On the Delay Performance in a Large-Scale Wireless Sensor Network: Measurement, Analysis, and Implications

Jiliang Wang, *Member, IEEE*, Wei Dong, *Member, IEEE*, Zhichao Cao, *Member, IEEE*, and Yunhao Liu, *Senior Member, IEEE*

***Abstract*—We present a comprehensive delay performance measurement and analysis in a large-scale wireless sensor network. We build a lightweight delay measurement system and present a robust method to calculate the per-packet delay. We show that the method can identify incorrect delays and recover them with a bounded error. Through analysis of delay and other system metrics, we seek to answer the following fundamental questions: What are the spatial and temporal characteristics of delay performance in a real network? What are the most important impacting factors, and is there any practical model to capture those factors? What are the implications to protocol designs? In this paper, we identify important factors from the data trace and show that the important factors are not necessarily the same with those in the Internet. Furthermore, we propose a delay model to capture those factors. We revisit several prevalent protocol designs such as Collection Tree Protocol, opportunistic routing, and Dynamic Switching-based Forwarding and show that our model and analysis are useful to practical protocol designs.**

***Index Terms*—Delay measurement, impacting factor, large-scale, wireless sensor networks.**

I. Introduction

RECENT advances in wireless sensor networks (WSNs) have fostered a large number of applications, such as structural protection and health-monitoring WSNs [1], [2], etc. Those WSN applications often require quality-of-service (QoS) guarantees to fulfill the system requirements, e.g., real-time data delivery. Of the major factors that affect system QoS, delay is an important one.

There are many research works on delay analysis and measurement. Kompella *et al.* [3] present a fine-grained latency measurement method in presence of packet losses for the Internet. Wilson *et al.* [4] present delay analysis results in data centers, providing guidelines to practical data center design. There are tremendous research efforts made to delay analysis and modeling in WSNs. For example, probabilistic delay bounds are presented in [5]–[8] by extending network calculus. Furthermore, stochastic delay models are proposed by combining real-time theory and queuing theory [9]–[11] or applying Discrete Markov Process [12]. There are also some empirical network delay models such as [13] and [14].

While there are excellent research works for WSNs, Internet, and data centers, a practical end-to-end delay performance measurement and analysis in an operational large-scale WSN is still missing. On the other hand, considering the emerging demand of WSN applications, it is important to understand the delay performance in practical large-scale networks.

Delay performance measurement and analysis, in a large-scale WSN, face nontrivial challenges. First, different from the Internet and data centers [3], [4], [15], there are no effective methods in WSNs supporting per-packet delay measurement. Traditional delay measurement methods rely on network synchronization, which introduce additional overhead. Thus, it is not always efficient to apply network synchronization to an operational network. Second, analyzing the collected information is challenging. Collecting all required information from the network incurs a high network overhead. Thus, the information is usually incomplete due to resource constraints for sensor nodes and packet losses in the network. Meanwhile, there are various performance metrics in the network, and a single delay change may be accompanied by variation of multiple metrics. Moreover, with low power listening, each node switches between wake-up and sleep states to save energy. The delays exhibit intrinsic randomness in its distribution, introducing difficulty to efficient and automatic analysis.

In this paper, we build an infrastructure for delay measurement in CitySee, a large-scale WSN consisting of 1200 nodes. The infrastructure does not rely on network synchronization and thus does not introduce additional overhead. We present basic statistical characteristics based on the collected data. To systematically and automatically identify important impacting factors from various parameters, we build a method based on Rulefit for the collected data trace. Furthermore, we quantitatively calculate the correlation between different impacting factors and the delay performance. Based on those important factors, we build a practical delay model and validate the model using the collected data trace. Finally, we revisit three important protocols based on

Manuscript received October 10, 2012; revised July 01, 2013 and November 04, 2013; accepted November 26, 2013; approved by IEEE/ACM Transactions on Networking R. Mahajan. Date of publication January 15, 2014; date of current version February 12, 2015. This work was supported in part by the NSFC under Grants 61202359, 61373166, and 61202402; the NSFC Major Program 61190110; the National High-Tech R&D Program of China (863) under Grant 2011AA010100, and the Research Fund for the Doctoral Program of Higher Education of China under Grant 20120101120179. The preliminary version of this work was published at the IEEE Real-Time Systems Symposium (RTSS), San Juan, Puerto Rico, December 4–7, 2012.

J. Wang, Z. Cao, and Y. Liu are with the School of Software and TNLIST, Tsinghua University, Beijing 100084, China (e-mail: jiliang@greenorbs.com; caozc@greenorbs.com; yunhao@greenorbs.com).

W. Dong is with the College of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: dongw@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.2013.2296331

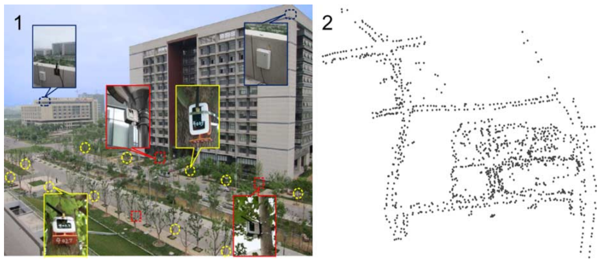


Fig. 1. Deployment and sensor nodes in CitySee: 1) overview of the deployment area; 2) node locations in the network.

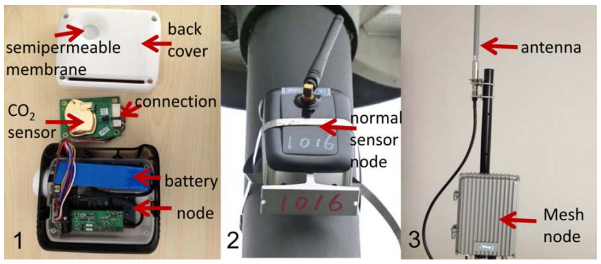


Fig. 2. Sensor nodes in CitySee: 1) CO_2 sensor node, 2) normal sensor node, and 3) mesh node.

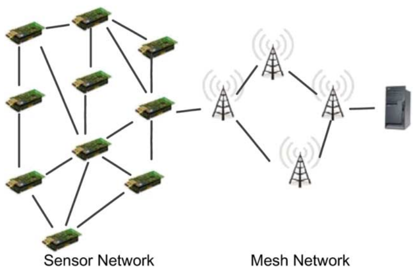


Fig. 3. Overview of network architecture.

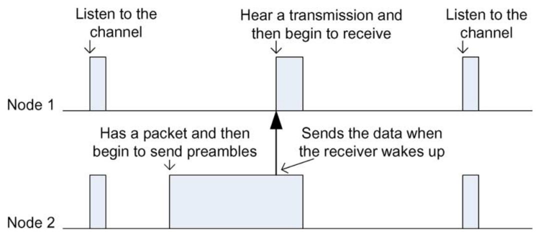

Listen to the channel
Hear a transmission and then begin to receive
Listen to the channel
Node 1
Has a packet and then begin to send preambles
Sends the data when the receiver wakes up
Node 2

Fig. 4. Basic mechanism of LPL protocols.

the measurement results and propose a practical delay model. In summary, the contributions of this paper are as follows.

- We build a measurement infrastructure in an operational large-scale WSN with little network overhead. Based on the collected data, we present the spatial and temporal characteristics of delay performance.
- We present an automatic method based on Rulefit [16] to identify important impacting factors to the delay performance.
- We propose a practical model and validate it with the collected data. We show the implications to protocol designs.

The rest of the paper is organized as follows. Section II presents the system overview. Section III shows the delay measurement method. Section IV presents the delay distribution overview. Section V introduces the method of identifying important factors. Section VI builds a delay model based on the analysis. Section VII shows the implications of the analysis and the evaluation results. Section VIII introduces the related work. Finally, Section IX concludes this work.

II. System Overview

A. Network

The primary goal of CitySee is to precisely measure CO_2 emissions in a citywide area. We started the project since July 2011. Fig. 1 shows an overview of the network. Totally, we have deployed 1200 nodes. The network employs a tiered architecture with three kinds of nodes, i.e., normal telosB nodes, CO_2 nodes, and mesh nodes.

In the network, each normal sensor node reads the sensing data and records system status. CO_2 nodes can also read the CO_2 concentration. The CO_2 sensing component is connected to the main board through the UART connection. The normal nodes and CO_2 nodes form a network and deliver their data to a sink node (normal node) in the network. Fig. 2 shows the sensor nodes in our network.

The mesh nodes have a high bandwidth of several megabytes per second and a long transmission distance. They comprise the network backbone. Sink nodes of different subnets are connected to the network backbone in order to deliver packets to the base station. Fig. 3 illustrates the overview of the system architecture.

B. Protocols

1) Low Power Listening: Low power listening (LPL) is widely adopted in WSNs to save energy. In LPL, each node switches between awake and sleep state to save energy. Most LPL protocols share the similar principle as shown in Fig. 4. Each node samples the channel for a short duration in each cycle. If energy is detected, the node stays awake for another short duration to receive packets. Otherwise, the node turns off the radio, and in the next cycle (e.g., 500 ms later) resamples the channel. To transmit a packet, the sender continues sending packets as preambles until the receiver wakes up. For broadcast, the preamble lasts for a cycle duration in order to ensure that all neighboring nodes wake up once. For unicast with link-layer ACK, the sender can stop the preambles until an ACK is received or the end of a cycle. Another type of LPL protocol is receiver-initiated low-duty cycle protocol. Each node periodically wakes up and sends probe packets to see if there are transmissions intended for it. If a node has packets to send, it will keep awake and send the packets once receiving a probe packet from the receiver. Since a sender may begin to send packets at any time, the time the sender needs to wait is randomly distributed in the cycle. This introduces randomness to packet delay.

2) Collection Tree Protocol: Collection Tree Protocol (CTP) [17] is used to build a routing tree in the network. CTP adopts the ETX metric [18], the expected transmission count, as the path quality metric. Each node selects a path with minimum ETX. The ETX of a link is calculated as $1/q$, where q is the packet reception ratio. The path ETX is calculated as the sum of all link ETXs along the path.

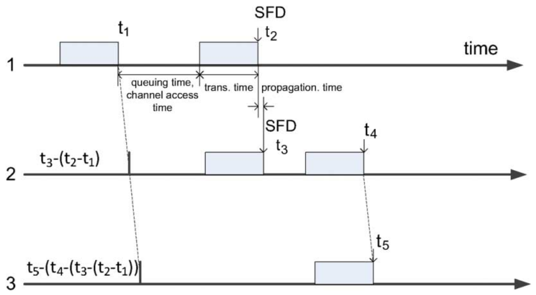


Fig. 5. Measuring delay in the network.

C. Measurement Infrastructure

Time synchronization can be used to measure delay in the network. However, time synchronization protocols in WSNs, such as FTSP [19], etc. [20]–[22], incur additional traffic overhead into the network in order to maintain a global timestamp for all nodes.

We use a lightweight approach to measure the end-to-end packet delay without incurring synchronization traffic. For each packet, we define the delay as the time from the packet is generated at the source node to the time that the packet is received at the sink node. As shown in Fig. 5, the delay for a packet on a path mainly consists of the following parts: 1) the packet transmission time for a packet, i.e., the time used to modulate the packet to signal; 2) the channel accessing time, i.e., the time used to contend for the channel, including the backoff time; 3) the queuing time; and 4) the propagation time on each hop, i.e., the traveling time for the signal from the sender to the receiver, which can be calculated as the distance between the sender and receiver divided by the light speed.

Our approach is based on the MAC-layer time-stamping technique (MLT) [19]. In MLT, a packet is time-stamped just before the first byte is transmitted after backoff (with respect to the sender's clock) and just before the first byte is received (with respect to the receiver's clock).

Our approach measures the end-to-end delay as follows. Fig. 5 shows an example of three nodes. Assuming a packet is generated at node 1's local time t_1, we show how to measure the generation time to node 2's and node 3's clock. Suppose the packet is transmitted and time-stamped at t_2, with the time t_1 contained in the packet. Here, node 1 performs backoffs during the time $[t_1, t_2]$. Thus, at time t_2, the packet is modulated and emitted through the antenna. Then, the packet is received at node 2 and time-stamped at t_3 by MLT. Since the distance between two nodes is usually several hundreds of meters, we ignore the signal propagation time from node 1 to node 2. Node 2 calculates the packet generation time with respect to node 2's clock by subtracting the time difference $t_2 - t_1$ from time t_3, i.e., $t_3 - (t_2 - t_1)$. Intuitively, it seems that the packet is generated at time $t_3 - (t_2 - t_1)$ to node 2's clock.

Here, we denote the time $t_2 - t_1$ as the waiting time (t_{w}) at node 1. This is calculated as the time a packet is transmitted minus the time the packet is received at node 1. Similarly, the sink node, i.e., node 3, can calculate the packet generation time after receiving a packet from node 2 as $t_5-(t_4-(t_3-(t_2-t_1)))$, and the receiving time as t_5, both are with respect to node 3's clock. Then, the delay of the packet is calculated as $t_4 - (t_3 - (t_2 - t_1))$.

D. Collected Data

The network collects four types of packets, denoted as *C1*, *C2*, *C3*, and *C4*, respectively. Each node sends each type of packet in every 10 min. There is a common header with four fields, which records a common sequence number and the timestamps used for delay measurement. We will introduce in Section III how to use the timestamps for delay measurement.

III. Delay Measurement

In this section, we show how to derive the delay for each packet.

A. Delay Measurement Analysis

We assume the time t provided by a sensor node follows the linear clock model [19], i.e.,

$$t = (1+\alpha)(\tau - t_{\mathrm{o}}) \tag{1}$$

where τ is the time provided by a perfect clock, t_{o} is the offset, and α is the clock drift bounded by $[-\hat{\alpha}, \hat{\alpha}]$. Each packet from a node has an unique increasing sequence number i. For packet i, we denote the receiving time of the packet i on node j with respect to the perfect clock as $t_{\mathrm{r}}(i,j)$ and the transmitting time of the packet i on node j as $t_{\mathrm{x}}(i,j)$. The transmitting/receiving time is the time just before the first byte of a packet is transmitted/received. We denote O as the source node and D as the sink node for a path. Thus, $t_{\mathrm{r}}(i,O)$ is the generation time of packet i on the source node. Therefore, the end-to-end delay of packet i for a path can be calculated as

$$t_{\mathrm{d}}(i) = t_{\mathrm{r}}(i,D) - t_{\mathrm{r}}(i,O). \tag{2}$$

Denote the waiting time for packet i at node j on the path as $t_{\mathrm{w}}(i,j)$. We have $t_{\mathrm{w}}(i,j) = t_{\mathrm{x}}(i,j) - t_{\mathrm{r}}(i,j)$. The waiting time $t_{\mathrm{w}}(i,j)$ contains the backoff time on node i to contend for the channel. We ignore the signal propagation time in our analysis considering the distance between the sender and receiver is small. Therefore, the end-to-end delay is also calculated as

$$t_{\mathrm{d}}(i) = \sum_{j=1}^{n-1} t_{\mathrm{w}}(i,j). \tag{3}$$

In practical networks, we denote $\tilde{t}_X(i,j,k)$ the measured time of operation X for packet i on node j with respect to k's clock. We can measure the following:

- the SourceTime $\tilde{t}_{\mathrm{r}}(i,O,O)$, i.e. the generation time of packet i to the source's clock;
- the SinkTime $\tilde{t}_{\mathrm{r}}(i,D,D)$, i.e. the receiving time of packet i to the sink's clock;
- the waiting time $\tilde{t}_{\mathrm{w}}(i,j,j)$ for packet i on node j on a path;
- the SourceTimeAtSink as $\tilde{t}_{\mathrm{a}}(i) = \tilde{t}_{\mathrm{r}}(i,D,D) - \sum_{j=1}^{n-1} \tilde{t}_{\mathrm{w}}(i,j,j)$.

We summarize the parameters in Table I.

For two packets i_1 and i_2, we denote the measured time difference on node j as $\Delta\tilde{t}_X(i_1,i_2,j) = \tilde{t}_X(i_1,j,j) - \tilde{t}_X(i_2,j,j)$ and the time difference with respect to the perfect clock as $\Delta t_X(i_1,i_2,j) = t_X(i_1,j) - t_X(i_2,j)$. Denote $\Delta\tilde{t}_{\mathrm{a}} = \tilde{t}_{\mathrm{a}}(i_1) - \tilde{t}_{\mathrm{a}}(i_2)$, we have the following theorem.

Theorem 1 (Drift Constraint): For two packets i_1 and i_2 $(i_1 > i_2)$, the measured SourceTimeAtSink satisfies

$$(1-2\hat{\alpha})\Delta\tilde{t}_{\mathrm{r}}(i_1,i_2,O) \le \Delta\tilde{t}_{\mathrm{a}} \le (1+2\hat{\alpha})\Delta\tilde{t}_{\mathrm{r}}(i_1,i_2,O). \tag{4}$$

TABLE I
NOTATIONS

parameter	description
$\hat{\alpha}$	The maximum clock drift.
O	The source node of a path.
D	The sink node of a path.
$t_X(i,j)$	The time of operation X for packet i on node j with respect to the perfect clock. X={x, r} denotes the time the packet is transmitted and the time the packet is received. X=w denotes the waiting time. On the source node, we use $t_r(i,O)$ to denote the packet generation time.
$t_d(i)$	The end-to-end delay for packet i. Thus we have $t_d(i) = t_r(i,D) - t_r(i,O)$. We also have $t_d(i) = \sum_{j=1}^{n-1} t_w(i,j)$ for a path consisting of n nodes.
$\tilde{t}_X(i,j,k)$	The measured time of operation X for packet i on node j with respect to k's clock.
$\tilde{t}_a(i)$	The measured packet generation time for packet i with respect to the sink's clock.

Proof: See Appendix A. □

The intuition of the drift constraint theorem is as follows. Due to a maximum drift of $\hat{\alpha}$ for each node, the maximum relative drift between the SourceTimeAtSink and SourceTime should be less than $2\hat{\alpha}$. We can check the correctness of the measured timestamps in the received packets and thus filter the incorrect timestamps. Denote the calculated delay as

$$\tilde{t}_{\mathrm{d}}(i) = \tilde{t}_{\mathrm{r}}(i, D, D) - \tilde{t}_{\mathrm{a}}(i) \tag{5}$$

we have the following theorem.

Theorem 2 (Delay Error Bound): The calculated delay $\tilde{t}_{\mathrm{d}}(i)$ satisfies

$$(1-\hat{\alpha})t_{\mathrm{d}}(i) \le \tilde{t}_{\mathrm{d}}(i) \le (1+\hat{\alpha})t_{\mathrm{d}}(i). \tag{6}$$

Proof: See Appendix B. □

This theorem shows that the calculated delay has a bounded error. Intuitively, the delay error is introduced by measuring the waiting time on each node. As the delay $t_{\mathrm{d}}(i)$ is small, the error is also very small. This is different from time synchronization protocols that need to maintain a global clock all the time with periodical beacon packets.

Furthermore, for two packets i_1 and i_2 $(i_1 > i_2)$, we assume the timestamp of SourceTimeAtSink in packet i_1 is incorrect. We can use the correct timestamps in the packet i_2 to recover the i_1. We calculate the delay of packet i_1 by

$$t'_{\mathrm{d}}(i_1) = \tilde{t}_{\mathrm{r}}(i_1, D, D) - \left(\tilde{t}_{\mathrm{r}}(i_1, O, O) - \left(\tilde{t}_{\mathrm{r}}(i_2, O, O) - \tilde{t}_{\mathrm{a}}(i_2)\right)\right). \tag{7}$$

It can also be shown that the error of the recovered delay is proportional to the delay $t_{\mathrm{d}}(i_1)$ and the generation time difference between two packets.

B. Delay Processing

The data processing consists of the following steps.

1) We first calculate $\tilde{t}_{\mathrm{d}} = \tilde{t}_{\mathrm{r}}(i, D) - \tilde{t}_{\mathrm{a}}(i)$. The calculated $\tilde{t}_{\mathrm{d}}$ is shown in Step 1 in Fig. 6.
2) There exist various types of errors in the data. The first type of error comes from MLT. Due to packet overflow in the limited receiving buffer and packet losses, MLT cannot guarantee to provide correct stamps. To address this problem, we first validate the delay values and exclude the incorrect timestamps by Theorem 1. We group the delays satisfying the drift constraint into the same group. Since incorrect delays are randomly distributed, we omit those groups with much less elements than other groups. The result is shown in Step 2.
3) After removing the incorrect delays, we find there are very large $\tilde{t}_{\mathrm{d}}$. Those large values are due to timestamp overflow. In our measurement method, the Souce-TimeAtSink provided by MLT is a 4-B timestamp based on a 32-kHz timer. Hence the maximum time is $t_{\max} = 0xFFFFFFFF/32$ ms, i.e., about 1.5 days. Since normally $\tilde{t}_{\mathrm{d}}$ is smaller than 1.5 days, we set $\tilde{t}_{\mathrm{d}} = \tilde{t}_{\mathrm{d}} \bmod t_{\max}$. We show $\tilde{t}_{\mathrm{d}}$ in Step 3.
4) Until now, we have calculated the delays for the correct timestamps. At this step, we recover those incorrect delays according to (7) with previously received correct packets.

The final result is shown in Step 4 in Fig. 6.

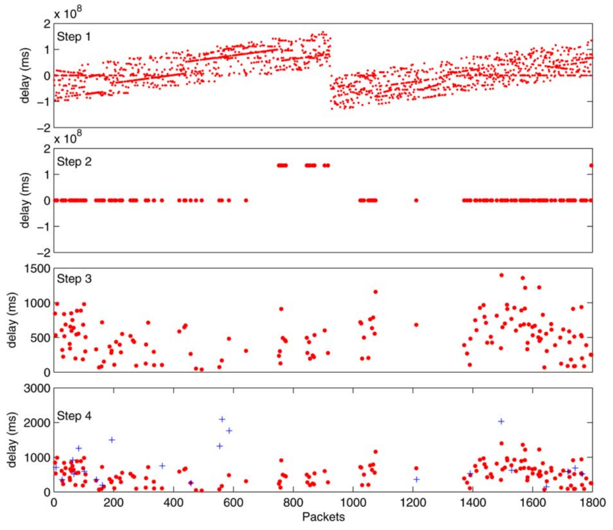


Fig. 6. Delay processing. Step 1: Original data. Step 2: Result after filtering the incorrect delays. Step 3: Result after recovering the overflow timestamps. Step 4: Result after recovering the incorrect delays based on the result in Step 3. Crosses are the recovered delays.

IV. Delay Overview

A. Overall Distribution

Normally, the maximum single-hop delay is L according to the LPL mechanism without other impacting factors, where L is the cycle length of LPL, e.g., $L = 500$ ms in our network. For a packet of k hops, the delay should be distributed between 0 and kL without other impacting factors. The expected delay for such a path should be $kT/2$. Fig. 7 shows the overall delay distribution for one subnet. The x-axis is the node ID, and y-axis is the delay. For each node, we show the statistics of the delays with the median, 25th percentile, 75th percentile, $k \times 500$, and the lowest delay, where k is the average hop count for this node. We sort all the nodes with respect to the median delay. The crosses in the figure represent delays larger than $k \times 500$. We denote those delays as large delays. Overall, this figure presents several kinds of information: 1) The delay distribution exhibits randomness. 2) Though the delay of different nodes varies in a large range, the median delays are evenly distributed between

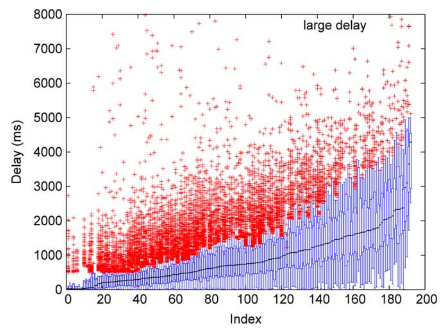


Fig. 7. Delay distribution for all nodes.

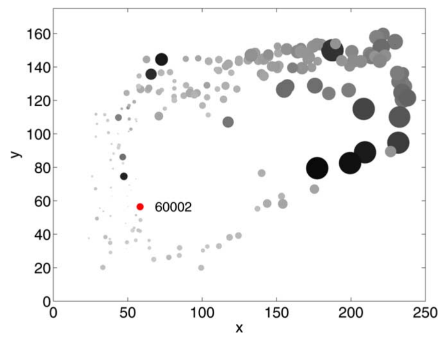


Fig. 8. Spatial distribution.

0–2 s. 3) There exist many large delays for most nodes. Later, we will explain the reasons for those large delays.

B. Spatial Distribution

We further look at the spatial distribution of delays in the network. In Fig. 8, each circle is plotted according to the physical location of different nodes. In the middle area, there is a building. The radius of each circle represents the average delay over the measurement period, and the depth of the color represents the delay variation. A darker color indicates a larger variation. The red node (60002) is the sink node. We can see that nodes far away from the sink node have large average delays as well as large delay variations. Nodes in the right top area are farthest from the sink node and have the largest delays and delay variations.

C. Overall Clock Drift

Before examining the delay details, we look at the clock drift of sensor nodes. To calculate the relative clock drift, as in [19], we apply robust linear fitting for the collected timestamps and then calculate the slope as the relative clock drift for all nodes to sink node. Fig. 9 shows the cumulative distribution function (CDF) of the relative clock drift for all nodes. The x-axis is the drift, and y-axis is the CDF of nodes. More than 90% of nodes have a clock drift less than 40 ppm. This coincides with the result from [23]. The result shows that the clock drift of most nodes in the outdoor environment is relatively stable.

V. Analysis of Different Factors

In the data, we have various parameters from different aspects. We first collect features that may have an impact on delay as reported in existing works, such as queue length, backoff

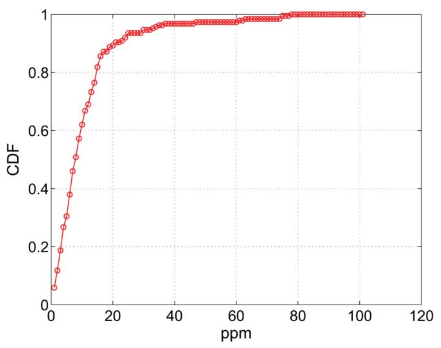


Fig. 9. CDF of drift for all nodes.

time, parent change, etc. We also collect environment factors (e.g., temperature, humidity, and light), routing parameters (e.g., routing loop), data transmission parameters (e.g., overflow, retransmission, duplicated counters), and sensor node status (e.g., radio duty cycle, radio on time, task execution time, etc.).

It is difficult to identify important impacting parameters to the delay performance. A single delay change may be accompanied by variations of different parameters. Moreover, the randomness introduced in LPL mechanism makes it even difficult to extract important factors. To address those issues, in this section, we leverage an automatic tool to identify important impacting factors and then investigate those important factors.

A. Important Factors

We use Rulefit [16] to find the important factors. Rulefit is a supervised learning approach to train predictors based on rule ensembles. Rulefit first trains a decision tree based on the input data. The decision tree can provide rules, each of which is a combination of one or more feature tests, i.e., combining one or more features into simple "and" tests. Let x be a vector of n features and s_j be a subset of possible values for feature x_j. Then, a rule takes the form of

$$r(x) = \prod_{j=1}^{n} I(x_j \in s_j) \tag{8}$$

where $I(\cdot)$ is an indicator function. A rule takes value one if all feature tests in the rule take value one. Rulefit provides the relative importance of different features based on rules. First, each rule is given an importance value according to its importance in the decision tree. Then, the importance for a variable in a rule is calculated as the importance of the rule divided by the number of variables in the rule. The importance for a variable is the sum of the importance for the variable in all rules containing this variable.

Rulefit has two properties: 1) it can rank features by their relative importance to the prediction goal; and 2) it can provide easy-to-interpret rules (combinations of features) for user understanding. Rulefit has been adopted in recent works, e.g., [24], to understand different impacting factors. Here, we present an overview of Rulefit approach. Interested readers can refer to [16] for more details.

We apply Rulefit to the collected parameters and delays. The ranking result of all parameters by Rulefit is shown in Fig. 10. The top five most important factors are retransmission, hop count, queue length, congestion backoff, and temperature. By

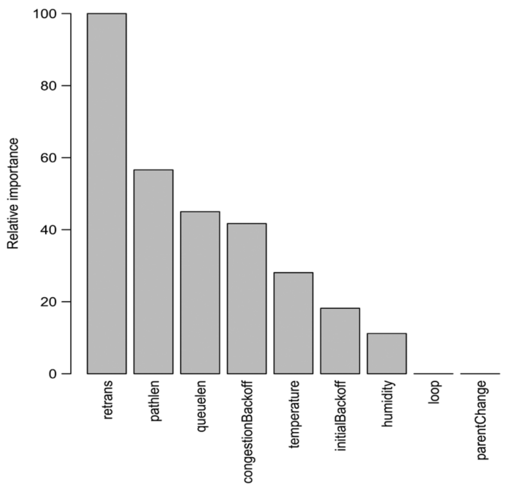


Fig. 10. Relative importance of the parameters.

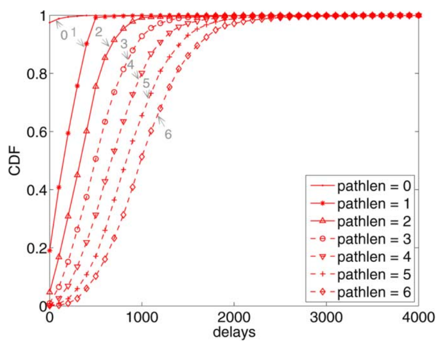


Fig. 11. Delay distributions to different hops.

applying Rulefit, we can filter those less important factors and then only focus on important factors for a real network.

B. Detailed Correlation

We now investigate those important factors and examine the relationship of those factors to the delay performance.

1) Hop Count: Fig. 11 shows the delay distribution with respect to different hop counts. Overall, packets with a larger hop count to the sink node have a larger delay. More specifically, this figure shows two kinds of information. First, for hop count k, most delays (more than 80%) are less than $k \times 500$ ms, which coincides with the settings of LPL in our network. Second, this figure also shows that for each hop count, there exist many large delays, indicating other impacting factors to the delay performance.

2) Retransmission: Fig. 12(a) shows the average retransmission count at each hop. We can see that retransmissions per packet at different hops are almost equal. Though the traffic is high for nodes near the sink, the corresponding retransmission count for those nodes is similar to other nodes. We investigate the data and find the collisions near the sink are not severe. Note that the retransmission count for nodes at hop 1 is much lower than other nodes since the radio of the sink is always on.

Fig. 12(b) shows the delay with different retransmission counts. It can be seen that the retransmission count and delay have a similar trend. The delay increases with the increasing of retransmission count. Meanwhile, we calculate the Pearson correlation for the retransmission and delay. The Pearson correlation between two variables X and Y is calculated as the covariance of the two variables divided by the product of their standard deviations, i.e.,

$$p_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}. \tag{9}$$

Then, we show the CDF of correlations for all nodes. As in Fig. 12(c), the x-axis is the correlation, and y-axis is the CDF of nodes. We find that most nodes have high correlations between retransmission and delay.

Furthermore, we look into those large delays. For hop k, we denote the delays larger than $k \times 500$ as large delays and other delays as normal delays. In order to correlate those large delays to retransmissions, we set a packet with retransmission count larger than 0 as a retransmission event. Then, in the entire network, we calculate how retransmission events can be used to predict large delays. For a packet with a retransmission event, if such a packet has a large delay, we say the event is correlated to a large delay and call it a *true positive* (TP). Otherwise, we call it a *false negative* (FN). For a packet without retransmission event, if such a packet is correlated to a normal delay, we call it a *true negative* (TN). Otherwise, we call it a *false positive* (FP). Then, we calculate the *accuracy*, which gives the probability that a retransmission event can be used to predict a large delay, i.e.,

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}. \tag{10}$$

Fig. 12(d) shows the CDF of accuracy for all nodes. The x-axis is the accuracy, and y-axis is the CDF of nodes. The result shows the accuracy for most nodes is high. More than 80% of nodes have an accuracy ratio higher than 60%.

Considering an extreme case when all packets have large delays, even randomly selecting a large enough subset of packets as retransmission events would lead to a high accuracy. To address such a problem, as in [25], we calculate the *balanced accuracy*, i.e.,

$$\text{balanced accuracy} = \frac{0.5 \times \text{TP}}{\text{TP} + \text{FN}} + \frac{0.5 \times \text{TN}}{\text{TN} + \text{FP}}. \tag{11}$$

Fig. 12(d) shows the CDF of balanced accuracy. We can see that for balanced accuracy, more than 70% of nodes still have a balanced accuracy higher than 0.6.

3) Queuing: We also examine the impact of queuing on the delay performance. There is only one queue on each node. For each packet along the path to sink node, we record the queue length when the packet arrives at each node. Fig. 13(a) shows the average queue length for different hops. Unlike the retransmission count, the average queue length varies for different hops. Nodes near the sink have larger average queue length. This indicates that although high traffic near the sink does not incur packet losses and retransmissions, it results in more congestions and thus a larger queue length. Since the sink is always on, nodes within 1 hop from sink node can quickly drain their packets and thus have a smaller queue length.

Fig. 13(b) shows the delay with respect to the total queue length along the path. We also find that many packets with a large queue length are correlated with large delays. We also calculate the Pearson correlation. Fig. 13(c) shows the CDF of correlation for all nodes. The x-axis is the correlation, and y-axis is the CDF of nodes. We can see that for most nodes, delay has

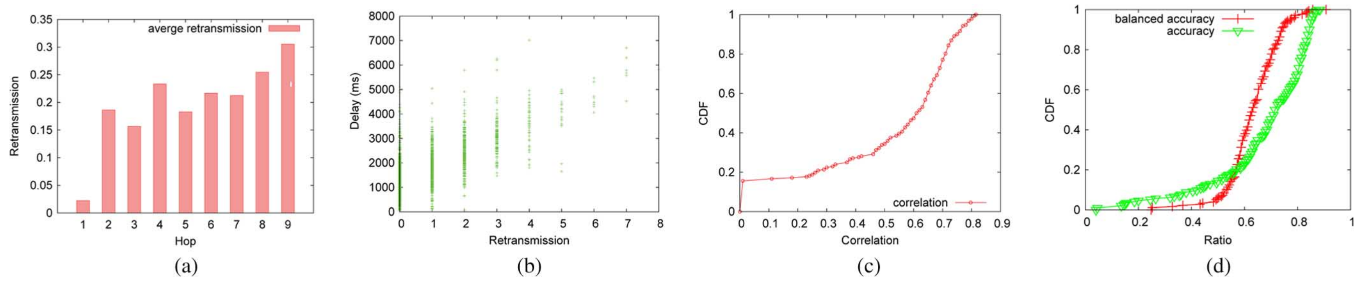


Fig. 12. Impact of retransmission to delay. (a) Hop count to average retransmissions. (b) Retransmission with respect to delay distribution. (c) CDF of correlation between retransmission and delay for all nodes. (d) Accuracy and balanced accuracy of using retransmission to predict large delay.

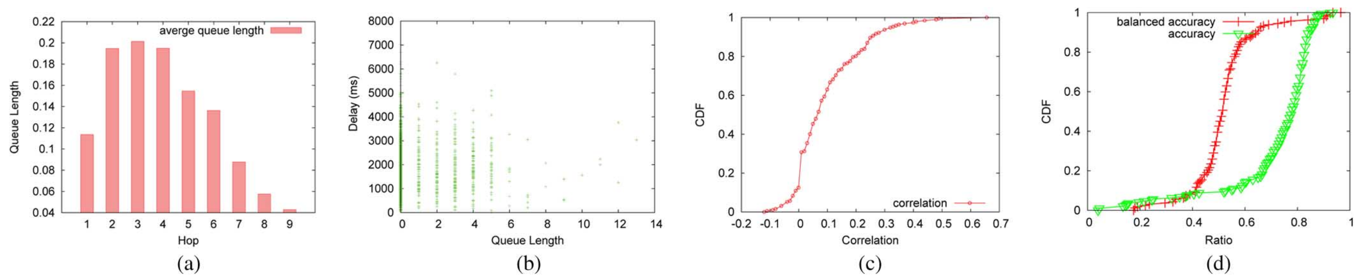


Fig. 13. Impact of queuing events to delay. (a) Queue length to average retransmissions. (b) Queue length with respect to delay distribution. (c) CDF of correlation between queue length and delay for all nodes. (d) Accuracy and balanced accuracy of using queuing events to predict large delay.

a positive correlation with the queue length. However, the correlation is not as strong as that of retransmission. In LPL, the receiver is awake with a high probability after the first packet in the queue is transmitted. Thus, consecutive packets may not need to wait until the receiver wakes up. Therefore, the impact of queue length on delay is relatively small. We further explain the reason in the delay model in Section VI.

We further examine the predictability of queue length to large delays. Similar to retransmission, we set a packet with a queue length larger than 0 as a queuing event. Then, we correlate those queuing events to large delays and calculate the accuracy and balanced accuracy. The CDF of accuracy and balanced accuracy are shown in Fig. 13(d). First, we can find that on many nodes, queuing events are correlated with large delays, e.g., about 80% of nodes have accuracy higher than 60%. For balanced accuracy, about 80% of nodes have balanced accuracy higher than 40%. In total, we can see that the correlation here is not as high as that for retransmission, which coincides with the result of Fig. 13(c).

4) Other Factors: From the result of Rulefit, we find that environment factors, MAC backoffs (including congestion backoff and initial backoff), and routing events (including parent change and loop event) are not as important as aforementioned factors.

It has been shown that environment factors such as temperature and humidity affect the clock drift and link quality [23], [26]. Fig. 9 shows the drift of sensor nodes is relatively small, and thus its impact to packet delay is also limited. The impact of environment on link quality has also been studied in different works such as [26]. Link quality is indeed related to delay performance. Such an impact is captured in retransmissions. Thus, we do not consider environment as a direct impacting factor.

The impact of MAC backoffs on delay is also extensively studied in different works such as [27]. In our network, we find that the average backoff for each packet is relatively small. Thus, the impact of backoff is not as significant as other factors.

The impact of routing events, such as parent change and loop events, has been studied [28] in Internet. It has been shown that those events may lead to a large end-to-end delay [28]. For different network deployments, the impacting factors may be different. The differences should be considered in protocol design. Different from Internet, our result shows that the impact of parent change is relatively small in WSNs. On the other hand, the impact of wireless link quality, retransmission, and so on becomes high due to the following reasons. First, the number of those events is relatively small. Second, nodes often switch among forwarders with similar hop counts. In practical protocol design, we should consider the scenario when there are no forwarders with a similar hop count. Third, events such as loop events can be quickly recovered by the network protocol. For example, when a loop is detected in CTP, the beacon interval is decreased to minimum in order to propagate the information as quickly as possible, alleviating the impact of loop event on end-to-end delay.

C. Apply Rulefit to Other Networks

To further show that our method can be applied in other networks, we implement our method with the recent proposed received-initiated low-duty cycle protocol A-MAC [29]. We evaluate the protocol in an indoor network. Fig. 14 shows the result of Rulefit. We can see that retransmission and queue length are still two of the most significant impacting factors. Different from the result in Fig. 10, the impact of other factors is much lower. We investigate the data and find that there are less contentions for indoor environment with a small network size. Meanwhile, the temperature for the environment is more stable than the outdoor environment. The loop event and parent

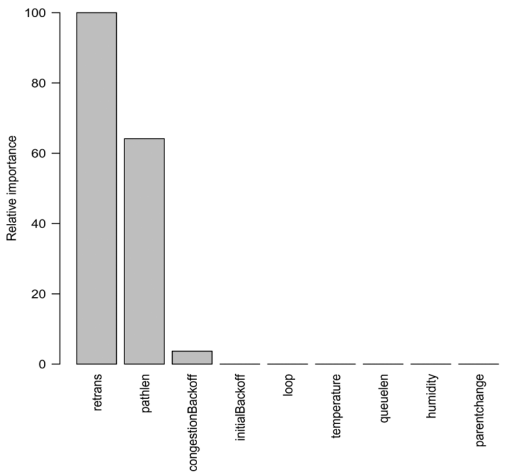


Fig. 14. Impacting factors of A-MAC.

change event are also very rare. Nevertheless, we can still obtain impacting factors using Rulefit for A-MAC.

VI. Delay Model

According to the analysis of different factors, in this section, we build a model for the end-to-end packet delay and validate it in our network.

A. Model

We first model the single-hop packet delay and then extend it to multihop end-to-end delay. To derive the model, we describe the following parameters:

- t_{s}: the sleep time in LPL;
- t_{w}: the awake time of the receiver in LPL;
- t_{b}: MAC layer backoff time, including the initial backoff time and congestion backoff time;
- u: the duty cycle ratio of the receiver, i.e., the awake time divided by the cycle length;
- t_{x}: the packet modulation/demodulation time. This is usually platform-dependent and is related to the packet size;
- r: the number of retransmissions for a packet.

We first calculate the delay for a single-hop packet transmission. Assume the packet is retransmitted r times. According to the mechanism of LPL, each unsuccessful transmission takes of time of length $t_{\mathrm{w}}+t_{\mathrm{s}}$. Thus, the time used for r retransmissions is $r(t_{\mathrm{w}}+t_{\mathrm{s}})$. For the $(r+1)$th transmission that is successful, there are two cases in LPL.

- *Case 1*: If the receiver is sleeping, the sender should wait until the receiver wakes up and then send the packet to the receiver.
- *Case 2*: Otherwise, the packet can be sent directly.

For Case 1, since the transmission can fall into any time during the sleeping period of the receiver, the delay is $U(0,t_{\mathrm{s}})+t_{\mathrm{b}}+t_{\mathrm{x}}$, where $U(0,t_{\mathrm{s}})$ is a random distribution between 0 and t_{s}. For Case 2, the delay for a packet is $t_{\mathrm{b}}+t_{\mathrm{x}}$. According to the duty cycle ratio of each receiver, the probability for Case 1 is $1-u$, and for Case 2 is u.

Consequently, the 1-hop delay is given by

$$T(t_{\mathrm{s}},t_{\mathrm{w}},r)=\begin{cases} r(t_{\mathrm{w}}+t_{\mathrm{s}})+t_{\mathrm{b}}+t_{\mathrm{x}}, & \text{with prob. } u \\ r(t_{\mathrm{w}}+t_{\mathrm{s}})+U(0,t_{\mathrm{s}}) & \\ \quad +t_{\mathrm{b}}+t_{\mathrm{x}}, & \text{otherwise.} \end{cases} \tag{12}$$

Based on the single-hop delay, we derive the delay for a multihop path. A packet p is transmitted on a path consisting of n nodes from node 1 to n. At each node, the packet is first put into the transmission queue and then transmitted after prior packets in the queue are transmitted. To calculate the multihop delay, we first describe the following parameters:

- l_i: queue length at node i, i.e., the number of packets in the transmission queue, including the packet p. The packet needs to wait until prior l_i-1 packets are transmitted.
- $r_{i,j}$: the number of retransmissions for the jth packet in the queue on node i.

The time for the first packet in the queue can be calculated according to (12). We then calculate the time for following packets in the queue. In LPL, the receiver should stay awake after receiving a packet. When $r=0$, the packet in the queue is directly sent to the receiver since the receiver is awake. Otherwise, the packet is retransmitted since the receiver is sleeping. Thus, for packets except the first one in the queue, the delay is

$$T_{\mathrm{q}}(t_{\mathrm{s}},t_{\mathrm{w}},r)=\begin{cases} t_{\mathrm{b}}+t_{\mathrm{x}}, & r=0 \\ T(t_{\mathrm{s}},t_{\mathrm{w}},r), & \text{otherwise.} \end{cases} \tag{13}$$

Therefore, the delay for a path consisting of n nodes $(1,2,\ldots,n)$ is given by

$$D(n)=\sum_{i=1}^{n-1}T\left(t_s^{i+1},t_w^{i+1},r_{i,l_i}\right)+\sum_{i=1}^{n-1}\sum_{j=1}^{l_i-1}T_q\left(t_s^{i+1},t_w^{i+1},r_{i,j}\right) \tag{14}$$

where $T(t_s^{i+1},t_w^{i+1},r_{i,l_i})$ is the single-hop delay for the first packet in the queue at node i, $T_{\mathrm{q}}(t_{\mathrm{s}}^{i+1},t_{\mathrm{w}}^{i+1},r_{i,j})$ is the delay for the jth packet in the queue, t_{s}^{i+1} is the sleep time, and t_{w}^{i+1} is the awake time of node $i+1$.

B. Model Validation

We validate our model with the collected data. In CitySee, we have recorded the queue length on each node of the path, i.e., l_i. We also recorded the ETX value from each node to its neighbors. This can be used to approximate $r_{i,j}$, for $j>1$. For each packet, we also recorded the retransmissions on each hop, which is $r_{i,j}$, for $j=1$. On each node, we calculate the average duty cycle u and average backoff time t_{b}. Using those parameters as input, we calculate the delay with the model. Then, we compare the result of the model to the delay calculated from the packets. The result is shown in Fig. 15. We show the average delay and variations for different hops. We can see that both results increase linearly with the hop count. The model and practical delay have very similar distribution, showing that our model is effective to capture those important factors.

VII. Implications

In this section, we revisit three prevalent protocols and discuss the implications of our analysis and delay model.

A. Routing Protocol

We first analyze the commonly used data collection protocol CTP in WSNs. Through analysis, we find that CTP protocol, with ETX as the routing metric, may not appropriately choose a good path. For brevity, we assume the queuing delay on each

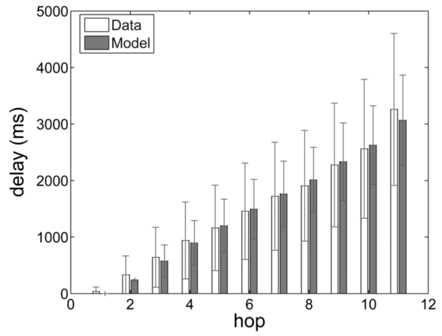


Fig. 15. Delay distribution of the model and collected data with respect to different hop counts.

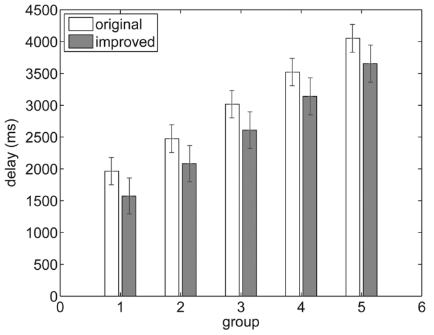


Fig. 16. Comparison of EDETX with CTP protocol.

hop is 0, i.e., $l_i = 1$ on each hop. Then, we have

$$D(n) = \sum_{i=1}^{n-1} T\left(t_{\mathrm{s}}^{i+1}, t_{\mathrm{w}}^{i+1}, r_{i,1}\right).$$

According to (12), the expected delay is

$$E\left(D(n)\right) = E\left(\sum_{i=1}^{n-1}\left(r_{i,1}(t_{\mathrm{w}}+t_{\mathrm{s}})+t_{\mathrm{b}}+t_{\mathrm{x}}\right)\right) + (1-u)E\left(\sum_{i=1}^{n-1} U(0,t_{\mathrm{s}})\right).$$

Denote ETX_i as the ETX for the link from node i to $i+1$. According to the definition of ETX, we have $ETX_i = E(r_{i,1})+1$. The expected delay on the path is

$$\begin{aligned} E\left(D(n)\right) &= \sum_{i=1}^{n-1}\left((ETX_i-1)(t_{\mathrm{w}}+t_{\mathrm{s}})+t_{\mathrm{b}}+t_{\mathrm{x}}\right) \\ &\quad + (1-u)(n-1)\frac{t_{\mathrm{s}}}{2} \\ &= PathETX(t_{\mathrm{w}}+t_{\mathrm{s}}) \\ &\quad - (n-1)\left(t_{\mathrm{w}}+\frac{1+u}{2}t_{\mathrm{s}}-t_{\mathrm{b}}-t_{\mathrm{x}}\right). \end{aligned} \quad (15)$$

We can see that even for paths with the same path ETX, the expected end-to-end delay can be different. In practical settings, the awake time t_{w} and sleep time t_{s} are usually larger than the backoff time t_{b} and data transmission time t_{x}. Thus, we have $(n-1)(t_{\mathrm{w}}+((1+u)/2)t_{\mathrm{s}}-t_{\mathrm{b}}-t_{\mathrm{x}})>0$. According to (15), for the same path ETX, $E(D(n))$ decreases as the hop count n increases. This indicates that longer paths, which are often prohibited previously, may even be better than shorter ones.

Considering a simple example for two paths with the same ETX of 2: The first path consists of one link with a link ETX 2; the second path consists of two links, each of which has a link ETX 1. CTP treats those two paths equally and thus randomly chooses one as the routing path (in current implementation, the one that appears earlier in the routing table is selected). In fact, according to the delay model, the second path is better than the first one in terms of delay, even though the second one has a larger hop count. For the first path, there are two transmissions in expectation including one retransmission. For the second path, there are two transmissions in expectation without retransmissions. Intuitively, according to the mechanism of LPL, if there is no retransmission, the expected time is $L/2$, where L is the cycle in LPL. However, if a packet is retransmitted, the expected time is L. Hence, the second path is better. This also explains in Fig. 10 why retransmission count is more important than hop count. In current design, the ETX metric only counts the expectation of transmissions and makes no difference for retransmissions and successful transmissions.

Accordingly, we propose a new path quality metric, i.e., expected delay based on ETX (EDETX). According to (12), the $EDETX_{i,j}$ for a single hop from node i to node j is calculated as $EDETX_{i,j} = (ETX_{i,j}-1)(t_{\mathrm{w}}+t_{\mathrm{s}})+t_{\mathrm{b}}+t_{\mathrm{x}}+(1-u)t_{\mathrm{s}}/2$, where $ETX_{i,j}$ is the ETX from node i to node j. Those parameters can be calculated by each node locally. Based on the single-hop EDETX, each node can calculate the path EDETX to the sink node, which is the sum of link EDETXs on the path. We improve the original CTP protocol with the EDETX metric. In the improved protocol, to minimize the transmission cost, each node first selects a parent with minimal ETX. Then, the node will select a parent with minimal EDETX. Thus, EDETX will not incur additional transmission overhead. It is worth noting EDETX can also be used without ETX to improve delay performance. Fig. 16 shows the comparison result for the improved protocol and CTP. The x-axis is the group of packets with the same ETX. We can see that for different groups, the average delay with EDETX is reduced.

B. Opportunistic Routing

Opportunistic routing can significantly improve system performance [30], [31]. We evaluate the practical performance of opportunistic routing with a real trace from CitySee. For each node, we calculate the probability that the delay performance can be improved, i.e., the portion of neighbors with smaller delays than the node itself. We evaluate two different strategies as shown in Fig. 17. First, we evaluate the strategy that a node can select the next forwarder in all neighbors. For more than 50% of nodes, the probability to select a better node is higher than 50%. However, opportunistic routing in all neighbors may still result in selection of a forwarder with a larger delay.

Furthermore, we investigate the second strategy. Each node only selects neighbors closer to the sink node. We can see that it is better to select nodes from neighbors with smaller hop counts to the sink node. For more than 80% of nodes, the probability of improvement is larger than 95%.

C. Dynamic Switching-Based Forwarding

Dynamic switching-based forwarding (DSF) [32] is proposed to optimize the expected end-to-end delay in WSNs by selecting a set of forwarders. In DSF, each node searches in its neighbors to find an optimal forwarding set with minimum expected end-

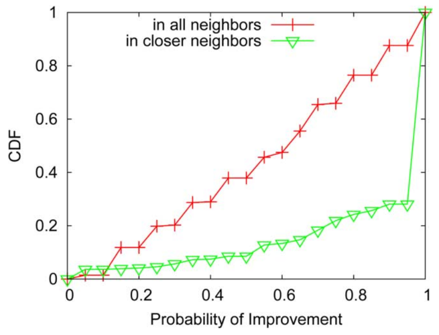


Fig. 17. Evaluation of opportunistic routing.

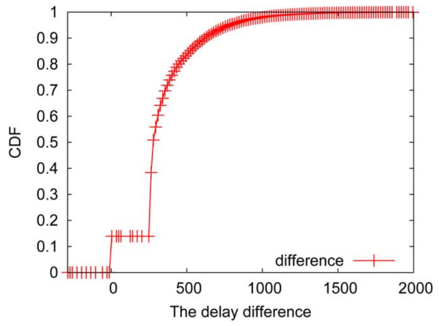


Fig. 18. Difference of the practical packet delay and DSF.

to-end delays. More specifically, a node searches through its neighbors and calculates whether it is beneficial to add each neighbor into the forwarding set.

For each node, we calculate its link quality q_i to each neighbor i and the delay T_i of neighbor i. Assume the forwarding set is $F = \{v_1, v_2, \ldots, v_{|F|}\}$ and $v_1, v_2, \ldots, v_{|F|}$ are in ascending order of waking-up time. For brevity, for a node that wakes up multiple times, we let it appear multiple times in the set. Then, the expected delay is

$$D(F) = \left(\sum_{i=1}^{|F|} \prod_{k=0}^{i-1} (1-q_k) q_i T_i\right) \tag{16}$$

where $q_0 = 0$.

DSF uses dynamic programming to find F such that $D(F)$ is minimized with a certain packet reception ratio being satisfied. We compare the delay of DSF using our trace to the practical measured packet delay in our network. Fig. 18 shows the delay difference of measured delay in our network and the delay of applying DSF to our trace, i.e., practical delay $-$DSF delay. First, for most packets, the difference is larger than 0. We can see that for more than 75% of packets, the practical delay is larger than the DSF delay, indicating that using DSF is beneficial in terms of delay performance. Meanwhile, note that the improvement for most nodes is less than 500 ms, indicating further improvement is possible. In LPL protocol, we can extend the delay model to the network without fixed scheduling for different nodes. In this case, as in (12), the probability for retransmission count r can be calculated [33] as

$$p(r=k) = \prod_{i \in F} (1-q_i)^k \left(1 - \prod_{i \in F}(1-q_i)\right). \tag{17}$$

Thus, the expected number of retransmission can be calculated as

$$E(r) = \frac{\prod_{i \in F}(1-q_i)}{1 - \prod_{i \in F}(1-q_i)}. \tag{18}$$

Then, the single-hop delay can be calculated as

$$D_{e,F} = E(r)(t_{\mathrm{w}} + t_{\mathrm{s}}) + t_{\mathrm{b}} + t_{\mathrm{x}} + \mu t_{\mathrm{s}} \tag{19}$$

where e is the sender and μt_{s} is the expected delay for a successful transmission. We can further calculate the multiple-hop delay and improve the performance according to techniques described in Section VI.

VIII. Related Work

Delay Analysis: There are extensive works for delay performance analysis in WSNs. First, probabilistic delay bounds are proposed in [5]–[8] by extending network calculus. In the second category, stochastic delay models are proposed. For example, in [9]–[11], different models are proposed by combining real-time theory and queuing theory. In those models, unreliable networks with heavy traffic are considered. There are some empirical network delay models [13], [14] proposed for end-to-end delay measurements. A delay model based on Discrete Markov Processing in the network is proposed in [12]. Besides those end-to-end delay models, the single-hop channel access delay models are also analyzed in [27], [34], and [35]. However, those works proposed in WSNs are often based on assumptions—e.g., traffic or routing path—and not evaluated in a real large-scale network. In this paper, we are the first to propose a lightweight delay measurement and analysis in an operational LPL WSN.

Delay in Internet/Data Center: There are also a large number of research works in Internet and data centers. Pucha *et al.* [28] show the impact of routing events on end-to-end delay in Internet. Kompella *et al.* [3] present fine-grain latency measurements in presence of packet losses for internet with a lossy difference aggregator. To measure the per-flow delay, Lee *et al.* [15] present a measurement method with reference delay interpolation. As the development of data center technologies, Wilson *et al.* [4] present delay analysis results in the data center and provide guidelines to the data center design.

Time Synchronization: There are also many global time synchronization methods in wireless sensor networks, e.g., [19], [20], and [36]. Those methods can synchronize all nodes in the network and provide synchronized timestamps. Our measurement method is different from those methods in two aspects. First, our method does not require time synchronization among all nodes in the network. We recover the timestamp at the sink side. Thus, this reduces the message exchange overhead among all nodes. Second, our method does not need to maintain a synchronized timestamp all the time. Therefore, our method does not require periodical message exchange.

Large-Scale Sensor Network Deployment: There are many sensor network deployments in the world. Table II shows several network deployments in the world.

IX. Conclusion

We focus on delay measurement, analysis, and implications in an operational large-scale LPL wireless sensor network. We propose a lightweight delay measurement method to efficiently calculate delay without time synchronization, which is applicable to operational networks. We analyze the spatial and temporal characteristics of delay distribution through carefully examine system metrics. Furthermore, we extract different impacting parameters to delay performance with the incomplete data, propose a practical delay model to capture those factors, and validate it in a large-scale network. Finally, we show the implications of measurement and analysis to protocol design.

Appendix A

Proof: According to the linear clock model, we have

$$\Delta\tilde{t}_{\mathrm{r}}(i_1,i_2,O) = (1+\alpha_1)\Delta t_{\mathrm{r}}(i_1,i_2,O) \quad (20)$$

$$\Delta\tilde{t}_{\mathrm{r}}(i_1,i_2,D) = (1+\alpha_2)\Delta t_{\mathrm{r}}(i_1,i_2,D) \quad (21)$$

$$\sum_{j=1}^{n-1}\tilde{t}_{\mathrm{w}}(i_1,j,j) = (1+\alpha_3)\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_1,j) \quad (22)$$

$$\sum_{j=1}^{n-1}\tilde{t}_{\mathrm{w}}(i_2,j,j) = (1+\alpha_4)\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_2,j) \quad (23)$$

where α_1, α_2, α_3 and α_4 are clock drifts bounded by $[-\hat{\alpha},\hat{\alpha}]$.

Thus, we have

$$\Delta\tilde{t}_{\mathrm{a}} = \left(\tilde{t}_{\mathrm{r}}(i_1,D,D) - \sum_{j=1}^{n-1}\tilde{t}_{\mathrm{w}}(i_1,j,j)\right) - \left(\tilde{t}_{\mathrm{r}}(i_2,D,D) - \sum_{j=1}^{n-1}\tilde{t}_{\mathrm{w}}(i_2,j,j)\right). \quad (24)$$

According to (20)–(23), we have

$$\begin{aligned}\Delta\tilde{t}_{\mathrm{a}} &= \left(\tilde{t}_{\mathrm{r}}(i_1,D,D) - \sum_{j=1}^{n-1}\tilde{t}_{\mathrm{w}}(i_1,j,j)\right) - \left(\tilde{t}_{\mathrm{r}}(i_2,D,D) - \sum_{j=1}^{n-1}\tilde{t}_{\mathrm{w}}(i_2,j,j)\right)\\ &= (1+\alpha_2)\Delta t_{\mathrm{r}}(i_1,i_2,D) - (1+\alpha_3)\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_1,j)\\ &\quad + (1+\alpha_4)\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_2,j). \end{aligned} \quad (25)$$

Since α_2, α_3, and α_4 are bounded by $[\hat{\alpha},\hat{\alpha}]$, we have

$$\begin{aligned}\Delta\tilde{t}_{\mathrm{a}} &\le (1+\hat{\alpha})\Delta t_{\mathrm{r}}(i_1,i_2,D) - (1-\hat{\alpha})\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_1,j)\\ &\quad + (1+\hat{\alpha})\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_2,j). \end{aligned} \quad (26)$$

According to (2) and (3), we have

$$\Delta\tilde{t}_{\mathrm{a}} \le (1+\hat{\alpha})\Delta t_{\mathrm{r}}(i_1,i_2,O) + 2\hat{\alpha}\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_1,j). \quad (27)$$

TABLE II
WSN Deployments

System	Scale	Power supply	environment
Great Duck island [37]	100+	Battery	Outdoor
VigilNet [38]	200	Battery	Outdoor
MoteLab [39]	190	Tethered	Indoor
SensorScope [40]	97	Solar	Outdoor
Trio [41]	557	Solar/Battery	Outdoor
CitySee	1200	Battery	Outdoor

Since $\hat{\alpha}$ is in the magnitude of 10^{-6} and $\sum_{j=1}^{n-1} t_{\mathrm{w}}(i_1,j)$ is in the magnitude of 10^{-3} s, which are much smaller than $(1+\hat{\alpha})\Delta t_{\mathrm{r}}(i_1,i_2,O)$, we ignore the second term and obtain

$$\Delta\tilde{t}_{\mathrm{a}} \le (1+\hat{\alpha})\Delta t_{\mathrm{r}}(i_1,i_2,O). \quad (28)$$

Combining (20), for small $\hat{\alpha}$, we have

$$\begin{aligned}\Delta\tilde{t}_{\mathrm{a}} &\le \frac{(1+\hat{\alpha})\Delta\tilde{t}_{\mathrm{r}}(i_1,i_2,O)}{1-\hat{\alpha}}\\ &\approx (1+2\hat{\alpha})\Delta\tilde{t}_{\mathrm{r}}(i_1,i_2,O). \end{aligned} \quad (29)$$

Similarly, we have

$$\Delta\tilde{t}_{\mathrm{a}} \ge (1-2\hat{\alpha})\Delta\tilde{t}_{\mathrm{r}}(i_1,i_2,O). \quad (30)$$

Combining (29) and (30), we prove the theorem. □

Appendix B

Proof: According to (5), we have

$$\begin{aligned}\tilde{t}_{\mathrm{d}}(i) &= \left(\tilde{t}_{\mathrm{r}}(i,D,D) - \tilde{t}_{\mathrm{a}}(i)\right)\\ &= \sum_{j=1}^{n-1}\tilde{t}_{\mathrm{w}}(i,j,j). \end{aligned} \quad (31)$$

Based on (22), we have

$$\begin{aligned}\tilde{t}_{\mathrm{d}}(i) &\le (1+\hat{\alpha})\sum_{j=1}^{n-1} t_{\mathrm{w}}(i,j)\\ &= (1+\hat{\alpha})t_{\mathrm{d}}(i). \end{aligned} \quad (32)$$

Similarly, we have

$$\tilde{t}_{\mathrm{d}}(i) \ge (1-\hat{\alpha})t_{\mathrm{d}}. \quad (33)$$

Combining (32) and (33), we prove the theorem. □

References

[1] T. B. O. Chipara, C. Lu, 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.

[2] S. N. Pakzad, G. L. Fenves, S. Kim, and D. E. Culler, “Design and implementation of scalable wireless sensor network for structural monitoring,” *J. Infrastruct. Syst.*, vol. 14, no. 1, pp. 89–101, 2008.

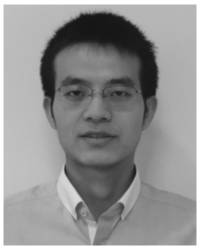
[3] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese, “Every microsecond counts: Tracking fine-grain latencies with a lossy difference aggregator,” in *Proc. ACM SIGCOMM*, 2009, pp. 255–266.

[4] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, “Better never than late: Meeting deadlines in datacenter networks,” in *Proc. ACM SIGCOMM*, 2011, pp. 50–61.

[5] M. Fidler, “An end-to-end probabilistic network calculus with moment generating functions,” in *Proc. IEEE IWQoS*, 2006, pp. 261–270.

[6] A. Koubaa, M. Alves, and E. Tovar, “Modeling and worst-case dimensioning of cluster-tree wireless sensor networks,” in *Proc. IEEE RTSS*, 2006, pp. 412–421.

[7] A. Burchard, J. Liebeherr, and S. Patek, "A min-plus calculus for end-to-end statistical service guarantees," *IEEE Trans. Inf. Theory*, vol. 52, no. 9, pp. 4105–4114, Sep. 2006.
[8] J. Schmitt and U. Roedig, "Sensor network calculus—A framework for worst case analysis," in *Proc. IEEE DCOSS*, 2005, pp. 141–154.
[9] J. Lehoczky, "Real-time queueing theory," in *Proc. IEEE RTSS*, 1996, pp. 186–195.
[10] J. P. Lehoczky, "Real-time queueing network theory," in *Proc. IEEE RTSS*, 1997, pp. 58–67.
[11] S.-N. Yeung and J. Lehoczky, "End-to-end delay analysis for real-time networks," in *Proc. IEEE RTSS*, 2001, pp. 299–309.
[12] Y. Wang, M. C. Vuran, and S. Goddard, "Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 305–318, Feb. 2012.
[13] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural, "Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2005, pp. 2646–2657.
[14] K. Gopalan, T. Chiueh, and Y.-J. Lin, "Probabilistic delay guarantees using delay distribution measurement," in *Proc. ACM MULTIMEDIA*, 2004, pp. 900–907.
[15] M. Lee, N. G. Duffield, and R. R. Kompella, "Not all microseconds are equal: Fine-grained per-flow measurements with reference latency interpolation," in *Proc. ACM SIGCOMM*, 2010, pp. 27–38.
[16] J. Friedman and B. Popescu, "Predictive learning via rule ensembles," *Ann. Appl. Statist.*, vol. 2, no. 3, pp. 916–954, 2008.
[17] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. ACM SenSys*, 2009, pp. 1–14.
[18] 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.
[19] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proc. ACM SenSys*, 2004, pp. 39–49.
[20] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. USENIX OSDI*, 2002, pp. 147–163.
[21] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. ACM/IEEE IPSN*, 2011, pp. 73–84.
[22] Y. Wang, Y. He, X. Mao, Y. Liu, Z. Huang, and X. Li, "Exploiting constructive interference for scalable flooding in wireless networks," in *Proc. IEEE INFOCOM*, 2012, pp. 2104–2112.
[23] T. Schmid, Z. Charbiwala, J. Friedman, Y. H. Cho, and M. B. Srivastava, "Exploiting manufacturing variations for compensating environment-induced clock drift in time synchronization," in *Proc. ACM SIGMETRICS*, 2008, pp. 97–108.
[24] I. Cunha, R. Teixeira, D. Veitch, and C. Diot, "Predicting and tracking internet path changes," in *Proc. ACM SIGCOMM*, 2011, pp. 122–133.
[25] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase, "Correlating instrumentation data to system states: A building block for automated diagnosis and control," in *Proc. USENIX OSDI*, 2004, vol. 6, p. 16.
[26] W. Xi, Y. He, Y. Liu, J. Zhao, L. Mo, Z. Yang, J. Wang, and X. Li, "Locating sensors in the wild: Pursuit of ranging quality," in *Proc. ACM SenSys*, 2010, pp. 295–308.
[27] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.
[28] H. Pucha, Y. Zhang, Z. M. Mao, and Y. C. Hu, "Understanding network delay changes caused by routing events," in *Proc. ACM SIGMETRICS*, 2007, pp. 73–84.
[29] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proc. ACM SenSys*, 2010, pp. 1–14.
[30] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. ACM SIGCOMM*, 2007, pp. 169–180.
[31] S. Biswas and R. Morris, "ExOR: Opportunistic multi-hop routing for wireless networks," in *Proc. ACM SIGCOMM*, 2005, pp. 133–144.
[32] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 12, pp. 1741–1754, Dec. 2011.
[33] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *Proc. ACM/IEEE IPSN*, 2012, pp. 185–196.
[34] T. Sakurai and H. Vu, "MAC access delay of IEEE 802.11 DCF," *IEEE Trans. Wireless Commun.*, vol. 6, no. 5, pp. 1702–1710, May 2007.
[35] O. Tickoo and B. Sikdar, "Modeling queueing and channel access delay in unsaturated IEEE 802.11 random access MAC based wireless networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 878–891, Aug. 2008.
[36] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *Proc. ACM/IEEE IPSN*, 2009, pp. 37–48.
[37] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proc. SenSys*, 2004, pp. 214–226.
[38] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "Vigilnet: An integrated sensor network system for energy-efficient surveillance," *Trans. Sensor Netw.*, vol. 2, no. 1, pp. 1–38, 2006.
[39] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: A wireless sensor network testbed," in *Proc. ACM/IEEE IPSN*, 2005, Art. no. 68.
[40] 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.
[41] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments," in *Proc. ACM/IEEE IPSN*, 2006, pp. 407–415.



Jiliang Wang (M'12) received the B.E. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2007, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2011.

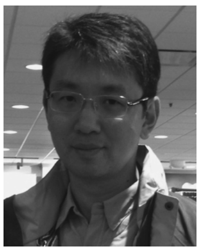
He is currently with the School of Software and TNLIST, Tsinghua University, Beijing, China. His research interests include wireless sensor networks, network measurement, and pervasive computing.

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

Zhichao Cao (M'13) received the B.E. degree in computer science from Tsinghua University, Beijing, China, in 2009, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2013.

His current research interests are wireless sensor networks and mobile 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 now Cheung Kong Professor with Tsinghua University. His research interests include RFID and sensor networks, the Internet, and pervasive computing.