Delay Measurement in Sensor Networks Using Passive Air Monitoring

Zhengming Bu, Bing Wang and Zhijie Shi
Computer Science & Engineering Department
University of Connecticut, Storrs, CT 06269

UCONN CSE Technical Report BECAT/CSE-TR-07-07

Abstract

Wireless sensor networks have been used in many applications that have realtime requirements, e.g., emergency response and medical care. For such applications, measuring per-hop and end-to-end delays inside a network is important for network management. Delay measurement in sensor networks is, however, a challenging problem due to the difficulties to synchronize the clocks at the sensor nodes. In this paper, we develop a methodology that uses passive air monitoring to measure per-hop and end-to-end delays in a sensor network. Our method does not generate any additional traffic in the network and requires no clock synchronization. Using this method, we characterize delays in a sensor-network testbed under various settings, e.g., with different network topologies and amount of medium contention.

I. Introduction

Wireless sensor networks have been used for many applications that have realtime requirements, e.g., emergency response, plant automation and control, and health care. In a sensor network for realtime applications, measuring end-to-end delay (i.e., from a sensing node to the end user) and per-hop delay (i.e., delay on each hop of the end-to-end path) is important for network management: end-to-end delays can be used to evaluate the performance of the network and detect paths with excessive delays; per-hop delays can be used to pinpoint the hops that cause large end-to-end delays.

One way (as often used in a wired network) to obtain the end-to-end delay from node s to node t is to first synchronize the clocks at these two nodes. Then node s places a timestamp into a packet when sending it, node t timestamps the packet when receiving it, and the difference of these two timestamps is an instance of transmission delay from s to t. Time synchronization is, however, a challenging task in large-scale sensor networks. Although numerous solutions have been proposed (see survey [13] and the references therein), they typically require (a large number of) message exchanges, which consume the scarce energy of the sensor nodes. One way to eliminate the need for time synchronization is using half of the RTT between s and t as the one-way transmission delay from s to t. However, this may lead to inaccurate estimates given the asymmetric communication in sensor networks [7], [15], [11].

In this paper, taking advantage of the broadcast medium in wireless sensor networks, we propose using *passive air monitoring* (simply referred to as *air monitoring* henceforth) for delay measurement. The main idea of air monitoring is to place a set of dedicated sensor nodes, called *air monitors*, inside a sensor network. Each air monitor captures and decodes MAC-layer packets in its neighborhood. The captured data can be analyzed locally or transmitted using other communication channels (e.g., Bluetooth in [6]) to a central server. Air monitoring has the advantage that it does not generate any additional traffic in the network. Furthermore, as we shall see (Section II), it provides a convenient technique to obtain both end-to-end and per-hop delays.

Our paper makes two main contributions. Firstly, we develop a methodology for per-hop and end-to-end delay measurement using air monitoring. Secondly, we use the methodology to characterize delays in a sensor-network testbed under various settings. We find that, for instance, in a setting with light load and no medium contention, per-hop delays are mostly in $[6, 14]$ ms with the mean of 9 to 11 ms; when two nodes compete for medium, one node can suffer from significantly larger delays, and the delays are in a much wider range ($[6, 70]$ ms) with the mean as high as 23 ms. To the best of our knowledge, our study is the first on delay measurement and characterization in a sensor network.

As related work, air monitoring has been successfully utilized in wireless LANs (WLANs) for network management and characterization (e.g., [2], [14], [8], [3], [5], [9], [4], [12]). A sensor network differs from a WLAN in that it is a multi-hop ad-hoc network while a WLAN is a single-hop (i.e., from a wireless host to an access point) network with infrastructure support. We are aware of only one study on air monitoring in sensor networks [6]. This study uses air monitoring for code debugging, which differs from our focus on delay measurement.

The rest of the paper is organized as follows. In Section II, we describe our measurement methodology. In Section III, we describe our experiment methodology. In Section IV, we characterize delays in a sensor-network testbed. Finally, Section V concludes the paper and presents future work.

II. Measurement Methodology

We first present the problem setting and then describe our methodology to obtain per-hop and end-to-end delays.

A. Problem setting

Consider an arbitrary source s inside a sensor network. This source transports the sensed data to sink t. Suppose there are k intermediate nodes along the path from source s to sink t. We denote these nodes as $n_1, \ldots, n_k$. For convenience, we also refer to source s as node n_0 and refer to sink t as node n_{k+1}. We represent the path as $(s = n_0, n_1, \ldots, n_k, n_{k+1} = t)$. The hops along the path are indexed from 1 to $(k + 1)$. That is, the i-th hop is the hop (n_{i-1}, n_i). On hop (n_{i-1}, n_i), we refer to node n_i as node n_{i-1}'s *parent* and node n_{i-1} as node n_i's *child*.

Let D_i denote the delay on the i-th hop, $i = 1, \ldots, k + 1$. It contains two components: the delay at node n_{i-1} and the radio propagation delay for a packet to reach node n_i from n_{i-1}. We ignore the latter since it is negligible (the transmission range in a sensor network is tens or hundreds of meters while the radio propagation speed is approximately 3×10^8 meters per second). Therefore, D_i is simply the delay at node n_{i-1}. The delay at source s is from when it sends the packet at the application level to when the sending process is completed. At an intermediate node n_i, the delay is from when the node receives a packet from its child to when the sending process is completed, $i = 1, \ldots, k$. Let D denote the end-to-end delay from source s to sink t. It is the delay from when the source sends a packet at the application-level to when the packet reaches the sink. We compute D by adding up the delays over all the hops, that is, $D = \sum_{i=1}^{k+1} D_i$.

We will describe a method that uses air monitoring to obtain per-hop and end-to-end delays. A sensor network with air monitoring is illustrated in Fig. 1, where the shaded nodes are air monitors. Each air monitor captures MAC-level packets in its neighborhood and records the time when a packet is captured. We assume that the air monitors are placed in such a way that, for each hop, there exists at least one air monitor that captures the transmissions from both nodes on the hop. Using air monitoring for delay measurement eliminates the need to synchronize the clocks

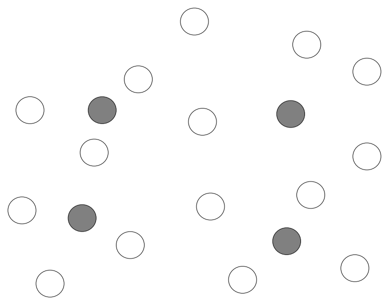


Fig. 1. Air monitoring in a sensor network. The shaded nodes are air monitors.

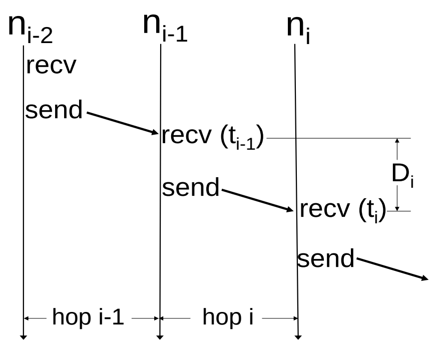


Fig. 2. Obtaining delay on the i-th hop, $i \geq 2$ (no packet retransmission).

at the sensor nodes. This is because an air monitor captures the transmissions from multiple nodes and timestamps these transmissions according to the same clock, i.e., that of the air monitor. We next describe our methodology to obtain per-hop and end-to-end delay using air monitoring in Sections II-B and II-C.

B. Per-hop delay measurement

We describe how to obtain per-hop delay using air monitoring in two cases: (i) no packet retransmission, and (ii) with packet retransmission. In the first case, a node (the source or an intermediate node) only transmits a packet once. In the second case, a node retransmits the packet for up to R ($R \geq 2$) times if not receiving an ACK from its parent (all ACKs in this paper refer to MAC-level ACKs).

1) No packet retransmission: We now describe how to obtain per-hop delay without packet retransmission. Consider an arbitrary packet and three adjacent nodes n_{i-2}, n_{i-1}, and n_i, $i \geq 2$, as illustrated in Fig. 2. In the figure, the vertical lines represent time. Let t_i denote the time when node n_i receives the packet from its child n_{i-1}, $i = 1, \ldots, k+1$. Since we ignore radio propagation delay, t_i is also the time when node n_{i-1} finishes transmitting the packet. Therefore, the delay on the i-th hop $D_i = t_i - t_{i-1}$, since t_{i-1} is the time when n_{i-1} receives the packet and t_i is the time when n_{i-1} finishes transmitting the packet. This delay can be obtained easily through air monitoring. Suppose the air monitor captures the packet from node n_{i-1} to n_i at time T_i (according to its local clock). Then T_i is the air monitor's local time corresponding to t_i. Since D_i is determined by the relative difference between t_i and t_{i-1}, even if the air monitor's local time is not set to the correct wall clock time, we still have $t_i - t_{i-1} = T_i - T_{i-1}$. Therefore, D_i can be obtained from air monitoring as $D_i = T_i - T_{i-1}, i = 2, \ldots, k+1$.

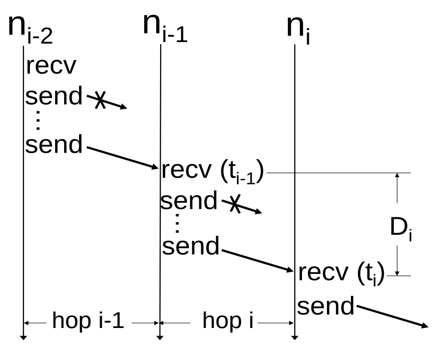


Fig. 3. Obtaining delay on the i-th hop, $i \geq 2$ (with packet retransmission).

The above approach obtains the delay on the i-th hop and requires that $i \geq 2$. It is not applicable to the first hop since the air monitor does not know when the source sends the packet at the application level. We use the following method to obtain the delay on the first hop (s, n_1). Source s records a timestamp T_s^b when it is about to send the packet at the application level, and records another timestamp T_s^e after finishing transmitting the packet. Then the first-hop delay of the packet is $T_s^e - T_s^b$. To pass this first-hop delay to the air monitor, source s embeds this delay in the payload of the next packet (we assume that the source transmits a sequence of packets) and the air monitor extracts the delay from the payload.

A natural question is: why not apply the method for the first hop to later hops? That is, the child node n on a later hop records the time when it receives a packet as T_n^b, records the time when it finishes transmitting the packet as T_n^e, and then embeds $T_n^e - T_n^b$ in later packets. This approach has the following drawback. It requires the forwarder to record times (i.e., T_n^b and T_n^b) and embed the calculated delay in later packets, which adds overheads in the forwarding process. When there are a large number of forwarders along a path, the aggregate overheads at the forwarders may become non-negligible. The approach that uses air monitoring is more convenient and generates no additional overheads.

2) With packet retransmission: We now describe how to obtain per-hop delay when a packet can be retransmitted. In particular, we assume that each node is allowed to transmit the packet for up to R ($R \geq 2$) times when not hearing an ACK from its parent. In this case, the delay on the first hop can be obtained in the same manner as that in Section II-B.1. For the delay on a later hop, since the air monitor may capture multiple transmissions of a packet from a node (i.e., when the packet is retransmitted), it needs to know which transmission reaches the parent node successfully. This can be solved using the DSN (Data Sequence Number) field as follows. As specified in the 802.15.4 standard [1], the MAC header of each packet contains a DSN field. A data packet and its corresponding ACK have the same DSN value. Therefore, the DSN field can be used to match a data packet and its corresponding ACK. Based on DSN, the air monitor identifies the successful transmission as follows. Suppose the air monitor captures multiple transmissions of a packet from node n_{i-1} to n_i. A successful transmission triggers an ACK from n_i to n_{i-1}, and the air monitor captures the ACK. Then the air monitor determines the successful transmission from the DSN of the ACK (the DSN of the successful transmission must match the DSN of the ACK). Once the successful transmission is determined, we can obtain the delay on the i-hop in a similar manner as in Section II-B.1, $i \geq 2$. More specifically, consider an arbitrary packet. Let t_i denote the time when node n_i receives the packet

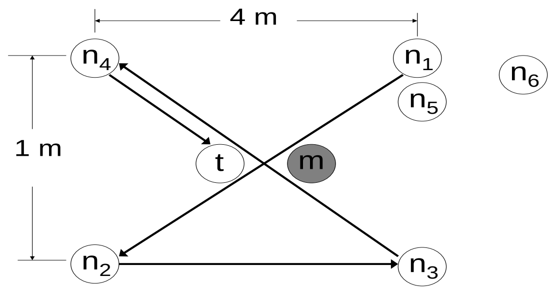


Fig. 4. Testbed setting: nodes t and m serve as the sink and air monitor respectively.

from node n_{i-1} successfully. Let T_i be the air monitor's local time corresponding to t_i, $i = 1, \ldots, k+1$. Then $D_i = t_i - t_{i-1} = T_i - T_{i-1}$ as illustrated in Fig. 3, $i = 2, \ldots, k+1$.

C. End-to-end delay measurement

We now describe how to obtain the end-to-end delay from source s to sink t. Let us consider two scenarios. In the first scenario, a single air monitor captures the transmissions from all the nodes on the path. In the second scenario, multiple air monitors are used to capture the transmissions from all the nodes on the path. In both scenarios, we consider an arbitrary packet.

In the first scenario, we have $D = \sum_{i=1}^{k+1} D_i = D_1 + \sum_{i=2}^{k+1} D_i = D_1 + \sum_{i=2}^{k+1}(T_i - T_{i-1}) = D_1 + T_{k+1} - T_1$, where D_1 is the first-hop delay (it is embedded in the next packet from the source, see Section II-B.1), T_i is the timestamp that the air monitor records when it captures the packet transmitted from node n_{i-1} to n_i (when allowing retransmission, it corresponds to the successful transmission).

In the second scenario, we segment the path into multiple sub-paths so that a single air monitor captures the transmissions from all nodes on a sub-path. Then, we obtain the end-to-end delay on each sub-path in a similar manner as in the first scenario; and the end-to-end delay of the entire path is the sum of the end-to-end delays on each sub-path.

III. Experiment methodology

We apply the delay measurement methodology to measure per-hop and end-to-end delays in a sensor-network testbed. Our goal is to characterize delays under both desirable and undesirable conditions. In the following, we first describe our experiment methodology in detail.

A. Testbed setting

Our testbed is placed in an office and consists of eight TelosB motes, as illustrated in Fig. 4. Node t is a sink. Node m is an air monitor. It captures packet transmissions from all the nodes and records the corresponding timestamps (with the granularity of 1 ms). Node n_6 is used to synchronize the sources so that they send packets at approximately the same time (as we shall describe soon). The rest of the nodes are sources and/or forwarders. The power level at a mote is set to 3, i.e., -25 dBm. The distances between two motes are in meters, as marked in Fig. 4. Under the above power level and node distances, the channel between two motes is in an ideal condition.

We conduct experiments under three settings. The first setting contains a single sender, n_1, which sends packets via nodes n_2, n_3, and n_4 to sink t. This setting represents a desirable operation environment, with light load and no contention in the medium. In the second setting, nodes n_1 and n_5 are sources, both sending packets via nodes n_2, n_3, and n_4 to sink t. This represents a setting where contention may occur since the two sources may send to node n_1 simultaneously. We refer to this setting as one with *parallel sources*. In the third setting, nodes n_1 and n_2 are sources and share part of their routes: n_1 sends its packets via nodes n_2, n_3, and n_4 to sink t, while n_2 sends its packets via nodes n_3 and n_4 to sink t. This represents a setting in which a sensor node, such as node n_2 in our setting, can be both a forwarder and a source. Contention may also occur in this setting when both sources transmit simultaneously. We refer to this setting as one with *tandem sources*.

In a settings with parallel or tandem sources, when purposely making the sources send at different times, we observe similar delay characteristics as those in the single-source setting. Therefore, in the following, we focus on the scenarios when the sources are synchronized. More specifically, at the beginning of the experiment, the synchronization node (n_6) sends a signal to both sources. After receiving the signal, the sources set their clocks to zero and start to transmit packets periodically. A scenario with synchronized sources represents an undesirable operation environment. As we shall see, the delay characteristics under desirable and undesirable conditions differ significantly.

The testbed uses B-MAC [10], the default MAC protocol in TinyOS. The route from a source to the sink is fixed. In each setting, a source sends 8000 to $10,000$ packets to the sink (we conduct four to five experiments, each containing 2000 packets). The air monitor runs `apps/TOSBase.nc` that captures MAC-level packets; all the other nodes run `apps/SurgeTelos.nc` that sends packet periodically or forward incoming packets depending on whether the node is a source or a forwarder. Both programs are provided by TinyOS 1.x. We investigate two scenarios: (1) a node does not retransmit a packet; and (2) a node retransmits a packet for up to five times when not receiving an ACK from its parent.

B. Metrics

In each experiment, we obtain four metrics: (1) per-hop delay, (2) end-to-end delay, (3) percentage of packets captured by the air monitor, (4) end-to-end loss rate. The first two metrics are obtained using the methodology in Section II. The third metric quantifies the amount of data captured through air monitoring. More specifically, for node n (a source, a forwarder, or both), we obtain the total number of packets transmitted by the node, N_n, and the number of packets captured the air monitor, N_n^c. Then the percentage of packets from node n that are captured by the air monitor, referred to as *packet capture percentage*, is N_n^c/N_n. We obtain N_n^c by simply counting the number of packets captured by the air monitor. To obtain N_n, we take advantage of the DSN fields in the MAC headers. By the specification of IEEE 802.15.4, each node starts with a certain initial value of DSN (not necessarily 0) and increases the DSN field by one after transmitting a data packet. Therefore, if the DSN field is sufficiently long, all data packets from a node will have different DSNs and the DSNs of two adjacent data packets differ by one. In this case, we can obtain N_n as the difference of the DSNs of the last packet and the first packet sent by node n. By default, however, the DSN field is 1 byte [1]. Therefore, the DSN fields of the data packets from a node are in the cycles of $0, 1, \ldots, 255$ (the first cycle may not start with 0). We hence determine N_n based on the number of cycles. The last metric, end-to-end loss rate, is the number of packets not received by the sink over the total number of packets sent by the source. Although it is not our main focus, it is closely related to delay measurement and helps understanding the results in each setting.

C. Packet format

A source sends a packet every two seconds to the sink. Each packet has a payload of 28 bytes. It carries the source and destination addresses. Furthermore, it carries the address of the node that sends the packet (which may be the source or a forwarder along the path). To differentiate the packets at the application level, each packet carries a sequence number in the payload. When allowing retransmissions, our methodology to identify successful transmissions requires the air monitor to captures ACKs (see Section II-B.2). By default, the library, `lib/CC2420Radio/CC2420RadioM.cc`, called by `apps/TOSBase.nc` at the air monitor does not capture ACKs. We modified the library to enable it to capture ACKs. After the modification, however, we found that the air monitor captures either a data packet or its corresponding ACK; it never captures both. After a careful analysis of the program, we conjecture that this exclusive capture is caused by the hardware[1]. We therefore resort to the following method to determine the successful transmission. Suppose node n_{i-1} transmits a packet to its parent n_i, $i = 1, \ldots, k+1$. In the packet payload, n_{i-1} adds a field to index the transmissions. That is, n_{i-1} sets the field to j when transmitting the packet for the j-th time. After receiving a transmission successfully, node n_i extracts the field from the packet and embeds it in its forwarding packet. To sum up, each packet forwarded by a node contains two additional fields in the payload, one indexing the transmissions by the current node, the other being the index of the successful transmission from its child node. By analyzing the payloads, the air monitor determines the successful transmission to obtain per-hop delays (see Section II-B.2).

D. Packet forwarding

When allowing a packet to be retransmitted for up to five times, we still observe significant losses in the setting with synchronized sources. Analysis of the program `apps/SurgeTelos.nc` indicates that the forwarding library that it calls, `/lib/MulyihopLQI/MultihopEngineM.nc`, drops an incoming packet if it arrives when another packet is being processed. To solve this problem, we modified the forwarding library so that an incoming packet is put into a waiting queue if another packet is being processed at its arrival time. After the modification, we confirm that the loss rate is very low and the losses occur during transmissions in the air instead of being dropped at a forwarder.

IV. Experiment results

We now report the results when each node retransmits a packet for up to five times. The results when each node does not retransmit packets are consistent and can be found in the Appendix.

A. Single-source setting

We first report the results under the setting with a single source. A total of $10,000$ packets are transmitted from the source to the sink. The end-to-end loss rate is 0.14%. Table I lists the percentage of packets captured by the air monitor. We observe that the capture percentage is close to 100%.

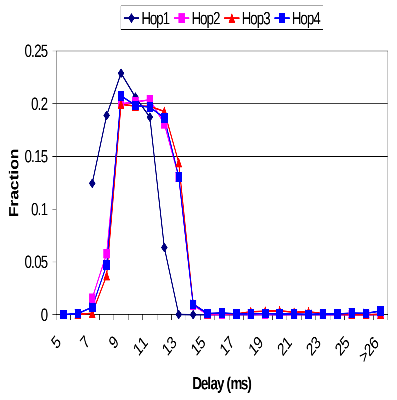
Fig. 5(a) plots the delay distributions on the four hops from the source to the sink. We observe that these distributions are similar. Most delays are in $[8, 14]$ ms. The variance in the delays are caused by the random delays to access the medium (B-MAC uses carrier sensing). On the first hop, the mean delay is 9.3 ms. On the later hops,

[1]In TelosB motes, ACKs are generated by hardware immediately after a mote receives a data packet. Therefore, an ACK arrives at the air monitor right behind its corresponding data packet. From our experiments, we conjecture that when both a data packet and its ACK arrive at the air monitor, the air monitor filters out the ACK; while if only the ACK arrives at the air monitor, the air monitor captures the ACK.

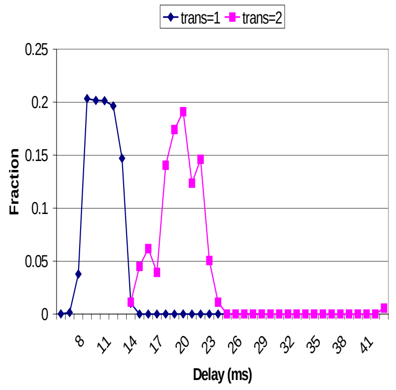
TABLE I

SINGLE SOURCE: PERCENTAGE OF PACKETS CAPTURED BY THE AIR MONITOR.

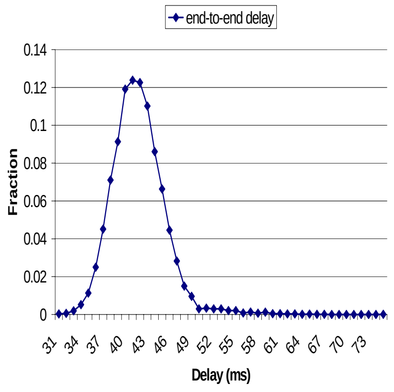
node ID	send num.	captured num.	percentage captured (%)
n_1	10700	10636	99.4
n_2	10683	10471	98.0
n_3	10941	10455	95.6
n_4	10702	10592	99.0
all	43026	42154	98.0



(a)



(b)



(c)

Fig. 5. Single source setting: (a) per-hop delay distribution, (b) the conditional delay distribution on the 3rd hop given that the number of transmissions is 1 or 2, (c) end-to-end delay distribution.

the mean delays are 10.6, 11.0 and 10.9 ms, respectively. The slightly larger average delay on a later hop than that on the first hop is due to the following reasons. At a later hop, after receiving a data packet from its child, a node first sends an ACK to the child and then forwards the packet to its next hop node. Due to medium contention, the data packet can only be forwarded after the ACK has been sent, which leads to an additional delay in forwarding the data packet. This additional delay is not present in the first hop since the source does not need to send any ACK.

We next investigate the conditional delay distribution on a hop given that a packet is transmitted for i times, $i \geq 1$. On the third hop, around 98% of the packets are transmitted once and 2% of the packets are transmitted twice. On the other hops, almost all of the packets are transmitted only once. Fig. 5(b) plots the conditional delay distribution on the 3rd hop given that a packet is transmitted once or twice. We observe that these two distributions are mostly disjoint: when packets are transmitted once, their delays are mostly in $[8, 14]$ ms; when packets are transmitted twice, their delays are mostly in $[14, 24]$ ms.

Last, Fig. 5(c) plots the end-to-end delay distribution. The majority of the delays are in $[33, 52]$ ms, approximately four times of the range on a single hop (i.e., [8,14] ms). The mean is 41.8 ms.

B. Synchronized parallel sources

We now report the results in the setting with synchronized parallel sources. This setting contains two sources, n_1 and n_5, and they compete with each other to send packets to node n_2. Sources n_1 and n_5 each transmits $10,000$ packets to the sink. Their end-to-end loss rates are 0.25% and 0.12% respectively. Table II lists the percentage of packets captured by the air monitor. We observe that the capture percentage is between 68.8% and 87.3%, lower

TABLE II

SYNCHRONIZED PARALLEL SOURCES: PERCENTAGE OF PACKETS CAPTURED BY THE AIR MONITOR.

node ID	send num.	captured num.	percentage captured (%)
n_1	13599	11006	80.9
n_5	14863	10226	68.8
n_2	22793	18650	81.8
n_3	22797	19899	87.3
n_4	22421	19262	85.9
all	96473	79043	81.9

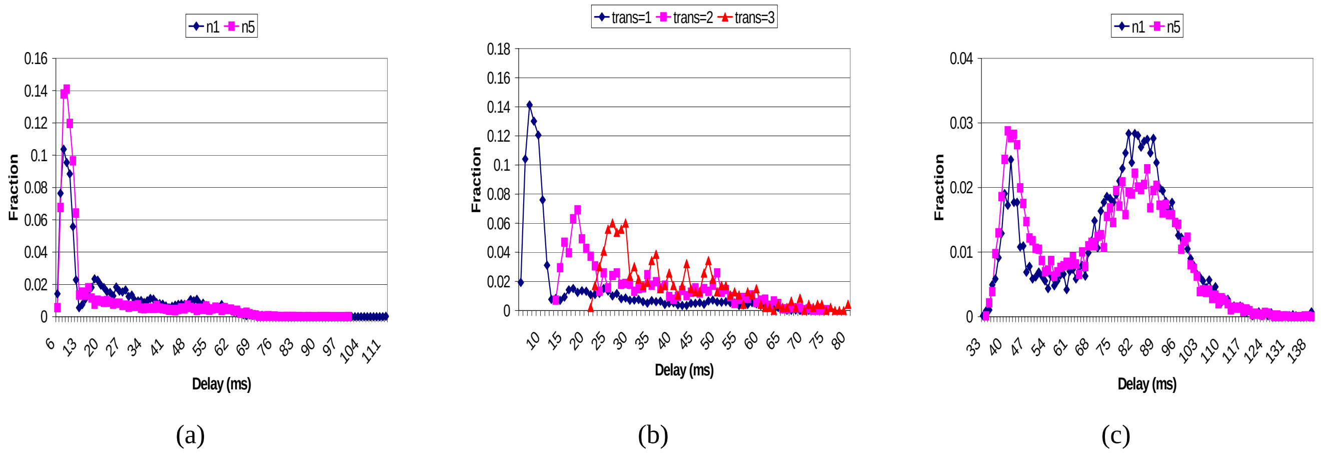


Fig. 6. Synchronized parallel sources: (a) first-hop delay distribution, (b) conditional delay distributions of source n_1 on the first hop, (c) end-to-end delay distribution.

than that when there is a single sender. The lower percentage is due to larger number of packets and larger amount of medium contention caused by synchronized sources.

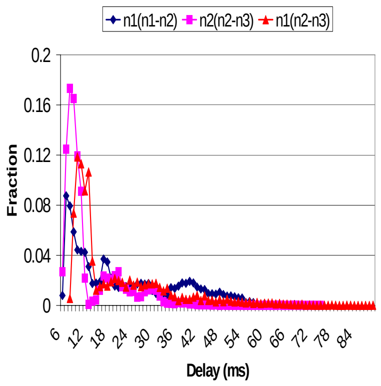
Fig. 6(a) plots the first-hop delay distributions of sources n_1 and n_5. We observe that these two distributions are of similar shape: most of delays are in $[6, 70]$ ms, in a much wide range than that in the single-sender setting. The wider range is caused by medium contention between these two sources. For source n_1, the mean delay is 22.5 ms; for source n_5, the mean delay is 18.7 ms. Looking into the trace at the air monitor, we find that, of all the packets, node n_5 sends 62% of the packets earlier than node n_1 (we refer to the first transmissions of the packets). This partly explains why node n_5 has shorter delays than node n_1. The reason why n_5 sends more packets earlier than n_1 might be due to settings of the experiments (e.g., n_5 receives the synchronization signal slightly earlier than n_1 and hence starts transmitting earlier). We also observe that node n_1 has more packet retransmissions than n_5: for source n_1, above 27% of the packets are transmitted more than once; while for source n_5, around 17% of the packets are transmitted more than once. The more retransmissions also contribute to larger delays of source n_1 than n_5. The above results indicate that, when two nodes compete for medium, one node can be penalized and suffer from higher delays.

Fig. 6(b) plots the conditional delay distribution of source n_1 on the first hop given that a packet is transmitted for i times, $i = 1, 2, 3$ (the results for source n_5 are similar). We observe that these distributions have significant overlaps. For instance, even when packets are only transmitted once, their delays are in a wide range of $[6, 70]$ ms, overlapping with those when packets are transmitted for two or three times. The longer delays are caused by

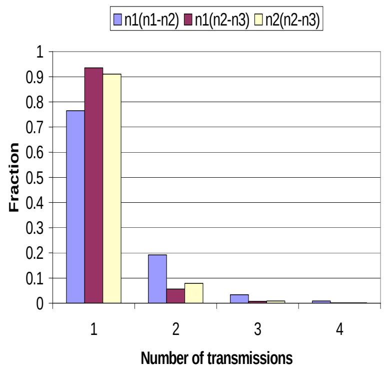
TABLE III

SYNCHRONIZED TANDEM SOURCES: PERCENTAGE OF PACKETS CAPTURED BY THE AIR MONITOR.

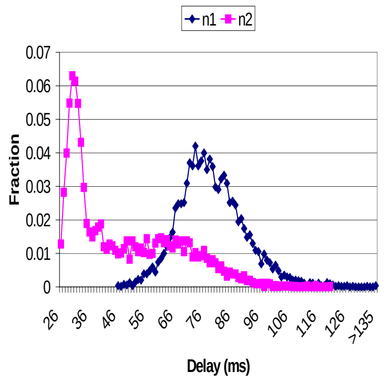
node ID	send num.	captured num.	percentage captured (%)
n_1	13821	9601	69.5
n_2	23963	20653	86.2
n_3	22914	20395	89.0
n_4	22625	20286	89.7
all	83323	70935	85.1



(a)



(b)



(c)

Fig. 7. Synchronized tandem sources: (a) delay distributions of source n_1 on its first two hops; delay distribution of node n_2 on its first hop; (b) histogram of the number of transmissions; (c) end-to-end delay distribution.

longer waiting times due to medium contention.

We now describe the delays of sources n_1 and n_5 on later hops. For source n_1, the delays on the hops of (n_2, n_3), (n_3, n_4) and (n_4, t) are 18.3, 17.4 and 16.5 ms respectively; for source n_5, the delays are 17.0, 17.3 and 17.0 ms respectively (figures omitted). The less disparity between n_1 and n_5 at later hops than that on the first hop is because on a later hop, packets from the two sources are transmitted by the same node (in sequence) instead of two synchronized competing nodes as on the first hop. On the other hand, these delays are significantly larger than a per-hop delay in the single-source setting (around 11 ms). This is due to packet queuing and medium contention.

Last, Fig. 6(c) plots the end-to-end delay distributions for sources n_1 and n_5. Most delays are in the range of $[33, 124]$ ms. This range is much wider than that without medium contention. The average delays of sources n_1 and n_5 are respectively 75.3 and 70.7 ms, 80% and 69% higher than that with a single sender. Compared to the end-to-end delay of node n_5, the slightly larger end-to-end delay of n_1 is mostly due to its larger delay on the first hop.

C. *Synchronized tandem sources*

We now report the results in the setting with synchronized tandem sources. This setting contains two sources, n_1 and n_2. Furthermore, n_2 is also a forwarder for n_1. Sources n_1 and n_2 each transmits 10,000 packets to the sink. The end-to-end loss rate of source n_1 is 0.29% and the end-to-end loss rate of source n_2 is 0.26%. Table III lists the percentage of packets captured by the air monitor. Again, the lower percentages than those in the single-source setting is due to larger number of packets and larger amount of medium contention caused by synchronized sources.

Fig. 7(a) plots the delay distributions on the hops of (n_1, n_2) and (n_2, n_3) for source n_1, and the delay distribution on the hop of (n_2, n_3) for source n_2. For the two sources, on their respective first hop, the delays of packets from n_1 (with the mean of 22.6 ms) are higher than those from source n_2 (with the mean of 12.9 ms). Looking into the trace, we find that, of all the packets, source n_1 transmits around 61% of the packets later than n_2 (we again refer to the first transmissions), which partly explained why the delays at n_1 are larger. Furthermore, as shown in Fig. 7, packets from n_1 are retransmitted for more times than those from n_2: for source n_1, over 19.2% and 3.4% of the packets are transmitted for two or three times; while for source n_2, only 7.9% and 0.9% of the packets are transmitted for two or three times. This again demonstrates that when two nodes compete to send, one can suffer from much higher delays. On the hop of (n_2, n_3), the delays of packets from source n_1 (with the mean of 20.0 ms) are much higher than those of source n_2 (with the mean of 12.9 ms). This is because packets from n_1 must be forwarded to n_2 first, while packets from n_2 are transmitted directly from n_2. Therefore, packets from n_1 need to wait in the queue at node n_2 to be transmitted, which causes larger delays than those from n_2. The above is confirmed by the traces: at n_2, all packets from source n_1 are transmitted later than their corresponding packets from n_2 (we again refer to the first transmission of a packet).

On later hops (i.e., hops (n_3, n_4) and (n_4, t)), the delays of packets from sources n_1 and n_2 are similar. On the hop of (n_3, n_4), the average delays of packets from source n_1 and n_2 are 17.4 and 17.3 ms, respectively. On the hop of (n_3, t), the average delays of packets from source n_1 and n_2 are 15.5 and 17.9 ms, respectively. On these hops, all packets from n_1 are sent later than their corresponding packets (packets with the same sequence numbers) from n_2. However, due to the delays on (n_1, n_2) and (n_2, n_3), packets from source n_1 are far behind (around 30 ms) those from n_2. Therefore, the packets from n_1 and n_2 do not affect each other on the same hop.

On all the hops, for both sources, the per-hop conditional delay distributions given the number of transmissions are similar to that in Fig. 6(b). Last, Fig. 7(c) plots the end-to-end delay distribution. For both sources n_1 and n_2, we observe larger variances than that without medium contention (in Fig. 5(c)). The average delay of source n_1 is 78.7 ms, 88% higher than that without contention. Source n_2 has three hops to the sink. Its average end-to-end delay is 47.6 ms, beyond the range of three-hop delays without contention (the range for per-hop delay without contention is $[8, 14]$ ms).

V. Conclusions and Future Work

In this paper, we proposed a method that uses passive air monitoring to measure per-hop and end-to-end delays in a sensor network. We also used this method to characterize delays in a sensor-network testbed under various settings. We find that, in a setting with light load and no medium contention, per-hop delays are mostly in $[6, 14]$ ms with the mean of 9 to 11 ms; when two nodes compete for medium, one node can suffer from significantly larger delays, and the delays are in a much wider range ($[6, 70]$ ms) with the mean as high as 23 ms. As future work, we plan to characterize delays in a larger-scale testbed with multiple air monitors.

References

[1] IEEE Standard 802.15.4. http://www.michaelwhyte.net/zigbee/802154.pdf.

[2] A. Adya, V. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks. In *Proc. of ACM MobiCom*, September 2004.

[3] P. Bahl, R. Chandra, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill. Enhancing the security of corporate Wi-Fi networks using DAIR. In *Proc. of ACM MobiSys*, 2006.

TABLE IV

SINGLE SOURCE, NO RETRANSMISSION: PERCENTAGE OF PACKETS CAPTURED BY THE AIR MONITOR.

node ID	send num.	captured num.	percentage captured (%)
n_1	10763	10731	99.7
n_2	10753	9920	92.3
n_3	10627	10547	99.3
n_4	10500	10283	97.9
all	42643	41481	97.3

[4] Y.-C. Cheng, M. Afanasyev, P. Verkaik, P. Benko, J. Chiang, A. C. Snoeren, S. Savage, and G. M. Voelker. Automating cross-layer diagnosis of enterprise wireless networks. In *Proc. of ACM SIGCOMM*, Kyoto, Japan, August 2007.

[5] Y.-C. Cheng, J. Bellardo, P. Benko, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *Proc. of ACM SIGCOMM*, Pisa, Italy, September 2006.

[6] F. Dressler, R. Nebel, and A. Awad. Distributed passive monitoring in sensor networks. In *Proc. of IEEE INFOCOM*, 2007. poster.

[7] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: an experimental study of low-power wireless sensor networks. Technical Report UCLA/CSD-TR 02-0013, February 2002.

[8] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding congestion in IEEE 802.11b wireless networks. In *Proc. of ACM SIGCOMM Internet Measurement Conference (IMC)*, 2005.

[9] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the MAC-level behavior of wireless networks in the wild. In *Proc. of ACM SIGCOMM*, Pisa, Italy, September 2006.

[10] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[11] N. Reijers, G. Halkes, and K. Langendoen. Link layer measurements in sensor networks. In *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Fort Lauderdale, FL, October 2004.

[12] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. C. Sicker. MOJO: A distributed physical layer anomaly detection system for 802.11 WLANs. In *Proc. of ACM MobiSys*, pages 191–204, 2006.

[13] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4), 2004.

[14] J. Yeo, M. Youssef, and A. Agrawala. A framework for wireless LAN monitoring and its applications. In *Proc. of ACM Workshop on Wireless Security (WiSe)*, 2004.

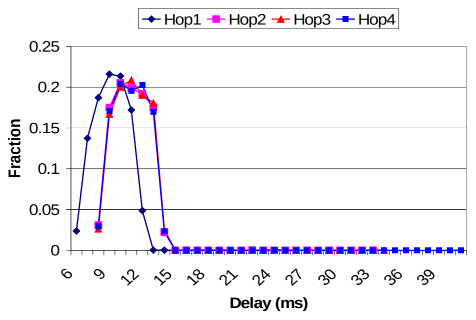
[15] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

APPENDIX
RESULTS WITHOUT PACKET RETRANSMISSION

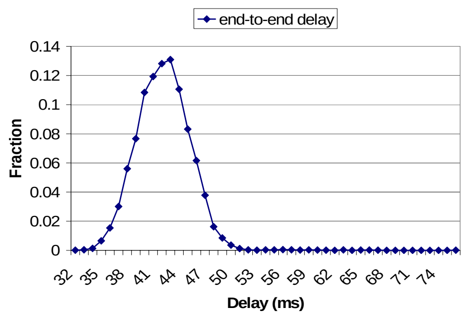
We now report the measurement results when a node does not retransmit packets. The results are consistent with those when a node retransmits a packet for up to five times.

A. Single-source setting

We first report the results under the setting with a single source. A total of $10,000$ packets are transmitted from the source to the sink. The end-to-end loss rate is 2.6%. Table IV lists the percentage of packets captured by the air monitor. We observe that the capture percentage varies from 92.3% to close to 100%. Fig. 8(a) plots the delay distributions on the four hops from the source to the sink. On the first hop, the mean delay is 9.2 ms. On the later hops, the mean delays are 11.0, 11.0 and 11.0 ms, respectively. Fig. 8(b) plots the end-to-end delay distribution. The mean is 42.2 ms.



(a)



(b)

Fig. 8. Single source setting, no retransmission: (a) per-hop delay distribution, (b) end-to-end delay distribution.

TABLE V

SYNCHRONIZED PARALLEL SOURCES, NO RETRANSMISSION: PERCENTAGE OF PACKETS CAPTURED BY THE AIR MONITOR.

node ID	send num.	captured num.	percentage captured (%)
n_1	8654	6123	70.8
n_5	8662	7769	89.7
n_2	14101	12508	88.7
n_3	13312	11697	87.9
n_4	12581	11580	92.0
all	47310	49677	86.7

B. *Synchronized parallel sources*

We now report the results in the setting with synchronized parallel sources, where two sources, n_1 and n_5, compete with each other to send packets to node n_2. Sources n_1 and n_5 each transmits $8,000$ packets to the sink. Their end-to-end loss rates are 37.3% and 19.7% respectively. Table V lists the percentage of packets captured by the air monitor.

Fig. 9(a) plots the first-hop delay distributions of sources n_1 and n_5. For source n_1, the mean delay is 20.9 ms; for source n_5, the mean delay is 10.0 ms. Looking into the trace at the air monitor, we find that for around 92% of the packets, node n_5 sends the packets earlier than node n_1. This explains why node n_5 has shorter delays than node n_1. We also observe that the delays of n_1 are in a wide range, indicating the effect of medium contention.

We now describe the delays of sources n_1 and n_5 on later hops. Fig. 9(b) plots the second-hop delay distributions for sources n_1 and n_5. These two distributions are similar. The average delays for sources n_1 and n_5 are respectively 19.1 and 15.3 ms. For source n_1, The average delays on the third and fourth hops are 16.9 and 15.6; for source n_5, these delays are 15.2 and 15.0.

Last, Fig. 9 (c) plots the end-to-end delay for sources n_1 and n_5. The average delays of sources n_1 and n_5 are respectively 74.2 and 55.4 ms, respectively 76% and 31% higher than that with a single sender.

C. *Synchronized tandem sources*

We now report the results in the setting with synchronized tandem sources. This setting contains two sources, n_1 and n_2. Furthermore, n_2 is also a forwarder for n_1. Sources n_1 and n_2 each transmits $10,000$ packets to the sink. The end-to-end loss rate of source n_1 is 34.6% and the end-to-end loss rate of source n_2 is 18.4%. Table VI lists the percentage of packets captured by the air monitor.

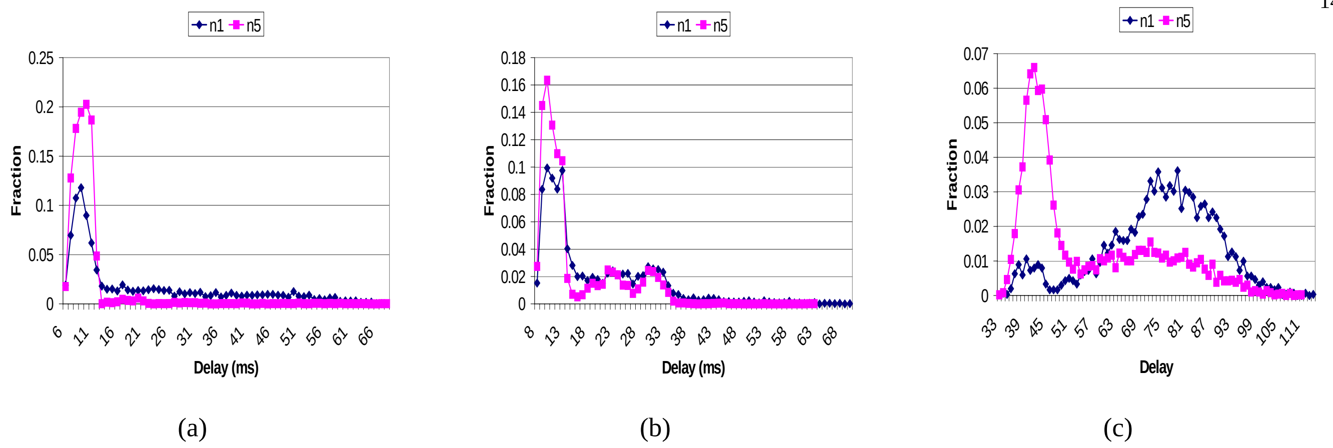


Fig. 9. Synchronized parallel sources, no retransmission: (a) first-hop delay distribution, (b) 2nd-hop delay distribution, (c) end-to-end delay distribution.

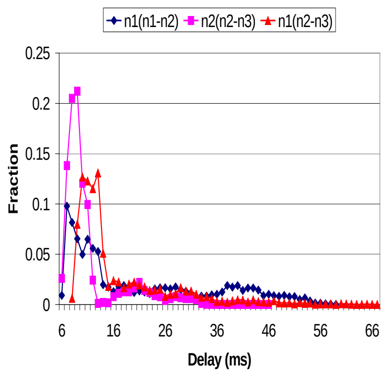
TABLE VI
SYNCHRONIZED TANDEM SOURCES, NO RETRANSMISSION: PERCENTAGE OF PACKETS CAPTURED BY THE AIR MONITOR.

node ID	send num.	captured num.	percentage captured (%)
n_1	10770	7236	67.2
n_2	18308	15621	85.3
n_3	16756	15238	90.9
n_4	15859	14448	91.1
all	61693	52543	85.2

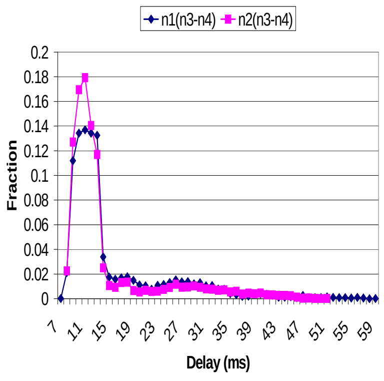
Fig. 10(a) plots the delay distributions on the hops of (n_1, n_2) and (n_2, n_3) for source n_1, and the delay distribution on the hop of (n_2, n_3) for source n_2. For source n_1, the average delay on the hop of (n_1, n_2) is 21.2 ms; the average delay on the hop of (n_2, n_3) is 17.0 ms. For source n_2, the average delay on the hop of (n_2, n_3) is 11.2 ms. For the two sources, on their respective first hop, the delay of packets from n_1 are higher than those from source n_2. Looking into the trace, we find that, of all the packets, source n_1 transmits 65% of the packets later than n_2. Again, at node n_2, all packets from source n_1 are transmitted later than their corresponding packets from n_2.

On later hops (i.e., hops (n_3, n_4) and (n_4, t)), the delays of packets from sources n_1 and n_2 are similar. Fig. 10 (b) plots the delay distributions on hop (n_3, n_4), where the average delays of packets from source n_1 and n_2 are 15.6 and 14.5 ms, respectively. On the hop of (n_3, t), the average delays of packets from source n_1 and n_2 are 14.7 and 15.3 ms, respectively.

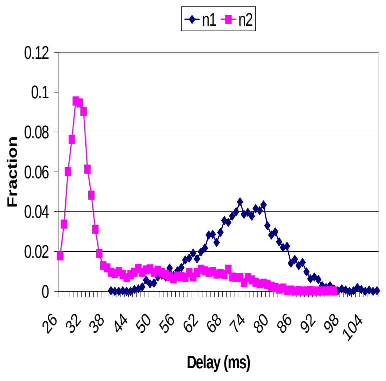
Fig. 10(c) plots the end-to-end delay distribution. The average delay of source n_1 is 72.2 ms, 71% higher than that without contention. The average end-to-end delay of source n_2 is 40.4 ms — it is close to the four-hop end-to-end delay in the single-sender setting although n_2 is only three hops away from the sink.



(a)



(b)



(c)

Fig. 10. Synchronized tandem sources, no retransmission: (a) delay distributions of source n_1 on its first two hops; delay distribution of node n_2 on its first hop. (b) delay distributions on the hop of (n_3, n_4). (c) end-to-end delay distributions.