On the Delay Performance Analysis in A Large-Scale Wireless Sensor Network

Jiliang Wang[1], Wei Dong[3], Zhichao Cao[2] and Yunhao Liu[1,2]
1. Tsinghua University, 2. Hong Kong University of Science and Technology, 3. Zhejiang University

***Abstract*—We present a comprehensive delay performance measurement and analysis in an operational large-scale urban wireless sensor network. We build a light-weight delay measurement system in such a network and present a robust method to calculate per-packet delay. Through analysis of delay and 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 design? In this paper, we explore the important factors from the data in presence of various metrics and randomness, and show that the important factors are not necessarily the same with that in Internet. Further, we propose a delay model to capture those factors and validate it in the network. We revisit several prevalent protocol designs such as Collection Tree Protocol, opportunistic routing and Dynamic Switching based Forwarding, and show the implications to protocol designs.**

I. Introduction

Recent advances in Wireless Sensor Networks (WSNs) have fostered a large number of applications, such as structural and health monitoring WSNs [1] [2] and 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 tremendous research efforts made to delay analysis and modeling in WSNs. For example, probabilistic delay bounds are presented in [3] [4] [5] [6] by extending network calculus. Further, stochastic delay models are proposed by combining real-time theory and queuing theory [7] [8] [9] or applying Discrete Markov Process [10]. There are also some empirical network delay models [11] [12] proposed for end-to-end delay measurements. Different from end-to-end delay models, the single hop channel access delay models are proposed in [13] [14] [15]. Those models and analysis assume specific network conditions, e.g., heavy traffic and fixed forwarding path, which are not always satisfied in real WSNs. Moreover, those studies lack comparison and validation with a real-world large-scale network.

There are also many research works on delay analysis and measurement in Internet and data centers. Kompella et al. [16] present a fine-grained latency measurement method in presence of packet losses for Internet with a lossy difference aggregator. This method can measure per-packet delay in Internet while incurring very limited additional traffic overhead. To measure the per-flow delay, Lee et al. [17] present a measurement method with reference delay interpolation. This work extends existing works to efficiently measure per-flow delay, which is important for QoS in real applications. As the development of data center technologies, Wilson et al. [18] present delay analysis results in data centers, providing guidelines to practical data center design.

We can see that delay measurement in practical systems has attracted many research attentions. 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, however, 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 an operational large-scale WSN, face non-trivial challenges. First, different from Internet and data centers, there are read-to-use software components in WSNs supporting per-packet delay measurement. Delay measurement relies on network synchronization. Traditional network synchronization introduces additional overhead and may not be reliable considering packet losses. Thus it is not always affordable to apply network synchronization to an operational network due to the limited node resource and large network scale. Second, analysis of the collected information is challenging. There are various performance metrics and a single delay change may be accompanied by variation of multiple metrics. On the other hand, collecting all required information from the network incurs high network overhead. Thus the information is usually incomplete due to resource constraints and packet losses. Moreover, the delay itself, according to protocol design, presents intrinsic randomness in its distribution. Efficiently and automatically extracting useful information from collected data becomes difficult.

In this work, 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 not introduce additional overhead. We present basic statistical characteristics based on the collected data. To systematically explore important impacting factors from various parameters, we leverage Rulefit to automatically identify the most important impacting factors, and show that the identified important factors are different from those in Internet. Further, we quantitatively calculate the correlation between different impacting factors and the delay performance. Based on those important factors, we present a practical delay model and validate the model using the collected data. Finally, we revisit three important protocols and show the implications to the protocol designs.

In summary, the contributions of this paper are as follows.

- We first build a measurement infrastructure in an operational large-scale wireless sensor network with little network overhead. Based on the data collected, we

Fig. 1: Deployment and sensor nodes in CitySee.

present the spatial and temporal characteristics of delay performance.

- We present a robust method to extract per-packet delay information. We present an automatic method based on Rulefit [19] to identify important impacting factors to delay from the data with randomness.
- We propose a practical model and validate it with the collected data. We revisit three important protocols and show the implications to those protocol designs.

The rest of the paper is organized as follows. Section II presents the system overview and delay measurement infrastructure. Section III presents the data processing and distribution overview of delay. Section IV introduces the method of identifying important factors from the data, presents a delay model and validates it. Section VI shows the implications of the analysis. Section VII introduces the related work. Finally, Section VIII concludes this work.

II. System Overview

In this section, we briefly introduce the overview of CitySee, including the sensor nodes, network structure and network protocols. Based on the system, we describe the delay measurement infrastructure and collected data.

A. The Network

The primary goal of CitySee is to precisely measure CO_2 emissions in a city-wide area. We started the project since July, 2011. 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, records system status, and then delivers those information to the sink. For CO_2 nodes, besides the functionalities of normal nodes, CO_2 nodes can also read the CO_2 concentration. The normal nodes and CO_2 nodes are deployed in an area to form a network and deliver their data to a sink node (normal node) in the network.

The mesh nodes have a high bandwidth of several MB/s and comprise the network backbone. Sink nodes of different subnets are connected to the network backbone in order to transfer packets from different subnets to the base station.

In CitySee, we incrementally deployed the network since July, 2011. The network covers an area of approximately 1,000,000 square meters in the Wuxi city, China. We have deployed about 1200 sensor nodes in total, with 1100 normal nodes and 100 CO_2 nodes. Figure 1 shows the deployment of the network.

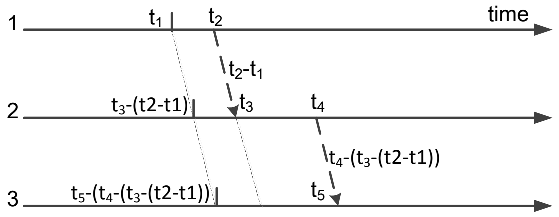

Fig. 2: Measuring delay in the network.

B. Protocols

1) Low Power Listening: At MAC level, we adopt the Low Power Listening (LPL) to save energy. In LPL, each node periodically samples the channel for a short duration (11ms) in each cycle. If energy is detected, the node stays awake for another short duration (30ms) to receive packets. Otherwise, the node turns off the radio and in the next cycle (e.g., 500ms 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 the cycle duration, in order to ensure that all neighboring nodes will wake up during the preamble time. For unicast with link layer ACK, the sender can stop the preambles until an ACK is received or the end of a cycle. Since a sender may begin to send packets at any time, the time the sender needs to wait for the receiver to wake up is randomly distributed in the cycle. This introduces randomness to packet delay in LPL.

2) Collection Tree Protocol: Collection Tree Protocol (CTP) [20] is used to build the routing tree in the network to collect data. CTP adopts the ETX metric [21], 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. In practice, the ETX is estimated by combining both the control plane and the data plane traffic. To reduce control traffic, CTP uses the Trickle timer [22] to control the beacon rate. When a node reboots or detects a loop, the timer interval decreases to its minimum (i.e., highest beaconing rate) in order to quickly find a good path. The timer interval increases exponentially to its maximum (about 8 minutes) in order to reduce unnecessary control plane traffic when the network is relatively stable.

C. Measurement Infrastructure

A naive approach to measuring delay in the network is to use network synchronization. For a synchronized network, all nodes in the network have agreed on a global time. Then we can record the global time of packet transmission at the source node and the global time of packet reception at the sink node, and thus calculate the delay of the packet as the time difference. However, commonly used network synchronization protocols in WSNs, such as FTSP [23], incurs additional traffic overhead into the network in order to frequently resynchronize and update the time stamps. Recently, more efficient synchronization protocols are proposed in [24] [25]. However, those protocols have special requirements on the network scale and network connectivity structure.

In our network, we use a light-weight approach to measure the end-to-end packet delay without incurring synchronization traffic. Our approach is based on the MAC layer time stamping technique (MLT) [23]. The MAC layer time stamp can precisely record the time when the packet is transmitted and received. A packet is time stamped, when it is being transmitted (with respect to sender's clock), and when the packet is received (with respect to the receiver's clock).

Our approach measures the end-to-end delay as follows. Figure 2 shows an example of 3 nodes from node 1 to node 3. Assume an event happens at node 1's local time t_1, we first measure the event time to node 2's and node 3's local clock. Suppose node 1 transmits a packet at t_2, with the event time t_1 contained in the packet. Then the packet is received at node 2 and time stamped at t_3 by MLT. Assume the propagation time is negligible, the receiver node 2 can calculate the event time with respect to its local clock, by substracting the time difference $t_2 - t_1$ from time t_3, i.e., $t_3 - (t_2 - t_1)$. Intuitively, it seems that the event happens at time $t_3 - (t_2 - t_1)$ to node 2's clock. Similarly, node 3 can calculate the event time after receiving a packet from node 2. Therefore, suppose a packet is transmitted (event) at time t_1 on the source node of a path. The sink node, e.g., node 3 in Figure 2, can calculate the transmitted time of the packet 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 $t_4 - (t_3 - (t_2 - t_1))$. Hereafter, we denote the packet transmission time to source's clock (t_1) as *SourceTime* of the packet, the receiving time of the packet at the sink node (t_5) as *SinkTime* of the packet and the packet transmitted time translated to the sink's clock $(t_5 - (t_4 - (t_3 - (t_2 - t_1))))$ as *SourceTimeAtSink*. Consequently, the delay can be calculated as

$$delay = SinkTime - SourceTimeAtSink. \quad (1)$$

Meanwhile, the source node can be synchronized to the sink's clock. Assume a widely used linear clock model [23] for sensor nodes, we have

$$SourceTimeAtSink = \alpha_1 SourceTime + offset_1 \quad (2)$$

where α_1 and $offset_1$ are the relative drift and offset. At time instant T_{source} in the source's clock, the corresponding SourceTimeAtSink T_{sink} is calculated by

$$T_{sink} = \alpha_1 T_{source} + offset_1. \quad (3)$$

According to Eq. 2, the time different between the SourceTimeAtSink and SourceTime is

$$\begin{aligned} offset_2 &= SourceTimeAtSink - SourceTime \\ &= \alpha_2 SourceTime + \beta \end{aligned} \quad (4)$$

where α_2 and β are the relative drift and offset for the SourceTimeAtSink and SourceTime.

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 minutes. There is a common header with four fields for those four types, which records a common sequence number, the SourceTime, SourceTimeAtSink and SinkTime of the packet, respectively. The content of payload for those four packets is as follows. *C1*: sensing data, including temperature, humidity and illumination; IDs of nodes on the path from the source to the destination. *C2*: IDs of neighboring nodes; RSSI, link quality and ETX to those neighbors; path ETX values to the sink through those neighbors. *C3*: accumulated counters of different events: MAC layer initial backoffs and congestion backoffs, radio on time and etc. *C4*: retransmission count and queue length at each hop along the path. In this paper, we analyze the one week data starting from Jul. 19, 2011.

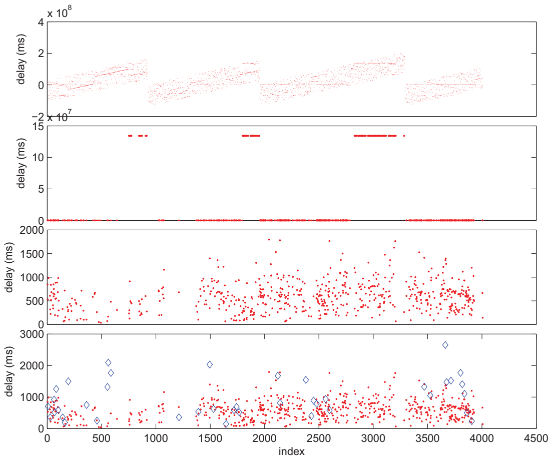

Fig. 3: Delay processing.

III. Delay Overview

In this section, we show the delay distribution overview for all nodes in the network.

A. Data Processing

We first present a robust delay processing method for the data. In a practical network, there are different types of errors. We need to cope with the errors in order to calculate the correct delay. To conquer those problems, the processing of delay consists of four steps.

1) Calculate Delay by Eq. 1. The calculated delay is shown in the top most figure in Figure 3. The delay is distributed in a very large range.
2) The first type of error of delay comes from the error of MLT. Due to packet overflow in the limited receiving buffer and packet losses, the MLT cannot guarantee to provide correct stamps. To address this problem, we validate the delay values and exclude the incorrect time stamps by the offset constraint. As in Eq. 4, we calculate the offset by $offset_2 = SourceTimeAtSink - SourceTime$. Assume $offset_2(t_1)$ and $offset_2(t_2)$ are two offsets calculated at time t_1 and t_2 where $t_1 < t_2$. For a maximum clock drift α_{max}, according to Eq. 4, the offsets should satisfy $|offset_2(t_2) - offset_2(t_1)| \leq (t_2 - t_1)\alpha_{max}$. Based on this constraint, we group the delays satisfying the offset constraint into the same group. Since incorrect delays are randomly distributed, we omit those groups with much less elements than other

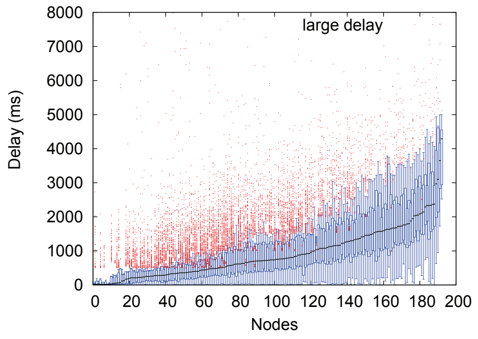

Fig. 4: Delay distribution for all nodes.

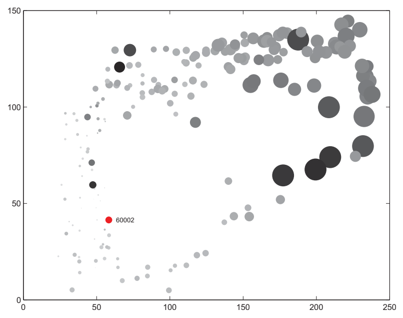

Fig. 5: Spatial distribution of sensor nodes.

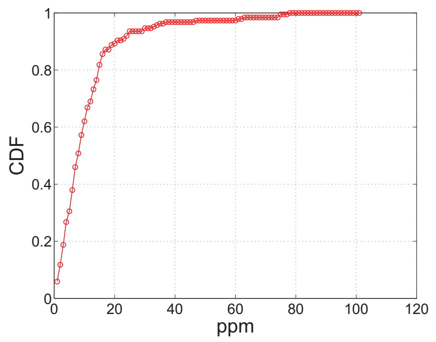

Fig. 6: CDF of drift for all nodes.

groups. The result is shown as the second subfigure in Figure 3.

3) The second type of error comes from the overflow of the time stamps. The SouceTimeAtSink provided by MLT is a 4-byte time stamp based on a 32KHz timer. Hence the maximum time for 0xFFFFFFFF/32 ms, i.e., about 1.5 days. To recover those delays, we investigate Eq. 1 and find the time stamp overflow causes some groups of delay with exceptionally large values. Therefore, we recover those delay values by subtracting the maximum time stamp from the delay. Note that this processing assumes most normal delays are less than the maximum overflow time (i.e., about 1.5 days). The result is shown in the third subfigure of Figure 3.
4) Till now, we have identified the delays for the correct time stamps. However, there still exist many incorrect delays. At this step, we recover those incorrect delays. First, for correct delays, we can obtain the linear model of $SourceTimeAtSink$ by calculating α_1 and $offset_1$ according to Eq. 2. Then, for an incorrect delay at SourceTime t_2, we can first calculate the SourceTimeAtSink by Eq. 2 and then the delay by Eq. 1. The rationale for this method is using the correct delays to synchronize the source and sink node, and then calculating the delay for two synchronized nodes. Similar to [23], the introduced error is at most $error = \alpha(t_2 - t_1)$, where α is the relative clock drift and t_1 is the latest SourceTime for the correct delay. If we require $error \leq \delta$, we only conservatively recover delays of packets at t_2 with $t_2 \leq t_1 + \delta/\alpha$.

The final result is shown in Figure 3. From the result, we can also find that the delay presents randomness in its distribution due to the LPL mechanism.

B. Delay Distribution

After calculating the delay values, we extract information from the data with randomness. First, we show the basic characteristics of delay distribution.

1) Overall Distribution: Normally, without other impacting factors, the maximum single hop delay is L according to the mechanism of LPL, where L is the cycle length of LPL, e.g., $L = 500ms$ 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 $kT/2$. Figure 4 shows the overall delay distribution in 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 by box plot 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 red dots in the figure represent delays of packets 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 range of different nodes varies, the median of the delay is relative stable. (3) There exist many large delays for most nodes. Later we will explain the reasons for those large delays.

2) Spatial Distribution: We further look at the spatial distribution of delays in the network. In Figure 5, each circle represents a sensor node and is plotted according to the physical location of the node. 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. The darker the color is, the larger variation the delay has. The red node (60002) is the sink node. We can see that nodes far away from the sink node have larger average delays as well as delay variations. Nodes in the right bottom area are farthest from the sink node and have the largest delays and delay variations.

3) Overall Clock Drift: Before examining the delay details, we also look at the clock drift of sensor nodes. To calculate the relative clock drift, as in [23], we first calculate the offset by Eq. 4 from the collected data. We apply robust linear fitting for the offsets and then calculate the slope as the relative clock drift for all nodes to sink node. Figure 6 shows the CDF of the relative clock drift. The x-axis is the drift and y-axis is the

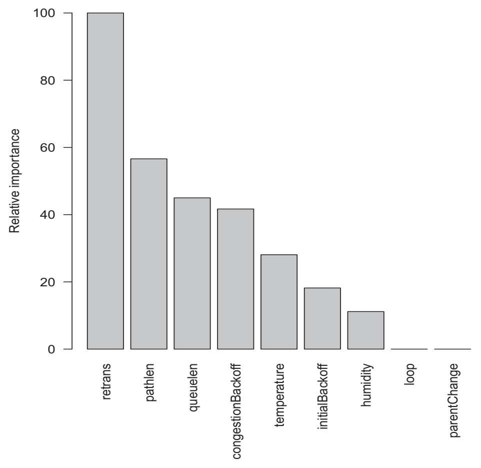

Fig. 7: Relative importance of the parameters.

CDF of nodes. More than 90% nodes have a clock drift less than 40 ppm. This coincides with the result from [26]. The result shows that the clock drift in the outdoor environment is relative stable and has a limited impact on the delay.

IV. Analysis of Different Factors

We have collected various parameters from the network. There are several challenges to identify the 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 makes it even difficult to extract those 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 [19] to find the important factors. Rulefit is a supervised learning approach to train predictors based on rule ensembles. 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., [27], to understand different impacting factors.

In Rulefit, each rule is a combination of one or more feature tests, i.e., combining one or more features into simple 'and' tests. Let x be the feature vector and s_f be a subset of the features. Then a rule takes the form of

$$r(x) = \prod_f I(x_f \in s_f) \tag{5}$$

where $I(\cdot)$ is an indicator function. A rule takes value one if all the feature tests in the rule take value one. To train the predictor, Rulefit first generates a large set of rules based on the decision tree. Then it trains the predictor by solving the optimization problem that minimizes the Huber loss. Finally, Rulefit provides the relative importance of the rules and features. The importance of the features is calculated based on the importance of rules containing the feature. Here we

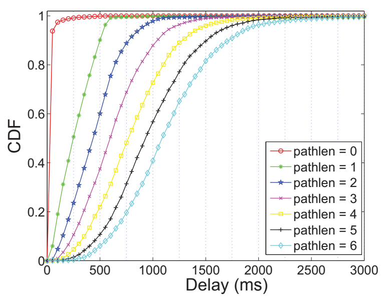

Fig. 8: Delay distributions to different hops.

present an overview of Rulefit approach. Interested readers can refer to [19] for more details.

We apply Rulefit to the collected parameters and corresponding delays. The ranking result of all parameters by Rulefit is shown in Figure 7. The top five most important factors are retransmission, hop count, queue length, congestion backoff and temperature. By applying Rulefit, we can filter those less important factors and then only focus on important factors for a real network. Later, we explain the result of Rulefit.

B. Detailed Correlation

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

1) Hop Count: Figure 8 shows the delay distribution with respect to different hop counts. Overall, packets with larger hop count to the sink node have larger delays. 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: Analysis in hop count shows that besides hop count, there exist other impacting factors to delay, which result in large delays. Figure 9 (a) shows the average retransmissions at each hop. We can see that retransmissions per packet at different hops are almost equal. Though there is more traffic for nodes near the sink, the corresponding retransmission count for those nodes is not much higher than other nodes. We investigate the data and find the collisions near the sink are not severe. Note that since the sink is always on, the retransmission count for nodes at hop 1 is much lower than other nodes.

Figure 9 (b) shows that the delay with different retransmission counts. It can be seen that retransmission count and delay have a similar trend. The delay increases as the retransmission count becomes larger. Further, we quantitatively 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.,

$$p_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}. \tag{6}$$

Pearson correlation is usually used as a measure of dependence between two variables.

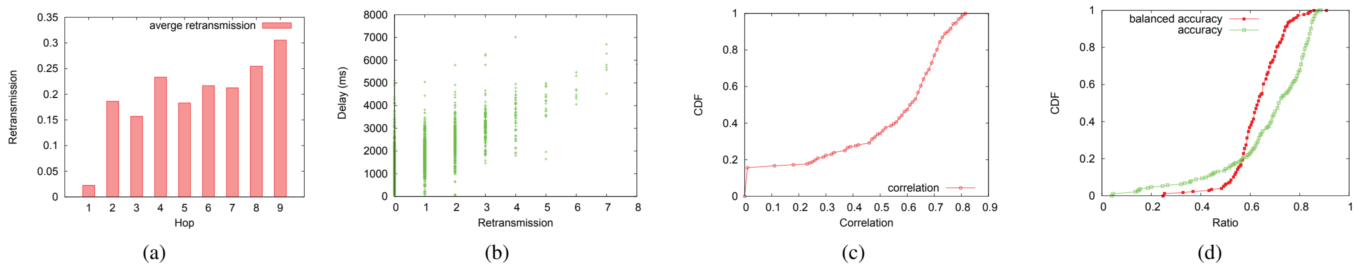

Fig. 9: Impact of retransmission to delay. (a) hop count to average retransmissions. (b) retransmission with respect to delay distribution. (c) CDF of correlation between retransmission and delay for all nodes. (d) accuracy and balanced accuracy of using retransmission to predict large delay.

For each node, we calculate the Pearson correlation for retransmission and delay. Then we show the CDF of correlations for all nodes. As in Figure 9 (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.

Now we investigate the data and particularly look at the large delays. For hop k, we denote the delays larger than $k \times 500$ as large delays and other delays as normal delays. Meanwhile, 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 that how retransmission events can predict large delays. For a packet with 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 predict a large delay, i.e.,

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN}. \tag{7}$$

Figure 9 (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 very high. More than 80% nodes have a accuracy ratio higher than 60%, indicating that retransmission event is a good predictor for large delays.

Considering an extreme case when all packets have large delays. In such a case, even randomly selecting a large enough subset of packets as retransmission events would lead to a high accuracy. However, in this case, the retransmission events cannot predict large delays. To address such a problem, as in [28], we calculate the *balanced accuracy*, i.e.,

$$balanced\ accuracy = \frac{0.5 \times TP}{TP + FN} + \frac{0.5 \times TN}{TN + FP}. \tag{8}$$

Figure 9 (d) shows the CDF of balanced accuracy. We can see that for balanced accuracy, more than 70% nodes have a balanced accuracy larger than 0.6, validating that retransmission is a good predictor to large delay.

3) Queuing: We also examine the impact of queuing to the delay performance. For each packet, at each hop along the path to sink node, we record in the packet the queue length when the packet arrives at each node. Figure 10 (a) shows the average queue length for different hops. Unlike the result for retransmission count, the average queue length varies for different hops. Nodes near the sink have larger average queue length. This indicates that though high traffic near the sink does not incur packet losses and retransmissions, it results in more congestions and thus larger queue length. Since the sink is always on, nodes within 1 hop from sink node can quickly drain their packets and thus the queue length is smaller.

Figure 10 (b) shows the delay with respect to the total queue length along the path. We also find that many packets with large queue length are correlated to large delays. We also calculate the correlation between total queue length and delay. Figure 10 (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 a positive correlation with the total queue length, but not as strong as retransmission. In LPL, the receiver is awake with high probability after the first packet in the queue. Thus consecutive packets may not need to wait until the receiver wakes up. Thus the impact of queue length to delay is relative small. We further explain the reason in the delay model in Section V.

To examine the predictability of queue length to large delay, similar to retransmission, we set a packet with 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 Figure 10 (d). First, we can find that on most nodes, queuing events are correlated to large delays with a relative high probability, e.g., about 80% nodes have an accuracy higher than 60%. For balanced accuracy, about 80% nodes have a balanced accuracy higher than 40%. In total, we can see that queuing event is a good predictor for large delay. Second, the correlation here is not as high as for retransmission. This coincides with the result of Figure 10 (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 less important factors to delay in our network.

It has been shown in [26] [29] that environment factors such as temperature and humidity affect the clock drift and link quality. Figure 6 shows the drift of sensor nodes are 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 [29]. Link quality is indeed

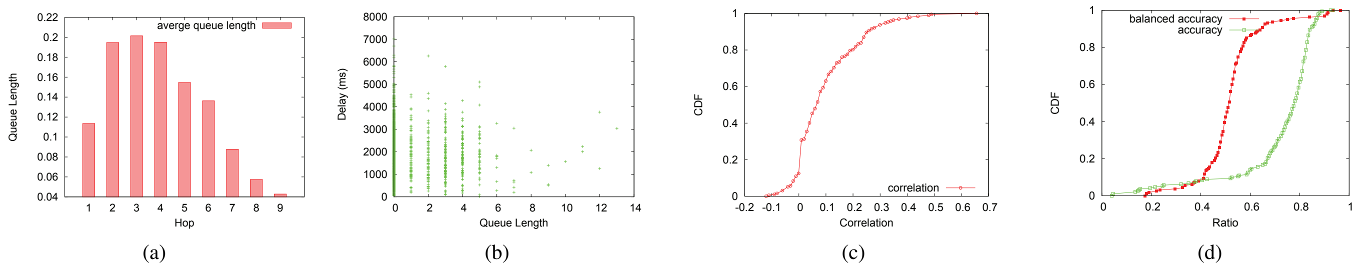

Fig. 10: Impact of queuing events to delay. (a) queue length to average retransmissions. (b) queue length with respect to delay distribution. (c) CDF of correlation between queue length and delay for all nodes. (d) accuracy and balanced accuracy of using queuing events to predict large delay.

related to delay performance. Such an impact is captured in retransmissions. Thus we do not consider environment as a direct impact factor.

The impact of MAC backoffs to delay is also extensively studied in different works such as [13]. However, the impact in LPL network is not as important as that in an always-on network. In LPL implementation, preambles are transmitted in order to wake up the receiver. Since the preambles always last until the receiver wakes up, the backoffs of preambles do not impact the delay. More backoffs do not increase the delay. Besides, backoffs of unsuccessful transmissions do not impact the delay as well. The delay for such kind of transmissions is equal to the cycle length in LPL. Therefore, backoffs have a relative smaller impact compared with other factors.

For routing events such as parent change and loop events, works such as [30] have studied the impact of those parameters on the delay performance in Internet. It has been shown that those events may lead to a large end-to-end delay [30]. However, different from Internet, our result shows that the impact is relatively small in WSNs. We find that there are several reasons. First, the number of such events is relative small. Second, nodes often switch among parents with similar hop counts. Third, those events, such as loop events, can be quickly recovered by the network protocol. For example, in CTP, when a loop is detected, the beacon interval of routing protocol is decreased to minimum in order to propagate the information as quickly as possible, alleviating the impact of loop event to end-to-end delay.

V. Delay Model

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

A. Model

We first model the packet delay for a single hop and then we extend it to the multi-hop end-to-end delay. To derive the model, we first 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., $\frac{t_w}{t_w+t_s}$.
- 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 time for a single hop packet transmission. Assume the packet is retransmitted for r times. According to the mechanism of LPL, each unsuccessful transmission will consume $t_w + t_s$ time. Thus the time used for r retransmissions is $r(t_w + t_s)$. For the $(r + 1)th$ transmission which 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.
- **Case 2**: otherwise, the packet can be sent without preambles to wake up the receiver.

For case 1, since the transmission can falls into any time during the sleep time of the receiver, the delay is $U(0, t_s) + t_b + t_x$, where $U(0, t_s)$ is a random distribution between 0 and t_s. For case 2, the delay for a packet is $t_b + t_x$. 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 in LPL is given by:

$$T(t_s, t_w, r) = \begin{cases} r(t_w + t_s) + t_b + t_x & \text{with prob. } u \\ r(t_w + t_s) + U(0, t_s) + t_b + t_x & \text{otherwise} \end{cases} \quad (9)$$

where $U(0, t_s)$ is a uniform distribution function between 0 and t_s.

Based on the 1-hop delay, we derive the delay for a multi-hop 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 multi-hop 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 all the prior $l_i - 1$ packets are transmitted.
- $r_{i,j}$: the number of retransmissions for the j-th packet in the queue on node i.

The time for the first packet in the queue can be calculated according to Eq. 9. We then calculate the time for following packets in the queue. In LPL, the receiver may be awake after the first 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 which takes time $t_w + t_s$ and the

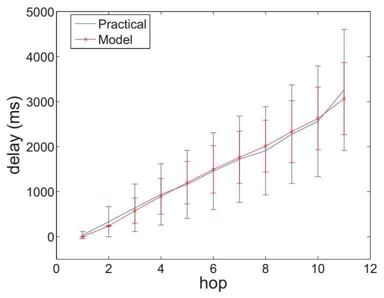

Fig. 11: Delay distribution of the model and real collected data with respect to different hop counts.

receiver is no longer awake. Thus for packets except the first one in the queue, the delay is given by

$$T_q(t_s, t_w, r) = \begin{cases} t_b + t_x & r = 0 \\ T(t_s, t_w, r) & \text{otherwise} \end{cases} \tag{10}$$

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

$$D(n) = \sum_{i=1}^{n-1} T(t_s^{i+1}, t_w^{i+1}, r_{i,l_i}) + \sum_{i=1}^{n-1} \sum_{j=1}^{l_i - 1} T_q(t_s^{i+1}, t_w^{i+1}, r_{i,j}) \tag{11}$$

where $T(t_s^{i+1}, t_w^{i+1}, r_{i,l_i})$ is the single hop transmission time for the first packet in the queue at node i, $T_q(t_s^{i+1}, t_w^{i+1}, r_{i,j})$ is the transmission time 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 network, we have recorded the queue length on each node of the path, i.e., l_i. From C2 packet, we have recorded the ETX value, which is the expected transmission count, from each node to its neighbors. This can be used for $r_{i,j}$, for $j > 1$. For each packet, we also recorded the retransmissions on each hop. This can be calculated as $r_{i,j}$, for $j = 1$. On each node, by the recorded information in C3 packet, we calculate the average duty cycle u and average backoff time t_b. Using those parameters as the input, we calculate the delay with the model. Then we compare the result of the model with the practical delay calculated from the packets. The result in Figure 11 shows that both the delay in the model and practical delay 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.

VI. Implications

The analysis of delay and the model can be used to direct the protocol design in WSNs. In this section, we revisit three prevalent protocols and discuss the implications of our analysis and delay model to those protocols.

A. Collection Tree Protocol

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

$$D(n) = \sum_{i=1}^{n-1} T(t_s^{i+1}, t_w^{i+1}, r_{i,1})$$

According to Eq. 9, the expected delay is

$$E(D(n)) = E(\sum_{i=1}^{n-1}(r_{i,1}(t_w + t_s) + t_b + t_x)) + (1-u)E(\sum_{i=1}^{n-1} U(0, t_s))$$

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

$$\begin{aligned} E(D(n)) &= \sum_{i=1}^{n-1}((ETX_i - 1)(t_w + t_s) + t_b + t_x) + (1-u)(n-1)\frac{t_s}{2} \\ &= PathETX(t_w + t_s) - (n-1)(t_w + \frac{1+u}{2}t_s - t_b - t_x). \end{aligned} \tag{12}$$

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 $(n-1)(t_w + \frac{1+u}{2}t_s - t_b - t_x) > 0$. According to Eq. 12, for the same path ETX, $E(D(n))$ decreases as the hop count n increases. This indicates that longer paths, which are often prohibited, may even be better than shorter ones.

Consider a simple example with two paths, the first path consists of only one link with link ETX 2; the second path consists of two links, each of which has a link ETX 1. In CTP, both those two paths have path ETX value 2. CTP treats those two paths equally and thus randomly chooses one as the routing path (in the current implementation, the one 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 it has a larger hop count. For the first path, there are 2 transmissions in expectation including 1 retransmission. For the second path, there are 2 transmissions in expectation without retransmissions. Intuitively, according to the mechanism of LPL, if a packet is successful transmitted, the expected time is $L/2$, where L is the cycle in LPL. However, if a packet is retransmitted, the expected time is L until the sender realizes the unsuccessful transmission. Hence, the second path is better. This also explains in Figure 7 why retransmission count is more important than hop count. Each retransmission accounts for L time, while a successful transmission only accounts for $L/2$ in expectation. In the current design, the ETX metric only counts the expectation of transmissions and makes no difference for retransmissions and successful transmissions. The analysis result shows the current CTP design can be improved by incorporating the delay model.

B. Opportunistic Routing

Based on the collected data, we then examine the widely adopted opportunistic routing techniques. In traditional routing, a sender often has one fixed forwarder to relay all the data packets. In opportunistic routing e.g., [31], considering the broadcast nature of wireless signal, a packet may be opportunistically forwarded by other nodes instead of the fixed

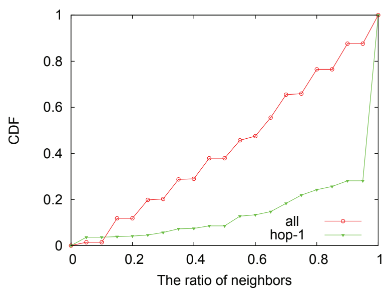

Fig. 12: Evaluation of opportunistic routing

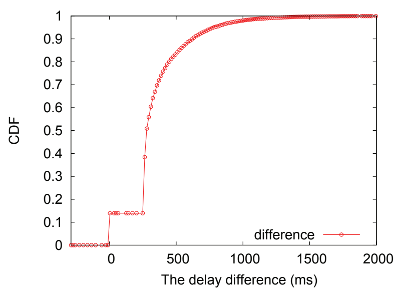

Fig. 13: Delay difference the practical packet delay and DSF.

forwarder. It has been shown in [32] [33] that opportunistic routing can significantly improve system performance. A problem in opportunistic routing is that the improvement is not guaranteed to opportunistically select a forwarder, e.g., in terms of delay. The opportunistic forwarder may even have a much larger delay than the fixed forwarder.

We evaluate the practical performance of opportunistic routing with real data and investigate how to improve it. For all neighbors of each node, we calculate the delay from each neighbor to the sink node. Since we consider opportunistic routing, we omit the delay from the sender to the neighbors. At the first step, for each node, we calculate the portion of neighbors with smaller delay than the node itself, which gives the probability that a node can benefit by opportunistically selecting a forwarder from its neighbors. The result is shown in Figure 12. We can see that a node can benefit with a high probability by opportunistic routing. For more than 50% nodes, the probability to select a better node is higher than 50%. Opportunistic routing with all neighbors may select a forwarder with a larger delay.

Further, we investigate possible improvement based on the collected data. For each node, we only use neighbors with smaller hop count to sink node than itself. We calculate the portion of nodes with smaller delay. We can see from Figure 12 that it is better to select nodes from neighbors with smaller hop count to sink node. For more than 80% nodes, the probability to select a node with smaller delay is larger than 95%.

C. Dynamic Switching based Forwarding

To improve the data delivery performance, in many data forwarding applications, instead of selecting a single node as a forwarder, a node will select a set of nodes as forwarders. For example, Dynamic switching based forwarding (DSF) [34] 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 a optimal forwarding set with minimum expected end-to-end delay. More specifically, a node searches through each neighbor and calculates whether it is beneficial to add this node into the forwarding set. This is different from opportunistic routing. In opportunistic routing, the forwarder is opportunistically selected. In DSF, once the forwarding set is calculated, the sender sends packet to nodes in the set until the packet is successfully received by one node.

In this paper, we evaluate DSF in a practical network. For each node, we calculate its link quality q_i to each neighbor i and the delay T_i of neighbor i. According to DSF, 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. Then in one cycle, the expected delay is

$$D_1 = \sum_{i=1}^{|F|} \prod_{k=0}^{i-1} (1 - q_k) q_i T_i, \tag{13}$$

where $q_0 = 0$. If all nodes in F fail to receive packet in one cycle, DSF takes a second round and repeatedly tries nodes in F until the packet is successfully received by one forwarder. If the transmission succeeds in the jth round, the delay is

$$D_j = (j - 1)L + D_1. \tag{14}$$

Accordingly, the expected delay is

$$E(D) = \sum_{j=1}^{+\infty} p_f^{j-1} (1 - p_f) D_j \tag{15}$$

where $p_f = \prod_{i=1}^{|F|}(1 - q_i)$ is the probability that transmissions to all nodes in the forwarding set fail in one round. As in DSF, we use dynamic programming to find F such that $\arg_F minE(D)$ and calculate the corresponding delay $E(D)$. We compare the delay of DSF with the practical packet delay. Figure 13 shows the delay difference of practical delay and DSF delay, i.e. practical delay$-$DSF delay. First, for most packets, the difference is larger than 0. We can see that the practical delay for more than 75% packets 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 that if a forwarder cannot be found in one cycle, we can use other approaches since DSF may not improve the performance.

VII. Related Work

There are extensive studies of delay performance analysis in WSNs. Those studies in WSNs can mainly be divided into two categories. The first category is to provide delay bounds. For example, probabilistic delay bounds are proposed in [3] [4] [5] [6] by extending network calculus. However, worst case delay bounds often deviate from practical delays in WSNs. In the second category, stochastic delay models are proposed. For example, in [7] [8] [9], different models are proposed by combining real-time theory and queuing theory. In those models, unreliable networks with heavy traffic are considered. Heavy traffic does not always hold in real WSNs and hence those models are not applicable. There are some empirical network delay models [11] [12] proposed for end-to-end delay measurements. A delay model based on Discrete Markov Processing in the network is proposed in [10]. Besides those end-to-end delay models, the single hop channel access delay models are also analyzed in [13] [14] [15]. However, those works proposed in WSNs are often based on assumptions, e.g., traffic, routing path, and not evaluated in a real large-scale network. In this work, we are the first to propose a light-weight delay measurement method and validate our model in a large-scale WSN.

There are also a large number of research works in Internet and data centers. Pucha et al. [30] show the impact of routing events to end-to-end delay in Internet. It has been shown that routing events such as parent change may have a large impact on delay performance. Kompella et al. [16] present fine grain latency measurements in presence of packet losses for internet with a lossy difference aggregator. By leveraging this method, delay for every packet in internet can be measured while incurring very limited additional traffic overhead. To measure the per-flow delay, Lee et al. [17] present a measurement method with reference delay interpolation. This work further extends existing works to efficiently measure per-flow delay. As the development of data center technologies, Wilson et al. [18] present delay analysis results in the data center.

VIII. Conclusion

We present the first comprehensive delay performance analysis in an operational large-scale urban wireless sensor network. Through carefully examine system metrics, we show the spatial and temporal characteristics of delay distribution. To combat with the incomplete data in presence of randomness, we leverage Rulefit to rank different parameters and find the most important impacting factors. Accordingly, we propose a practical delay model and validate it in a large-scale network. We revisit commonly used protocols with real data. In the future, we will investigate more detailed correlations between different parameters to the delay performance.

Acknowledgments

We would like to thank the anonymous reviewers and our shepherd for providing valuable comments. This work is supported in part by the NSFC Major Program 61190110, NSFC under grant 61202359, Fundamental Research Funds for the Central Universities (2012QNA5007) and the National Basic Research Program of China (973) under grant No. 2012CB316200.

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 *Proceedings of ACM SenSys*, 2010.

[2] S. N. Pakzad, G. L. Fenves, S. Kim, and D. E. Culler, "Design and implementation of scalable wireless sensor network for structural monitoring," in *Journal of Infrastructure Systems*, 2008.

[3] M. Fidler, "An end-to-end probabilistic network calculus with moment generating functions," in *Proceedings of IEEE IWQoS*, 2006.

[4] A. Koubaa, M. Alves, and E. Tovar, "Modeling and worst-case dimensioning of cluster-tree wireless sensor networks," in *Proceedings of IEEE RTSS*, 2006.

[5] A. Burchard, J. Liebeherr, and S. Patek, "A min-plus calculus for endtoend statistical service guarantees," *IEEE Transactions on Information Theory*, vol. 52, no. 9, pp. 4105–4114, 2006.

[6] J. Schmitt and U. Roedig, "Sensor network calculus - a framework for worst case analysis," in *Proceedings of IEEE DCOSS*, 2005.

[7] J. Lehoczky, "Real-time queueing theory," in *Proceedings of IEEE RTSS*, 1996.

[8] J. P. Lehoczky, "Real-time queueing network theory," in *Proceedings of IEEE RTSS*, 1997.

[9] S.-N. Yeung and J. Lehoczky, "End-to-end delay analysis for real-time networks," in *Proceedings of IEEE RTSS*, 2001.

[10] Y. Wang, M. C. Vuran, and S. Goddard, "Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 305–318, 2012.

[11] 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 *Proceedings of IEEE INFOCOM*, 2005.

[12] K. Gopalan, T.-c. Chiueh, and Y.-J. Lin, "Probabilistic delay guarantees using delay distribution measurement," in *Proceedings of ACM MULTIMEDIA*, 2004.

[13] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[14] T. Sakurai and H. Vu, "Mac access delay of ieee 802.11 dcf," *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1702–1710, 2007.

[15] O. Tickoo and B. Sikdar, "Modeling queueing and channel access delay in unsaturated ieee 802.11 random access mac based wireless networks," *ACM/IEEE Transactions on Networking*, vol. 16, no. 4, pp. 878–891, 2008.

[16] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese, "Every microsecond counts: Tracking fine-grain latencies with a lossy difference aggregator," in *Proceedings of ACM SIGCOMM*, 2009.

[17] M. Lee, N. G. Duffield, and R. R. Kompella, "Not all microseconds are equal: Enabling per-flow measurements with reference latency interpolation," in *Proceedings of ACM SIGCOMM*, 2010.

[18] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better never than late: Meeting deadlines in datacenter networks," in *Proceedings of ACM SIGCOMM*, 2011.

[19] J. Friedman and B. Popescu, "Predictive learning via rule ensembles," *Annals of Applied Statistics*.

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

[21] D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of ACM MobiCom*, 2003.

[22] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of USENIX NSDI*, 2004.

[23] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi, "FTSP: The Flooding Time Synchronization Protocol," in *Proceedings of SenSys*, 2004.

[24] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient Network Flooding and Time Synchronization with Glossy," in *Proceedings of ACM/IEEE IPSN*, 2011.

[25] Y. Wang, Y. He, X. Mao, Y. Liu, Z. Huang, and X. Li, "Exploiting constructive interference for scalable flooding in wireless networks," in *Proceedings of IEEE INFOCOM*, 2012.

[26] 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 *Proceedings of the ACM SIGMETRICS*, 2008.

[27] I. Cunha, R. Teixeira, D. Veitch, and C. Diot, "Predicting and tracking internet path changes," in *Proceedings of ACM SIGCOMM*, 2011.

[28] 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 *Proceedings of OSDI*, 2004.

[29] 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 *Proceedings of ACM SenSys*, 2010.

[30] H. Pucha, Y. Zhang, Z. M. Mao, and Y. C. Hu, "Understanding network delay changes caused by routing events," in *Proceedings of ACM SIGMETRICS*, 2007.

[31] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proceedings of SIGCOMM*, 2007.

[32] S. Biswas and R. Morris, "Exor: Opportunistic multi-hop routing for wireless networks," in *ACM SIGCOMM*, 2005.

[33] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: opportunistic routing meets duty cycling," in *Proceedings of ACM/IEEE IPSN*, 2012.

[34] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, pp. 1741–1754, 2011.