To answer these questions, we correlate reboot_3 and loop_3 to the corruption triggers. We find that 3001 corruptions triggers are correlated with either reboot_3 or loop_3, indicating that $3001/9511 = 31.6\%$ corruptions are highly likely to be caused by in-network interference. This implies that the current packet-level CRC mechanism cannot guarantee the correctness of a received packet.

D. Overflow Drops

From Fig. 2, we can get an initial guess that loops can cause overflow drops as there are many overflow drops (green lines) in occurrence with loops (vertical bands covering a subset of nodes).

Checking $C1 \rightarrow C$*:* To take a closer look, we try to correlate loop_1 triggers to overflow loss events. The scope of the trigger

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

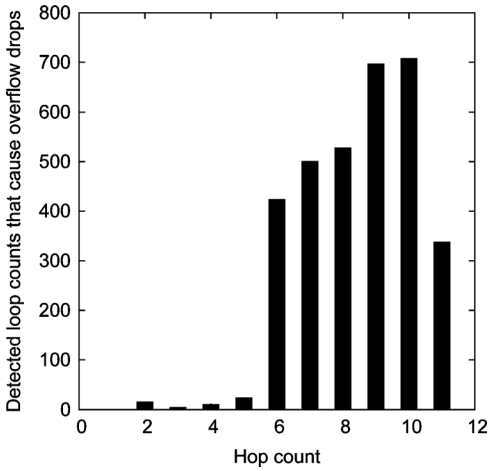


Fig. 6. Average detected number of loops at each hop count.

only includes loop.ID since in this case the nodes experiencing queue overflow should also see the loop events if the overflow drops are caused by the loop. As both the triggers and the loss events are identified using C3 packets, we use a small time lag of one second here. We have matched 399 loop_1 triggers to 322 overflow loss events, containing 5178 losses.

This result implies several facts. First, overflow drops are mainly caused by loops. Loop-induced overflow drops occupy $5178/5219 = 99.2\%$ (note that 5219 is the number of identified overflow losses) of the total identified overflow losses. Second, the non-loop overflow drops only occupy 0.8% of the identified overflow losses. We have manually inspected 15 non-loop overflow events and find that nodes experience non-loop overflows have two characteristics: They either experience a sudden increase in the number of received packets, or they have a high incoming traffic (>300 packets in one period of 10 min). Third, loops do not necessarily cause overflow drops: 93% of the loop events do not cause overflow drops. It is, however, possible that loops may cause other kinds of packet losses, e.g., interference-induced losses.

We are interested in the loop events that cause overflow drops. Where and how do they occur? To get a first impression, Fig. 6 shows the average detected number of loops for nodes at a specified hop count. The hop count of a node is calculated as the median hop count from packets from this node. We observe that nodes far away from the sink can be easily involved into loops that actually cause losses.

E. No-Ack Drops

No-ack drops constitute the largest portion of packet losses. Conceptually, this type of loss is incurred by poor link qualities to the neighboring nodes, causing the retransmission threshold to be exceeded. We consider two kinds of factors impacting the link quality, i.e.: 1) environment-induced link quality degradation, and 2) in-network interference. It is worth noting that environment changes can also cause interference in general. However, in this study, we mainly consider inferences coming from the network itself because of two reasons. First, our network uses channel 15, which does not overlap with any WiFi channels [29]. Second, we use sniffers before deployment and find that the external noise and interference at channel 15 is negligible.

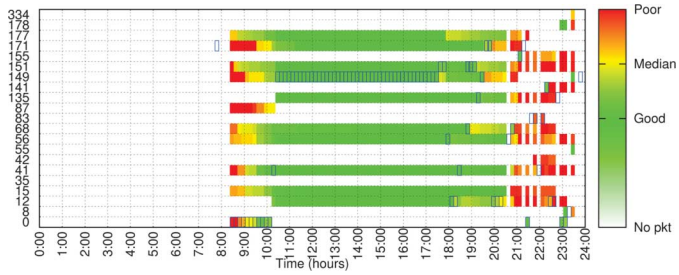


Fig. 7. Node 19's routing table on December 19. IDs in the y-axis show nodes that appear in node 19's routing table, with the color representing the link quality to that neighbor. The parent node of node 19 at a time instant is indicated by a blue rectangle.

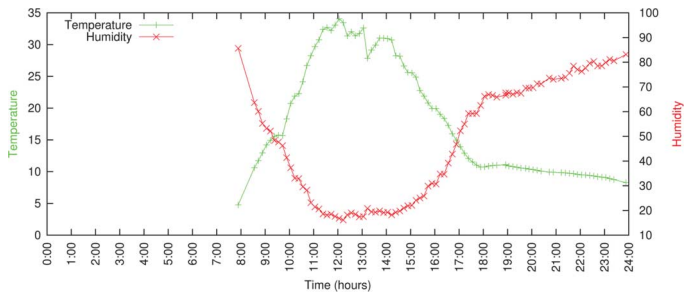


Fig. 8. Node 19's temperature and humidity on December 19.

1) No-Ack Loss Due to Environment-Induced Link Quality Degradation (Env-No-Ack Drops): From Fig. 2, we observe that a number of nodes experience serious packet losses (horizontal bands). Interestingly, those nodes experience recoveries in the midday. This phenomenon is more obvious in the first six days. To investigate the underlying causes, we inspect a representative node, node 19. Fig. 7 shows node 19's routing table on Dec. 19. The x-axis denotes the time, and the y-axis denotes the neighboring nodes that appeared in the routing table at least once during the day. The color represents the link quality to the neighbor. A green color indicates a good link quality, a yellow color indicates a median link quality, and the red color indicates a poor link quality. We also show the parent of node 19 by a blue rectangle. The figure does not show information near the start and end times of the day because no C2 packets from node 19 were received. We can see that the link qualities to all nodes experience an abrupt change during 9:00–10:00 a.m. (increase) and 20:00–21:00 p.m. (decrease). It makes us believe that the environment has a large impact on the link quality.

Therefore, we plot node 19's temperature and humidity on December 19 in Fig. 8. We see that the changes in link qualities seem to indeed be correlated with the environment: The link qualities degrade when the humidity exceeds 70%+. This result indicates that the current routing protocols can be greatly improved by using sensor hints and local buffering mechanisms: env-no-ack losses can be largely mitigated by sending packets when the link condition becomes good in the midday.

Fig. 9 shows the RSSI and link packet reception ratio (PRR) for some neighboring nodes in node 19's routing table on Dec. 19. The y-axis indicates the value of RSSI in dBm. Fig. 9 also

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

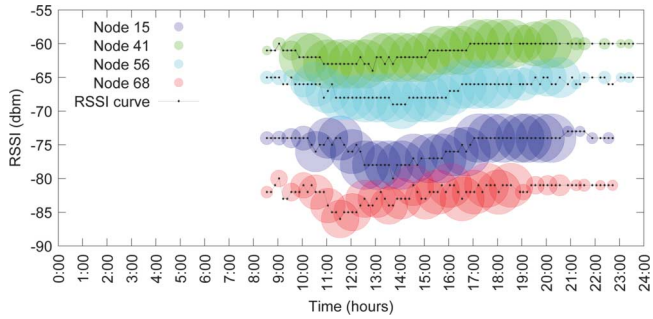


Fig. 9. RSSI and link PRR in node 19's routing table on December 19.

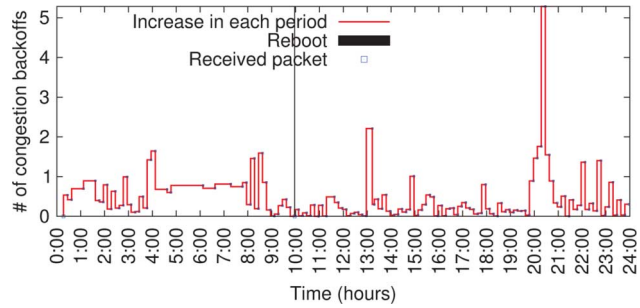


Fig. 10. Average number of congestion backoffs of node 3 on December 23.

plots a circle corresponding to a RSSI value with the radius of the circle indicating the link PRR from node 19 to the corresponding neighbor. A larger circle denotes a high-link PRR. Interestingly, we see that there is a negative correlation between RSSI and PRR, suggesting that the degradation in link quality is not due to channel fading.

While we only show behaviors of a particular node in Figs. 7–9, there are many other nodes included in the horizontal bands in Fig. 2. They exhibit a similar phenomenon. Such phenomena repeated for the whole measurement period. They were not incurred by external interference because the communication channel used by the sensor network did not overlap with any WiFi channels.

We think this problem is related to both the sensor node enclosure and the specific deployment (near a river). The same observation is also reported in a prior study [30] that states that particles and water pooling on the plastic enclosure are likely to alter the radiation patterns, causing link quality degradations. Such an observation is further confirmed by our additional prototype experiments in the field as well as controlled experiments reported in [30], where the authors artificially introduce water/moisture to sensor nodes with plastic enclosure. With more advanced enclosures, we think the problem can be mitigated. For example, in our later deployment of the CitySee project [31] in an urban area, we have replaced the plastic enclosures. We do not observe the same phenomenon.

Checking $D1 \rightarrow D$: We identify env-no-ack losses by identifying nodes that exhibit periodic behaviors in packet losses. There are 68 444 env-no-ack losses, i.e., 37.6% of the total identified losses.

2) No-Ack Loss Due to In-Network Interference (Interference-No-Ack Drops): Interference has a large impact on the performance of wireless links. From the CTP/LPL implementation, we know that both the reboot and loop events can cause a high beaconing rate since the Trickle timer will be reset to its minimum interval of 128 ms. With LPL, interference will be severe because of long preambles in packet transmissions.

To cross-validate the high beaconing rate after reboot and loop events, we plot node 3's average number of congestion backoffs (i.e., the number of congestion backoffs divided by the number of initial backoffs during the period) on December 23 in Fig. 10. The average number of backoffs roughly reflects how the channel is contended at that time. We see from Fig. 10 that the value experiences an increase around 13:00 p.m. and during 20:00–21:00 p.m. By examining the system events, we find that there are three reboot events around 13:00 p.m., and a loop is detected during 20:00–21:00 p.m. A high contention suggests a high probability of interference.

TABLE II
ROOT CAUSES OF IDENTIFIED LOSSES

Root cause	%
A. sink-side failure	12.5%
A1. PC-end failure	12.45%
A2. sink node loss	0.05%
B. corruption	5%
C. overflow drops	2.87%
C1. loop overflow drops	2.85%
C2. non-loop overflow drops	0.02%
D. no-ack drops	46.2%
D1. env-no-ack drops	37.6%
D2. interference-no-ack drops	2.4%
E. reboot (direct impact on loss)	~0

In our current study, we consider interference caused by reboots and loops. We do not consider interference caused by data packet transmissions because the exact timing of data packet transmissions is left unknown.

Checking $D2 \rightarrow D$: We try to correlate reboot_3 and loop_3 triggers to no-ack loss events in order to investigate interference-induced no-ack loss. The scope of the reboot and loop triggers are set to be the neighboring nodes that are interfered by the corresponding triggers. We have matched 247 loop events to 536 no-ack loss events, containing 4361 losses. We have also found that 10 no-ack losses are matched with reboot events with a time lag of 10 min. No loss event is found to be correlated to both reboots and loops. Therefore, the total number of losses correlated with interference is at least 4371, occupying 2.4% of the identified losses.

F. Summary

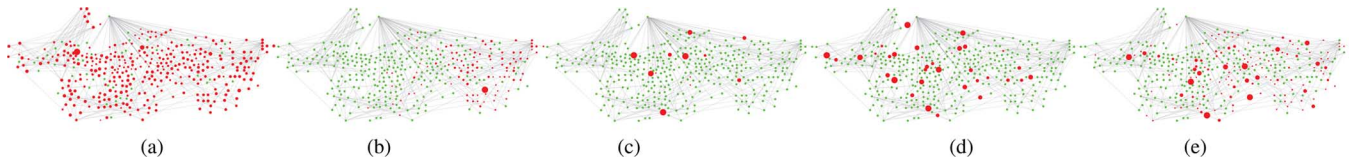
We give a summary about the root causes we have found so far. Table II gives the root causes and the percentage of identified losses they induce.

G. Understanding the Loss

We look at the characteristics of five important losses, i.e., corruption-induced loss, loop overflow drops, non-loop overflow drops, env-no-ack drops, and interference-no-ack drops.

Spatial Distribution: Fig. 11(a)–(e) shows the geographic distributions of five categories of losses. Interestingly, we see different spatial distributions of packet losses caused by different events. Fig. 12(a)–(e) shows the spatial distributions of five categories of losses with respect to the median hop count of each

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



(a) (b) (c) (d) (e)

Fig. 11. Geographic distribution of losses. (a) Corruption. (b) Loop overflow drops. (c) Non-loop overflow drops. (d) Env-no-ack drops. (e) Interference-no-ack drops.

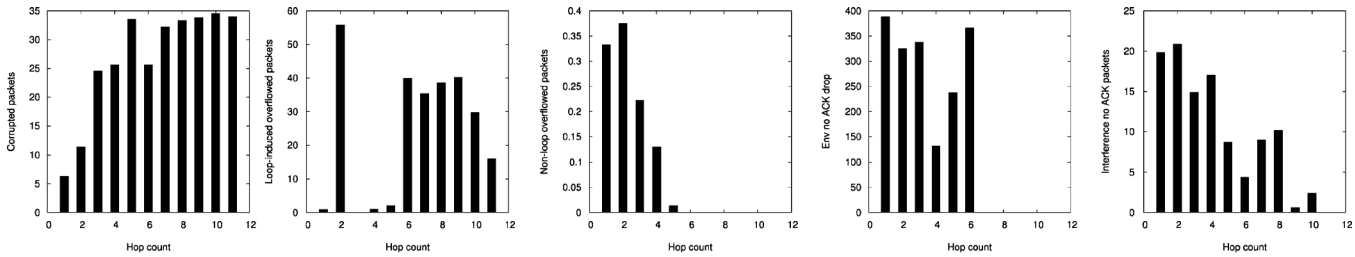


Fig. 12. Losses versus hops. (a) Corruption. (b) Loop overflow drops. (c) Non-loop overflow drops. (d) Env-no-ack drops. (e) Interference-no-ack drops.

node. Fig. 12(a) shows that while corruption-induced losses increase for the first five hops, it is not apparent for larger hops. Fig. 12(b) shows that loop overflow drops mainly occur at nodes with larger hop counts where routing loops can occur more easily. The large value for hop 2 is caused by a single node 576 involved in loops (which drops 1000+ packets because of overflow). Fig. 12(c) shows that non-loop overflows mainly occur at nodes near the sink. It is reasonable because those nodes are supposed to carry a higher traffic volume. Fig. 12(d) shows that env-no-ack drops occur at nodes near the sink. We suspect that it is related to our specific deployment where those nodes are close to a river. Fig. 12(e) shows that interference-no-ack drops mainly occur in nodes near the sink because of high traffic load.

VII. Implications and Lessons Learned

In this section, we give a summary on the observations, implications, and lessons learned from this study.

Observation 1: The overall delivery performance of our system is 81.3% with a radio duty cycle of 4.9%. Although the performance of old versions of TinyOS protocols is reported in prior works, the latest TinyOS protocol suite (version 2.1.1) seems not well evaluated in outdoor deployments. To our knowledge, we are the first to report quantitatively the performance of latest TinyOS protocol suite in a large-scale outdoor deployment.

Implication 1: Compared to tailored protocols that are reported to achieve 99.9% reliability at permilli in small-scale networks [27], [32], the current ready-to-use TinyOS protocols are not good enough in large networks.

Observation 2: The number of packet losses on long path is no higher than that on short path. For nodes near the sink, the PDRs have a high variance while for nodes far away from the sink, the PDRs are mainly concentrated around 80%.

Implication 2: This implies the existence of some bad links since otherwise all PDRs will be high. Those bad links do not impact nodes far away from the sink (distant nodes) but do impact nodes near the sink (nearby nodes). This further implies that the current routing metric can avoid the selection of bad links for distant nodes but cannot optimize the delivery performance for nearby nodes by utilizing a longer and more stable path.

Observation 3: For our deployment, sink-side failures are mainly incurred at the PC end instead of the sink node.

Implication 3: PC-end hardware and software should be closely monitored to minimize packet losses. The use of multiple sinks (including sink node and PC) will be effective in improving the reception reliability.

Observation 4: Packet corruption rates are relatively high (at least 5%). Packets can be corrupted in the network during transmission.

Implication 4: The current packet-level CRC mechanism is not enough to ensure the correctness of a received packet.

Observation 5: Overflow drops are mainly caused by routing loops whereas most loops are transient and show no strong correlation with packet losses.

Implication 5: Routing loops have different impacts on packet delivery performance. On one hand, it decreases the performance because of queue overflow (and interference). On the other hand, it can salvage transient packet losses. With respect to packet delivery performance, we should eliminate loops that cause overflow drops.

Observation 6: It is possible that there is a negative correlation between RSSI and PRR.

Implication 6: Link estimation protocols should use multiple factors to decide the link quality.

Observation 7: The environment has a large impact on packet delivery performance. A number of nodes exhibit highly periodic performance variations because many links severely degrade in the night. However, most of the nodes experience recoveries in the midday. We think this problem is related to both the sensor node enclosure and the specific deployment (near a river). With more advanced enclosures, the problem can be mitigated.

Implication 7: This result indicates that the current routing protocols can be greatly improved by using sensor hints and local buffering mechanisms: Packet losses can be largely

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

mitigated by sending packets when the link condition becomes good in the midday.

Observation 8: We find in our deployment an unnegligible number of node reboots and node failures. It also appears that the poor performance of some wireless links are highly related to our specific deployment where those links are near a river.

Implication 8: Both sensor node hardware and sensor network deployment have great impacts on the system performance. It is suggested that multiple rounds of indoor testbed experiments and outdoor prototype experiments are conducted before a large-scale and long-term sensor network is deployed.

VIII. Discussions

Our current data sources are collected in the form of data packets and are relatively easy to retrieve. Nevertheless, they cannot capture the complete set of system events in the network, and the additional transmissions of measurement data can affect the network performance. Despite the potential negative impact, it can reveal valuable information about the network, which greatly improves the effectiveness of network management. We also carefully study the network performance with/without additional measurement data in a 50-node testbed and find that the network performance in terms of PDR and radio duty cycle is not significantly affected. For some wireless behaviors, such as channel utilization, MAC efficiency and additional measurement infrastructures such as passive sniffing or local logging are required. We will explore those approaches as future work.

Our current study is based on a data collection network employing the TinyOS protocol suite. There are circumstances in which developers employ customized protocol suites (such as receiver-initiated MACs or backpressure routing) in specific deployments. However, our analysis using spatial-temporal correlation is general and can also be well applied in other areas. The causal graph and specific parameters to perform correlation analysis need to be slightly modified.

The benefit of using spatial-temporal correlation analysis is to *quantify* the impacts of different factors so that developers and network operators can focus on the bottleneck factors to improve the network performance. We see that a number of recent works are very relevant to our measurement results, implying that our findings in this study are potentially very useful to real sensor network designs and implementations (as discussed below).

We have mentioned that routing loops indeed occur in the current CTP protocol. Backpressure routing [33], [34] is a promising mechanism that mitigates routing loops and promises throughput-optimal performance. Unlike traditional routing mechanisms for wired and wireless networks, backpressure routing does not perform any explicit path computation from source to destination. Instead, the routing and forwarding decision is made independently for each packet by computing for each outgoing link a backpressure weight that is a function of localized queue and link-state information.

We have mentioned that both reboots and loops incur a high beaconing rate that introduces negative impacts to the network, especially for asynchronous low-power listening MACs. Broadcast-Free Collection Protocol, BFC [35], is a protocol to eliminate broadcast completely. We envision that the use of BFC can significantly reduce the chances of interference-induced losses.

We have also mentioned that environment can cause link quality degradations and the use of sensor hints and local buffering can greatly improve protocol performance. Ravindranath *et al.* [36] propose a network architecture that uses sensor hints to augment and improve wireless protocols. It is implemented on commodity smartphones and tablet devices equipped with a variety of sensors that can provide hints about the device's mobility state and its operating environment. We believe the same technique can be applied in existing sensor networks to improve the wireless network performance.

IX. Conclusion and Future Work

In this paper, we present MAP, a methodology for measuring and analyzing the packet delivery performance of a large operating WSN in the wild. Based on the collected data, we present an approach for uncovering the spatial-temporal distributions of the loss events as well as developing a causal graph with which we perform spatial-temporal correlation analysis for revealing the root causes. We summarize implications and lessons learned and give important guidance to future WSN deployments.

There are multiple dimensions to explore. First, we would like to examine more numbers of system events. Second, we would like to implement our methodology as a real-time service, augmented with limited use of passive sniffing or local logging for deep examination of wireless behaviors.

References

[1] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proc. ACM SenSys*, 2003, pp. 1–13.

[2] S. Bapat, V. Kulathumani, and A. Arora, "Analyzing the yield of ExScal, a large-scale wireless sensor network experiment," in *Proc. IEEE ICNP*, 2005.

[3] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, and J. Zhao, "Does wireless sensor network scale? A measurement study on GreenOrbs," in *Proc. IEEE INFOCOM*, 2011, pp. 873–881.

[4] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "Understanding the causes of packet delivery success and failure in dense wireless sensor networks," Stanford Univ., Stanford, CA, USA, Tech. Rep. SING-06-00, 2006.

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

[6] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *Proc. EWSN*, 2005, pp. 121–132.

[7] R. Szewczyk, J. Polastre, A. Mainwaring, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proc. ACM SenSys*, 2004, pp. 214–226.

[8] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," in *Proc. ACM SenSys*, 2005, pp. 51–63.

[9] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor networks," in *Proc. USENIX OSDI*, 2006, pp. 381–396.

[10] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. F. Abdelzaher, "Achieving long-term surveillance in VigilNet," *Trans. Sensor Netw.*, vol. 5, no. 1, pp. 1–39, 2009.

[11] A. Arora *et al.*, "ExScal: Elements of an extreme scale wireless sensor network," in *Proc. IEEE RTCSA*, 2005.

[12] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Out-of-the-box environmental monitoring," in *Proc. ACM/IEEE IPSN*, 2008, pp. 332–343.

[13] M. Keller, M. Woehrle, R. Lim, J. Beutel, and L. Thiele, "Comparative performance analysis of the permadozer protocol in diverse deployments," in *Proc. IEEE LCN*, 2011, pp. 957–965.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[14] O. Chipara, C. Lu, T. Bailey, and G.-C. Roman, "Reliable clinical monitoring using wireless sensor networks: Experience in a step-down hospital unit," in *Proc. ACM SenSys*, 2010, pp. 155–168.

[15] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *Proc. ACM SIGCOMM*, 2004, pp. 121–132.

[16] A. Natarajan, B. de Silva, K. K. Yap, and M. Motani, "Link layer behavior of body area networks at 2.4 GHz," in *Proc. ACM MobiCom*, 2010, pp. 241–252.

[17] R. Maheshwari, S. Jain, and S. R. Das, "A measurement study of interference modeling and scheduling in low-power wireless networks," in *Proc. ACM SenSys*, 2008, pp. 141–154.

[18] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Analyzing the MAC-level behavior of wireless networks," in *Proc. ACM SIGCOMM*, 2006, pp. 75–86.

[19] B. R. Chen, G. Peterson, G. Mainland, and M. Welsh, "LiveNet: Using passive monitoring to reconstruct sensor network dynamics," in *Proc. DCOSS*, 2008, pp. 79–98.

[20] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush, "A measurement study on the impact of routing events on end-to-end internet path performance," in *Proc. ACM SIGCOMM*, 2006, pp. 375–386.

[21] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California fault lines: Understanding the causes and impact of network failures," in *Proc. ACM SIGCOMM*, 2010, pp. 315–326.

[22] A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons, "Detecting the performance impact of upgrades in large operational networks," in *Proc. ACM SIGCOMM*, 2010, pp. 303–314.

[23] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. ACM/IEEE IPSN*, 2005, Art. no. 48.

[24] "TinyOS," [Online]. Available: http://www.tinyos.net

[25] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X.-Y. Li, and G. Dai, "Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest," in *Proc. ACM SenSys*, 2009, pp. 99–112.

[26] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. ACM MobiCom*, 2003, pp. 134–146.

[27] R. Musaloiu-E, C.-J. M. Liang, and A. Terzis, "Koala: Ultra-low power data retrieval in wireless sensor networks," in *Proc. ACM/IEEE IPSN*, 2008, pp. 421–432.

[28] 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.

[29] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi interference in low power ZigBee networks," in *Proc. ACM SenSys*, 2010, pp. 309–322.

[30] A. Markham, N. Trigoni, and S. Ellwood, "Effect of rainfall on link quality in an outdoor forest deployment," in *Proc. WinSys*, 2010, pp. 1–6.

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

[32] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: Ultra-low power data gathering in sensor networks," in *Proc. ACM/IEEE IPSN*, 2007, pp. 450–459.

[33] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without Routes: The backpressure collection procotol," in *Proc. ACM/IEEE IPSN*, 2010, pp. 279–290.

[34] R. Laufer, T. Salonidis, H. Lundgren, and P. L. Guyadec, "XPRESS: A cross-layer backpressure architecture for wireless multi-hop networks," in *Proc. ACM MobiCom*, 2011, pp. 49–60.

[35] D. Puccinelli, S. Giordano, M. Zuniga, and P. Jose, "Broadcast-free collection protocol," in *Proc. ACM SenSys*, 2012, pp. 29–42.

[36] L. Ravindranath, C. Newport, H. Balakrishnan, and S. Madden, "Improving wireless network performance using sensor hints," in *Proc. USENIX NSDI*, 2011, pp. 281–294.

Wei Dong (S'08–M'12) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 2005 and 2011, respectively.

He is currently an Associate Professor with the College of Computer Science, Zhejiang University. His research interests include networked embedded systems and wireless sensor networks.

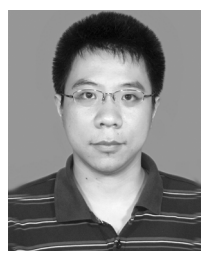
Yunhao Liu (M'04–SM'06) received the B.S. degree in automation from Tsinghua University, Beijing, China, in 1995, and the M.S. and Ph.D. degrees in computer science and engineering from Michigan State University, East Lansing, MI, USA, in 2003 and 2004, respectively.

He is a Professor, Dean of the School of Software, and a member of the Tsinghua National Lab for Information Science and Technology, Tsinghua University.

Prof. Liu is serving as the Chair of ACM China Council.

Yuan He (S'09–M'12) received the B.S. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2003, the M.S. degree in computer software and theory from the Chinese Academy of Sciences, Beijing, China, in 2006, and the Ph.D. degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2010.

He is a member of the Tsinghua National Lab for Information Science and Technology, Beijing, China. His research interests include wireless and sensor networks, mobile computing, and peer-to-peer computing.



Tong Zhu (M'13) received the B.S. degree in computer science from Nanjing University, Nanjing, China, in 2008, and the Ph.D. degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2013.

He is a member of the Tsinghua National Lab for Information Science and Technology, Beijing, China. His research interests include wireless and sensor networks and mobile computing.

Chun Chen (M'08) received the Bachelor of Mathematics degree from Xiamen University, Xiamen, China, in 1981, and the M.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1984 and 1990, respectively.

He is a Professor with the College of Computer Science and the Director of the Institute of Computer Software, Zhejiang University. His research interests include embedded system, image processing, computer vision, and CAD/CAM.