Domo: Passive Per-Packet Delay Tomography in Wireless Ad-hoc Networks

Yi Gao[1], Wei Dong[1*], Chun Chen[1], Jiajun Bu[1], Tianyu Chen[2], Mingyuan Xia[3], Xue Liu[3], Xianghua Xu[2]

[1]*College of Computer Science, Zhejiang University, China*

[2]*School of Computer Science, Hangzhou Dianzi University, China*

[3]*School of Computer Science at McGill University, Canada*

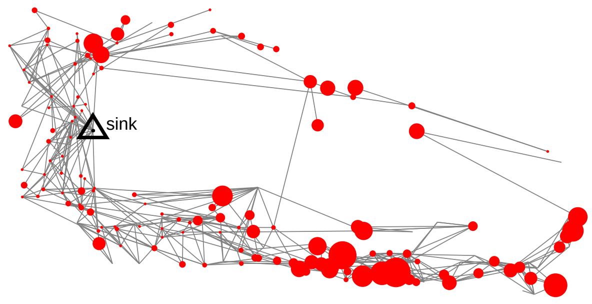
***Abstract*—In multi-hop wireless ad-hoc networks, packet delivery delay is one of the most important performance metrics. While a lot of research efforts have been spent on measuring and optimizing the end-to-end delay performance, there usually lack accurate and lightweight methods for decomposing the end-to-end delay into the per-hop delay for each packet. Knowledge on the per-hop per-packet delay can greatly improve the network visibility and facilitate network measurement and management. In this paper, we propose Domo, a passive, lightweight and accurate delay tomography approach to decomposing the packet end-to-end delay into each hop. The basic idea is to formulate the problem into a set of optimization problems by carefully considering the constraints among various timing quantities. At the network side, Domo attaches a small overhead to each packet for constructing constraints for the optimization problems. At the PC side, Domo employs semidefinite relaxation and several other methods to efficiently solve the optimization problems. We implement Domo and evaluate its performance extensively using large-scale simulations. Results show that Domo significantly outperforms two existing methods, nearly tripling the accuracy of the state-of-the-art.**

I. Introduction

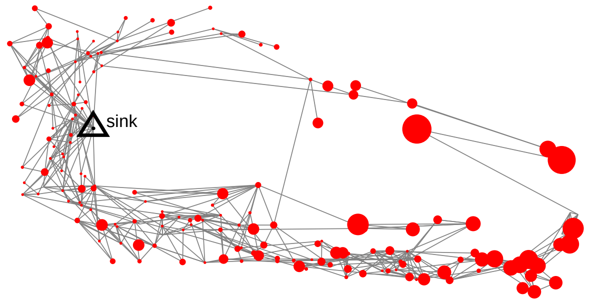
Wireless ad-hoc networks have been successfully applied in many application scenarios such as ecosystem management [1], structural protection [2] and urban CO_2 monitoring [3]. In these applications, data packets are usually delivered to a central sink through a multi-hop wireless network. Packet delivery delay is one of the most important performance metrics for many time-sensitive applications [4], [5], [2]. While a lot of research efforts have been spent on measuring and optimizing the end-to-end (i.e., node to sink) delay (or path delay) [6], [7], [8], there usually lack accurate and lightweight methods for decomposing the path delay into the per-hop delay (or node delay). This decomposition is called delay tomography in the literature [9].

Knowledge on the per-hop delay provides paramount benefits for network measurement and management. Without the fine-grained per-hop delay information, packet delivery behaviors inside the network are invisible to network managers. This lack of network visibility poses great difficulties for the network designers to further optimize various network performance [10]. Figure 1(a) shows the end-to-end delay distribution of a deployed network [3] at time t_1. A larger dot represents a longer end-to-end delay. We can see that end-to-end delays from different nodes can be very different. It is, however, difficult to pinpoint the problematic nodes with only the end-to-end delay information. In contrast, per-hop delay information enables efficient detection of the problematic nodes, as well as other detailed network analysis.

Traditional probing-based approaches in the Internet [11], [12], [13] inject probing packets to help recover the delays. These approaches are not applicable for wireless ad-hoc networks for two reasons. First, extra probing packets introduce high message overhead for resource constrained



(a) end-to-end delay distribution at time t_1



(b) end-to-end delay distribution at time t_2

Figure 1. End-to-end delay distributions of a deployed network [3] at two different times. The positions of nodes in the figures are based on their physical locations. The sink node locates in the triangle.

*The Corresponding Author is Wei Dong (dongw@zju.edu.cn). [1]Y. Gao, W. Dong, C. Chen and J. Bu are also with the Zhejiang Provincial Key Laboratory of Service Robot. [2]T. Chen and X. Xu are also with the Zhejiang Provincial Key Lab of Data Storage and Transmission Technology.

networks like sensor networks. Therefore, it is desirable to passively reconstruct the fine-grained delay information without additional traffic. Second, these approaches provide only statistical [9] information about per-hop delay, which fails to capture the link dynamics in wireless ad-hoc networks. For example, Figure 1(b) shows the delay distribution of the same network at a different time t_2. Compared with the delays at time t_1, delays of more than 50% of the nodes change more than 58% at time t_2. The time-varying nature of wireless networks motivates us to design a per-hop per-packet delay tomography approach, i.e., reconstructing the per-hop delay of each packet.

Basically, there are two requirements on the problem of per-hop per-packet delay tomography. First, the method should be lightweight such that the Heisenbugs will not be introduced into the network. This precludes native approaches which attach the per-hop delays into the packet itself since the overhead grows linearly as the path length increases. Second, the method should be accurate. Since the sojourn time of a packet stay on a node could be as short as several milliseconds, it is desirable to achieve millisecond accuracy.

Compared with end-to-end delays, which are relatively easy to be recorded in packets [7], it is challenging to achieve lightweight accurate delay tomography. On important reason is that there is an inherent tradeoff between accuracy and overhead. Our main contribution in this paper is to strike a good balance between the two goals. The basic idea is to formulate the problem into a set of optimization problems by carefully considering the constraints among various timing quantities as well as the inherent network operations.

We propose Domo, a **D**elay t**omo**graphy approach for achieving lightweight and accurate per-hop per-packet delay reconstruction in wireless ad-hoc networks. Domo consists of two parts. At the node side, Domo attaches a small overhead to each packet for constructing constraints for the optimization problems. At the PC side, Domo employs multiple techniques to efficiently solve the optimization problems. First, semi-definite relaxation is used to facilitate problem solving. After relaxation, the optimization is turned into a convex problem and can thus be solved efficiently. Second, a time window based method is used to further reduce the computation time while preserving the accuracy. Third, a sub-graph extraction method is used to efficiently calculate the upper bound and lower bound of the per-hop per-packet delay.

We implement Domo on TelosB/TinyOS and evaluate its performance extensively using large-scale simulations. Results show that Domo significantly outperforms two existing methods, i.e., MNT [14] and MessageTracing [15], in terms of accuracy. Specifically, Domo achieves an average accuracy of 3.58ms, which nearly triples the accuracy of the state-of-the-art.

The contributions of this paper are summarized as follows.

- We propose Domo, an accurate and lightweight approach for delay tomography in wireless ad-hoc networks.
- We formulate the per-hop per-packet delay tomography problem into optimization problems and propose multiple methods for efficient problem solving.
- We extensively evaluate the performance of Domo by both trace-driven studies and large scale simulations. Results show that Domo significantly outperforms existing methods.

The rest of the paper is organized as follows: Section II discusses the related work. Section III introduces the assumptions made and notations used in this work. Section IV presents the design of Domo. Section V describes the implementation details and Section VI presents the evaluation results and Section VII concludes this work.

II. Related Work

A. Delay Measurement in Wired IP Network

Delay reconstruction is widely studied in wired IP networks for various network analysis [11], [12], [9]. LDA [11] is a low-overhead mechanism for end-to-end latency measurement. RLI [12] obtains per-flow delay by performing linear interpolation based on the data collected from the reference packets being injected to the sender. Besides the end-to-end delay measurement approaches, there are also approaches aiming at reconstructing per-hop delay. A recent work [9] proposes a very effective algorithm to reconstruct additive link metrics (e.g., delay) by path measurements. It assumes the link metrics are constant during the measurement, which is not the case in wireless ad-hoc network. Another work in [16] measures and analyzes the single-hop packet delay on an IP backbone network. It requires global clock synchronization for all routers in the networks. These work cannot be migrated to wireless due to the link dynamics. Domo overcomes this difficulty by performing per-hop per-packet delay reconstruction. At the meantime, Domo does not depend on synchronized global clock, which is usually not available in wireless ad-hoc networks [7] due to its high message and energy overhead.

B. Delay Measurement in Wireless Ad-hoc Network

End-to-End delay. In wireless ad-hoc networks, there are several recent work [7], [17] focusing on recording the end-to-end delay of each collected packet at the sink. J. Wang et al. [7] propose to timestamp packets at the MAC layer to precisely record packet sending/receiving. Then the packet sojourn time at each hop can be accumulated and recorded in a particular field in the packet. When the packet reaches the sink, that field will store the whole end-to-end delay. Different with these approaches, Domo aims at reconstructing more fine-grained delay, i,e., per-hop per-packet delay.

Per-hop delay. There are also several approaches [14], [15] related to fine-grained delay reconstruction in wireless ad-hoc networks. MNT [14] is the state-of-the-art approach to reconstruct per-hop packet arrival order in sensor networks. For each packet p, MNT is able to infer the two local packets right before and after packet p at each hop. Since the generation time of each local packet is known, the per-hop delay of the packet p can be bounded by the two corresponding local packets at each hop. Further, the bounds can be improved by correlating information from packets passing through the same forwarding nodes as packet p. MessageTracing [15] records every packet sent and received into the local storage of each node. It does not reconstruct per-hop per-packet delay directly. However, by the information it records, we can infer the order of many packet sending/receiving events. From the perspective of revealing the internal behaviors of the networks, MessageTracing provides another possible method (i.e., local logging). In the evaluation section, we will show that Domo significantly outperforms these related approaches in terms of the reconstruction accuracy.

III. Network Model

In this section, we give the network model of this work. We also summarize the assumptions made and notations used in Domo. We assume that Domo works in a wireless ad-hoc network running data collection task to a single sink node. Each node generates and sends data packets periodically to the sink. All nodes in the network can forward packets for other nodes. The network includes common phenomena in wireless ad-hoc networks such as routing dynamics, packet loss and no global timing information.

A. Modeling

FIFO send queue. Each node includes an FIFO send queue for all outgoing packets. This queue keeps the packets generated by the node as well as packets to be forwarded. The node will keep sending the same packet at the queue head till a successful acknowledgement or the maximum number of retransmission is reached. FIFO send queue is widely used in routing protocols (e.g., CTP [18], MiniRoute) for wireless ad-hoc networks.

Routing path information. Since the wireless network topology is usually dynamic. Per-hop delays are only useful when the packet path is already known. Thanks to the recent progress in path reconstruction [14], [19], we are able to reconstruct the packet path with small overhead. In this paper, we assume the availability of per-packet routing path.

Node delay. Since wireless signal propagates very fast in the air, a packet delay from the source to the sink actually consists of the sojourn times the packet stays on the source node and all forwarding nodes. The sojourn time is the time between the node's radio starts to receive the packet and starts to transmit the packet. The sojourn time at each node can be measured accurately on that node. In the implementation section, we will give a detailed description of measuring the node delay at that node.

End to end delay. Recent work [7], [17] show that the end to end delay can be obtained accurately after a packet reaches the sink. MAC layer timestamping technique is used to record the actual time a packet is transmitted by the radio at the physical layer. Based on this technique, the end to end delay can be recorded in the packet by accumulating all the node delays [7]. Note that this accumulating process is done on forwarding nodes of the packet. When the packet reaches the sink, the end-to-end delay is recorded in the packet while the node delays are not recorded.

B. Notations

For a packet p, we define the following notations. Figure 2 visualizes the relationship of these notations..

- $path(p)$: the path of packet p, which is represented by a sequence of node ids along the routing path of packet p, from the source node to the sink;
- $|p|$: the path length of p, which is short for $|path(p)|$;
- $N_i(p)$: the i^{th} node in $path(p)$, $i \in [0, |p|-1]$, e.g., $N_0(p)$ is the source node of packet p and $N_{|p|-1}(p)$ is the sink node;
- $t_i(p)$: global arrival time on the i^{th} node in $path(p)$ of packet p, $i \in [0, |p|-1]$, e.g., $t_0(p)$ is the generation time of packet p at node $N_0(p)$;
- $D_n(p)$: node delay of packet p at node n, which is the sojourn time that packet p stayed on node n. It can be calculated by subtracting two adjacent arrival times, $D_n(p) = t_{i+1}(p) - t_i(p)$ where n is the i^{th} node in $path(p)$. Similarly, the arrival time on the i^{th} node $t_i(p)$ can be calculated by adding the previous node delays to the generation time $t_0(p)$, that is $t_i(p) = t_0(p) + \sum_{j=0}^{i-1} D_{N_j(p)}(p)$.
- $S(p)$: sum of the node delays of packets sent/forwarded by $N_0(p)$ between packet p and its previous packet q with the same source node. As described in the assumptions, the node delay of each forwarded packet is measurable at each forwarder. Therefore, each forwarder is able to record the sum of node delays into its local packets.

Before we start to reconstruct per-hop per-packet delays, we have the following information for each packet p.

- Arrival time at the sink $t_{|p|-1}(p)$;
- Packet generation time $t_0(p)$, which can be obtained by existing time reconstruction methods [7];
- The routing path $path(p)$ of packet p, which can be reconstructed by multiple path reconstruction approaches [14], [20], [19];
- Sum of node delays $S(p)$, which is calculated on packet p's source node $N_0(p)$ and attached in packet p.

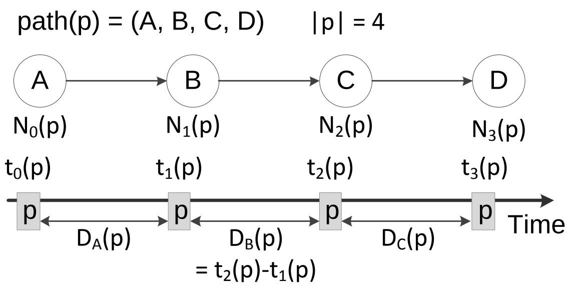


Figure 2. An example to illustrate the notations used in Domo and visualizes the relationship of them.

IV. Delay Reconstruction

In wireless ad-hoc networks, the sink node keeps receiving packets from other nodes. Each packet contains the routing information as well as the end-to-end delay between the source and sink. Then the sink would like to obtain the per-hop delay for every packet. In this section, we formulate this per-hop per-packet delay reconstruction problem into a set of optimization problems. We develop three convex constraints from network and packet transmission properties. With sufficient packets, the sink node is able to shrink the bounds of all per-hop delays for every packet by solving these optimization problems.

Since per-hop delays and the arrival times can be transformed to each other easily (as mentioned in Section III.B), we will use the one which makes the presentation more clear. For each packet p, the packet generation time $t_0(p)$ and the arrival time at the sink $t_{|p|-1}(p)$ are already known. The rest of its arrival times at other hops are the unknowns which need to be reconstructed.

When the unknown arrival times of each packet are considered separately, it is difficult to reconstruct them at the sink side. However, we observe that the arrival times of multiple packets are not independent, but highly correlated. By exploiting the correlation among per-hop arrival times of multiple packets, we can accurately reconstruct the arrival times.

Concretely, we construct three kinds of constraints of the unknown arrival times to represent the correlation among them. Depend on whether we need to calculate the estimated values or the bounds of the arrival times, different optimization problems can be constructed and solved.

A. Constraints Construction

FIFO constraint. The first kind of constraints is the FIFO constraint. Since there is a FIFO send queue on each node, packets always leave a node in the same order as they arrive. For any two packets x, y with a common node n (not the sink) in their paths, $n \in path(x)$ and $n \in path(y)$. Assume $n = N_{i_x}(x) = N_{i_y}(y)$, we have the following constraint.

$$(t_{i_x}(x) - t_{i_y}(y))(t_{i_x+1}(x) - t_{i_y+1}(y)) > 0. \tag{1}$$

The above constraint means if packet x arrives at node n earlier than packet y (i.e., $t_{i_x}(x) < t_{i_y}(y)$), packet x will also leave node n and arrive its next hop earlier than packet y (i.e., $t_{i_x+1}(x) < t_{i_y+1}(y)$).

This inequality constraint is not convex. To make this constraint practical for most solvers, we perform semidefinite relaxation [21] to obtain its convex relaxation. More specially for this inequality constraint, we can rewrite it as $(u_1 - u_2)(u_3 - u_4) > 0$. Its matrix form is shown as followings.

$$(u_1 \quad u_2 \quad u_3 \quad u_4)^T \begin{pmatrix} 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \end{pmatrix} (u_1 \quad u_2 \quad u_3 \quad u_4) > 0. \tag{2}$$

Let u be $(u_1\ u_2\ u_3\ u_4)$ and P be the symmetric matrix, we have $u^T P u > 0$. Since $u^T P u = \mathbf{Tr}(P(uu^T))$, we have $\mathbf{Tr}(P(uu^T)) > 0$. Let U be uu^T, the original constraint becomes the following two constraints.

$$\begin{aligned} \mathbf{Tr}(PU) &> 0 \\ U &= uu^T \end{aligned} \tag{3}$$

Note that the first one is a linear constraint and the second one is a non-convex equality constraint. The next step is the semidefinite relaxation. Change the second non-convex constraint into $U \preceq uu^T$ which is equivalent to $U - uu^T \preceq 0$. The following is its matrix form.

$$\begin{pmatrix} U & u \\ u^T & 1 \end{pmatrix} \preceq 0. \tag{4}$$

This changed constraint makes the original problem be an SDP [21] which has efficient algorithms to solve.

Order constraint. The second kind of constraints is the order constraint. The arrival times on all hops of a packet should be in order. That is, $t_0(p) < t_1(p) < ... < t_{|p|-1}(p)$. In practice, the per-hop delay has a minimum value due to the software processing delay. We use ω to denote this minimum software processing delay, and the second kind of constraint becomes the following.

$$t_0(p) < t_1(p) - \omega < \cdots < t_{|p|-1}(p) - (|p|-1)\omega. \tag{5}$$

Sum-of-delays constraint. The last kind of constraint is related to the sum of node delays $S(p)$. $S(p)$ can be calculated on node $N_0(p)$ accurately, but cannot be represented accurately by the arrival times t at the sink side. The reason is that the sink does not know which packets were forwarded by packet p's source node $N_0(p)$ between packet p and its previous packet q from the same source node. In order to relate the $S(p)$ in each packet p to the arrival times, we first define a candidate set of packets $C(p)$ as follows.

For a particular packet p, we define its candidate set $C(p)$ as a set of packets that satisfy the following three conditions.

1) For any packet $x \in C(p)$, $N_0(p)$ is in packet x's path; 2) For any packet $x \in C(p)$, its packet generation time $t_0(x)$ is earlier than packet p's generation time $t_0(p)$; 3) For any packet $x \in C(p)$, its arrival time at sink $t_{|x|-1}(x)$ is later than packet q's generation time $t_0(q)$, where packet q is the previous local packet from the same source node as packet p.

Packets in $C(p)$ are the *possible* packets whose delays may be included in $S(p)$. If there is no packet loss, the following constraint holds.

$$S(p) \leq D_{N_0(p)}(p) + \sum_{x \in C(p)} D_{N_0(p)}(x). \tag{6}$$

In deployed wireless ad-hoc networks, however, there are usually packet losses. Therefore, the candidate set of received packets at the sink side may be incomplete. As a result, the above constraint may not hold in case of packet losses. On the other hand, node delays of some packets in the candidate set are guaranteed to be included in $S(p)$. Let subset $C^*(p)$ be a set of these packets. For any packet $x \in C^*(p)$, if it is generated by the source and received by the sink between the generation times of packet p and its previous local packet q (i.e., $t_0(q) < t_0(x) < t_{|x|-1}(x) < t_0(p)$), its node delay at node $N_0(p)$ is guaranteed to be included in $S(p)$. Therefore, we can relate each $S(p)$ to the node delays of packets in set $C^*(p)$ as follows.

$$S(p) \geq D_{N_0(p)}(p) + \sum_{x \in C^*(p)} D_{N_0(p)}(x). \tag{7}$$

The above constraint is guaranteed to be true even under severe packet loss. We refer to this kind of constraints as sum-of-delays constraints.

B. Estimated Values of Arrival Times

There may be multiple solutions which satisfy all these constraints. We need an optimization goal to obtain a single solution as the estimated values of all unknown arrival times. We observe that within a relatively short period, the node delays of multiple packets on the same hop are similar. Therefore, we use the following optimization goal to obtain one solution of the arrival times.

$$\text{Minimize} \quad \sum_{n \in \mathscr{N}} \sum_{\substack{n \in path(x) \\ n \in path(y) \\ |t_0(x)-t_0(y)|<\varepsilon}} [D_n(x) - D_n(y)]^2 \tag{8}$$

where $\mathscr{N}$ represents all nodes in the network except the sink node; x, y present two packets with the same forwarder n; $t_0(x)$ presents the generation time of packet x and $D_n(x)$ represents the per-hop node delay of packet x at node n. This optimization goal means that we want to minimize the sum of variances of the per-hop node delays within a short period ε at all nodes. The sum of squares term represents the calculation of the per-hop node delay variance without the constant coefficient.

Note that this optimization goal is a quadratic function of the unknown arrival times. Since the summation of variances is always greater than or equal to zero, the coefficient matrix (i.e., matrix P if we write a quadratic function $f(x)$ in the form of $x^T P x + q^t x + c$) is positive semidefinite. In this case, the optimization problem is a convex optimization problem and can be solved efficiently [21].

Given a large number of received packets at the sink, however, it will still be time consuming to solve their per-hop delays. We divide the original trace into multiple time windows to improve the efficiency. One problem left is that the packets near the time window boundaries do not have enough number of constraints to limit their possible values. This will lower the accuracy of the arrival times of these packets. Therefore, we use the following improved time window approach instead.

For packets in a time window, we obtain their estimated per-hop arrival times by solving the optimization problem constructed by the packets in that time window. Due to the mentioned accuracy concern, we drop the estimated values near the time window boundaries and keep the ones far from the boundaries. Then we consider the next time window which has an overlapped area with the previous time window. We apply the same strategy on the second time window and drop some estimated values near the boundaries. The time windows are chosen so that after dropping a number of estimated values, the remaining values can also cover all unknown arrival times. Figure 3 illustrates this improved time window approach. There is a key parameter in this improved time window approach, the ratio of the number of remaining values and the number of all values in one time window. We refer this ratio as *effective time window ratio* in the following. A larger effective time window ratio means more values are kept in each time window and the accuracy is lower. A small effective time window ratio will increase the number of time windows and prolong the computation time. In practice, we set this ratio to be 0.5 which achieves both satisfactory accuracy and efficiency. In the evaluation section, we tune this ratio to validate its impact on the accuracy of Domo.

C. Bounds of Arrival Times

In some scenarios, the upper bounds and lower bounds of the arrival times are more useful than the estimated values. In this case, we use a different method to calculate the bounds of all unknown arrival times. The basic idea is just to find the upper bound and lower bound for each arrival time which satisfy the three kinds of constraints. For each unknown arrival time t, we construct two optimization problems to obtain its lower bound and upper bound, *Minimize t* and *Maximize t*, with considering the three kinds of constraints.

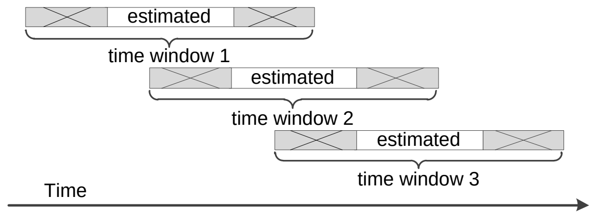


Figure 3. The improved time window approach to prevent the inaccuracy caused by the information loss near the boundaries. The arrival times in the grey areas are dropped. The remaining arrival times in the white areas of multiple time windows cover all the unknown arrival times.

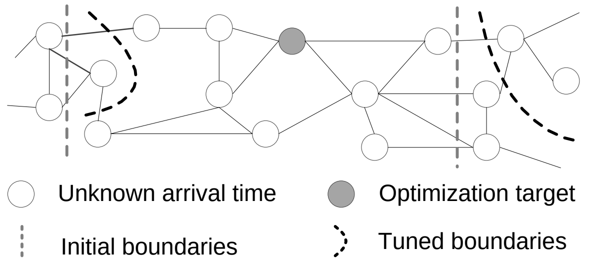


Figure 4. An example to illustrate how to extract a sub-graph without cutting too many edges. The initial solution cut 7 edges and the tuned solution only cut 4 edges.

The problem is that we need to solve twice number of optimization problems as the number of unknown arrival times. Although the problem is a convex optimization problem, we still need to improve its efficiency. If fact, it is not necessary to take all unknown arrival times and related constraints into account when we only want to optimize one arrival time. Intuitively, the arrival times near the one being optimized should be sufficient. Therefore, how to select the arrival times and the related constraints for the optimization problems becomes important.

We use a graph model to select the proper arrival times and the related constraints. Consider each unknown arrival time as a vertex. Add an edge between two vertices if there is at least one constraint involving these two arrival times. Then we extract a sub-graph for each arrival time for optimization. Figure 4 gives an example. An initial solution is generated based on two simple criteria. First, the extracted sub-graph contains a predetermined number of vertices. Second, The boundaries of the sub-graph needs to be far from the targeted vertex. Then, we employ a recently proposed massive graph cut algorithm BLP (balanced label propagation) [22] to tune the cutting boundaries so that the number of cut edges is minimized. The BLP algorithm iteratively tunes the cutting boundaries till optimal. Here, the number of vertices in each extracted sub-graph is an important parameter. Clearly, there is a tradeoff between the accuracy of bounds and the computational efficiency. In the evaluation section, we will show the impact of the *graph cut size*.

For each unknown arrival time, the extracted sub-graph

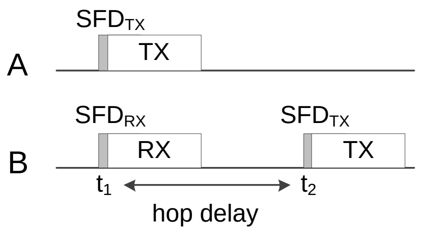


Figure 5. Node B measures the node delay locally by subtracting the two local times in the two SFD interrupts.

includes a number of vertices and edges. The upper bound and lower bound can be obtained by solving the two optimization problems considering the constraints related to these vertices and edges in the sub-graph. Since the number of constraints is much smaller, each optimization problem can be solved efficiently and the overall computing time is shortened significantly.

V. Implementation

In this section, we will describe the implementation details. Domo's implementation includes modification to node code to record the sum of delays and a program on PC that computes per-hop delays with the trace collected from the network. The PC side program mainly contains a data preprocessor and two kinds of optimizer. The data preprocessor is written in Perl. It transforms the raw packet trace into the three constraints. The SDP relaxation is done in the data preprocessor. Then two optimizers are used to obtain the estimated values and the bounds of the unknowns. Next, we will focus on the node side implementation which is mainly about how to record the sum of delays at each node.

Node side program is implemented in TinyOS2.1. When a node starts to receive a packet, it will generate a receive SFD (i.e., start frame delimiter) interrupt. In the interrupt handler, the node can record the current local time as t_1. After a period of time, the node starts to forward this packet to other nodes. In the transmit SFD interrupt handler, the node can record the current local time as t_2. Then $(t_2 - t_1)$ is the sojourn time (i.e., node delay) of the packet at this node. Figure 5 shows a simple example. Node A sends a packet to node B and node B forwards the packet to other node. The node delay of the packet at node B can be measured as the time difference between the two SFD interrupts at node B.

In order to reconstruct the per-hop delay of each received packet, each node need to record the sum of node delays in its local packets. A two-byte buffer is used to record the sum of node delays measured by the above technique. Algorithm 1 describes how the sum of node delays is recorded. For a local packet, its first node delay is recorded as the time difference between the generation time and the transmit SFD time (line 2, 3 and line 7). For a forwarded

Algorithm 1 Sum of node delays recording

```
1:  sum-hop-delays = 0
2:  event generates a local packet
3:      records current local time $t_1$
4:  event receives a packet
5:      records current local time $t_1$ in $SFD_{RX}$ handler
6:  event forwards a packet or sends a local packet
7:      records current local time $t_2$ in $SFD_{TX}$ handler
8:      sum-hop-delays += $t_2 - t_1$
9:      if sends a local packet then
10:         write sum-hop-delays to the transmission RAM
11:         sum-hop-delays = 0
```

Table I
OVERHEAD COMPARISON

Overhead/Approach	Domo	MNT	MsgTracing
Message	four bytes	four bytes	zero
Computation (node)	low	low	low
Computation (PC)	modest	modest	low
Memory (node)	low	low	high

packet, its node delay is recorded as the time difference between two SFD interrupts (line 4, 5 and line 7). Whenever a new local packet is being transmitted, the sum of node delays is attached at the end of the new local packet (line 10). The buffer is cleared when the new local packet is transmitted (line 11). Note that the sum of node delays is stored at the end of each local packet. The reason is that the sum of node delays is calculated in the transmit SFD of the new local packet which is being transmitted. Writing the sum of node delays to the transmit FIFO RAM is required to be done before the radio actually transmits that part of the packet.

A. Overhead

We then present the overhead in terms of message, computational and memory overhead of Domo and two related methods. Table I shows the results.

For message overhead, Domo introduces two data fields in each packet, the sum of node delays and a timestamp for recording the end-to-end delay in each packet [7]. In the current implementation, we use two bytes to store this sum-of-delays which can record a value up to 65 seconds when the precision is 1 millisecond. MNT also needs a timestamp in each packet to record the end-to-end delay. It does not have the sum of node delays, but requires the node ID of the first hop receiver. Since a two-byte timestamp is sufficient to record the end-to-end delay, Domo and MNT both have a message overhead of four bytes. MessageTracing does not have any message overhead since each node records the sent/received messages in its local storage.

In Domo, the computational overhead at the node side is negligible since there are only several arithmetic operations. MNT and MessageTracing also do not require high computational overhead at the node side. At the PC side, Domo needs to solve a number of optimization problems to obtain per-hop per-packet delays. Since all problems are convex optimization problems, they can be solved efficiently by a common desktop PC. In the evaluation section, we will show the detailed results about the execution time of Domo under different parameter settings. MNT needs to solve a maximum independent set problem at the PC side. Since its efficiency can be improved by approximation algorithms, it has modest computational overhead. The computational overhead of MessageTracing is relatively lower than other two methods.

The last type of overhead is node side memory overhead. In the current implementation, the memory overhead of Domo is less than 80 bytes. Consider a typical sensor node (e.g., TelosB) which has about 48k bytes ROM, this memory overhead is only 0.17% of the whole ROM. The memory overhead of MNT is also low. MessageTracing, however, has a high memory overhead since it needs to record all sent/received messages.

VI. EVALUATION

In this section, we evaluate the performance of the Domo in terms of accuracy of both the estimated value and the bounds. We also implement two related methods, MNT [14] and MessageTracing [15], to compare them with Domo.

A. Methodology

We conduct simulations based on TOSSIM [23] to evaluate the performance of the three approaches. TOSSIM is a standard simulator for TinyOS programs. We use Collection Tree Protocol [18] to generate an experimental network with 400 nodes uniformly distributed in a squared area. At the node side, Domo records the sum-of-delays into each packet. Then the collected packets are analyzed by the PC side program. All PC side calculations are done on a desktop PC with 2.3GHz quad-core CPU and 4GB memory. The per-hop per-packet delay is recorded by the simulator and used as ground truth.

The performance metric is the reconstruction accuracy. For accuracy of the estimated values, we calculate the average error of the estimated values compared with the ground truth. For accuracy of the bounds, we calculate the average uncertainty of the reconstructed bounds for each arrival times. Since the two related methods focus on different purposes, we use different methodologies to compare their performance with Domo.

MNT [14] is the state-of-the-art per-hop per-packet delay reconstruction approach. It can calculate the lower bound and upper bound of the per hop arrival time of some received packets. Therefore, we can compare the accuracy of the bounds calculated by MNT and Domo directly. We also calculate an estimated value for each per-hop delay from the upper bound and lower bound obtained by MNT. For simplicity, the estimated value is just the average value of the two bounds. Then we can compare the accuracy of MNT and Domo.

MessageTracing [15] proposes an efficient method to record messages sent and received in the local storage of each node. Based on an off-line analysis of the recorded traces, the order of some sending/receiving events can be reconstructed. In order to compare its performance with Domo, we use the estimated values of all arrival times to reconstruct the order of all sending/receiving events. Then we compare the order reconstructed by MessageTracing and Domo with the ground truth. For two sequences S_1 and S_2, we use the *average displacement* of all elements to measure the error. For example, the ground truth sequence is (a, b, c, d, e) and one reconstructed sequence is (b, a, e, d, c). Then the average displacement (i,e., error) can be calculated by (1+1+2+0+2)/5 which is 1.2.

We also evaluate the performance of Domo and two related methods under different network configurations in terms of packet loss rate and network scale. Further, we conduct simulations to reveal more system insights of Domo. In particular, we tune multiple parameters and evaluate their impacts to the performance of Domo. The parameters include the effective time window ratio (in Section IV.B) and the graph cut size (in Section IV.C).

B. *Results of Accuracy*

We first show the accuracy of Domo and MNT. Figure 6(a) shows each node's average node delay reconstructed by Domo and MNT. The ground truth is also shown in the figure for comparison. We can see that the accuracy of Domo is higher than MNT. For Domo, more than 70% of the reconstructed delays have an error less than 4ms. On average, the error of Domo is 3.58ms while the error of MNT is 9.33ms.

Figure 6(b) shows accuracy of bounds of Domo and MNT. The accuracy is measured as the distance between the lower bound and the upper bound of each per-hop per-packet delay. As we can see, Domo achieves higher accuracy of bounds than MNT. On average, the accuracy of bounds of Domo is about 16.11ms while the accuracy of bounds of MNT is about 40.97ms.

Figure 6(c) shows the displacements of Domo and MessageTracing [15]. The displacement mentioned in Section VI.A is a metric to compare the accuracy of Domo and MessageTracing. Since Domo can achieve high accuracy, its displacements are significantly smaller than MessageTracing. On average, the displacement of Domo is only about 0.03 while the displacement of MessageTracing is about 3.39.

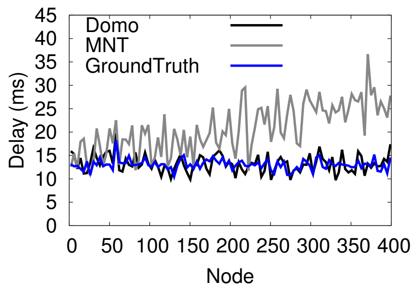
Impact of packet loss. We remove different number of packets randomly and use multiple approaches to reconstruct the per-hop delays of the rest packets. Figure 7(a) shows the accuracy of Domo and MNT under different packet loss rates. When we remove 10% to 30% packets from the original trace, Domo has an error from 3.62ms to 4.31ms on average and MNT has an error from 10.97ms to 12.29ms. Figure 7(b) shows the bound accuracy of Domo and MNT under different packet loss rates. When the loss rate varies, Domo achieves a bound from 16.21ms to 17.20ms on average and MNT achieves a bound from 41.03ms to 41.14ms. Figure 7(c) shows the displacements CDF of Domo and MessageTracing. When the loss rate varies, Domo achieves a displacement from 0.05 to 0.58 on average and MessageTracing achieves a displacement from 4.02 to 4.47. From the above results, we can see that the accuracy of Domo is higher than two related methods and keeps stable under different packet loss rates.

Impact of network scale. We also evaluate the performance of the three approaches in networks with 100, 225 and 400 nodes. In each network, all nodes are uniformly distributed in a squared area. Figure 8(a) shows the accuracy of Domo and MNT in different networks. Domo outperforms MNT in all network settings. On average, there is an error from 2.36ms to 3.58ms for Domo and from 4.51ms to 9.33ms for MNT. Figure 8(b) shows the bound accuracy of Domo and MNT. On average, there is a bound from 12.01ms to 16.11ms for Domo and from 25.56ms to 40.97ms for MNT. Figure 8(c) shows the displacements of Domo and MessageTracing. On average, there is a displacement from 0.001 to 0.03 for Domo and from 2.97 to 3.39 for MessageTracing. From these three figures, we can see that Domo achieves high accuracy in networks with different scales.

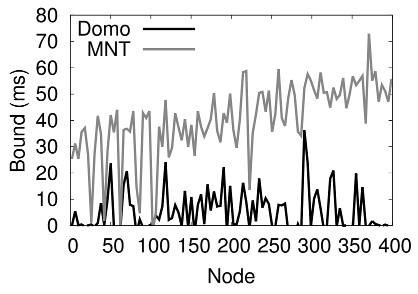
C. *System Insights*

In order to reveal more system insights of Domo, we evaluate its performance under different parameter settings. Specifically, we tune the two parameters in Domo, *effective time window ratio* and *graph cut size*.

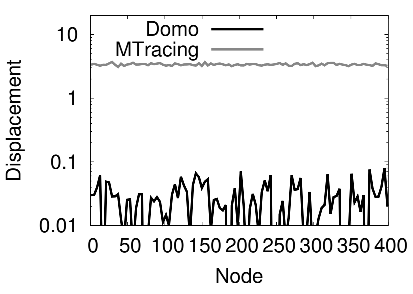
Impact of effective time window ratio. As mentioned in Section IV.B, this ratio affects the accuracy of Domo. We tune the ratio from 0.3 to 0.9. Since the estimated values of per-hop delays depend on this ratio, we show the accuracy of the estimated values in Figure 9(a). Although a larger ratio does decrease the accuracy, the impact of this parameter is not very significant. Figure 9(b) shows the execution time when the ratio varies. The larger the ratio is, the less time windows Domo needs. Therefore, the execution time per delay decreases under larger effective time window ratio. In the current implementation of Domo, this ratio is set to be 0.5. Under this setting, the average execution time for each per-hop delay at PC side is only 15ms.



(a) estimated value

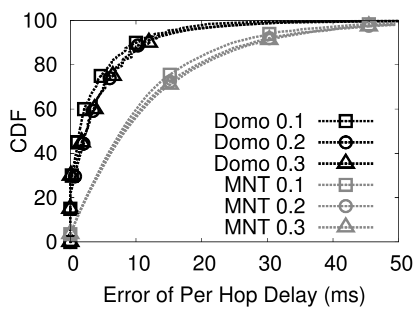


(b) bound

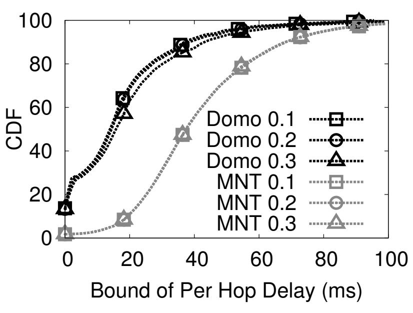


(c) displacement

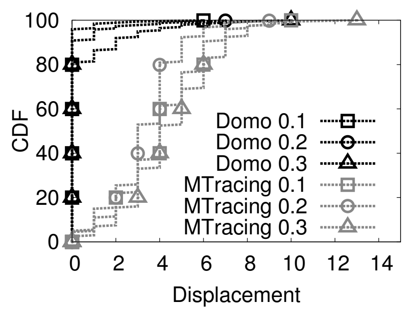
Figure 6. Per-hop delay reconstruction performance comparison. (a) Estimated value accuracy comparison of Domo and MNT. (b) Bound accuracy comparison of Domo and MNT. (c) Accuracy comparison of Domo and MessageTracing (measured by displacements).



(a) error

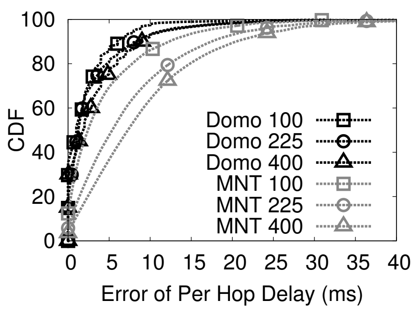


(b) bound

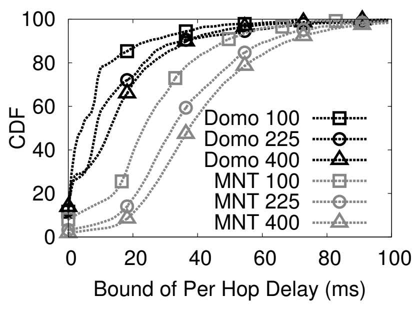


(c) displacement

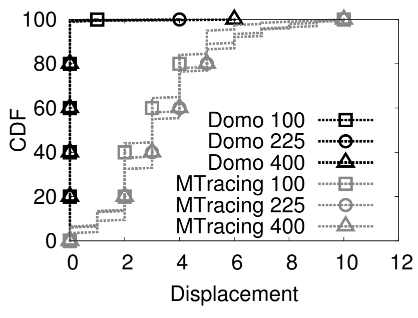
Figure 7. Impact of packet losses. (a) CDF of the reconstructed delay's error compared with the ground truth. (b) CDF of the calculated delay bound (measured by the distance between the upper bound and lower bound). (c) CDF of the displacement under different packet loss rates.



(a) error

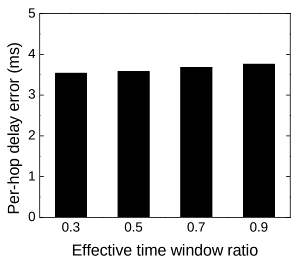


(b) bound

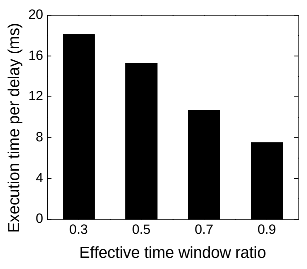


(c) displacement

Figure 8. Impact of network scale. (a) CDF of the reconstructed delay's error compared with the ground truth. (b) CDF of the calculated delay bound (measured by the distance between the upper bound and lower bound). (c) CDF of the displacement in networks with different number of nodes.



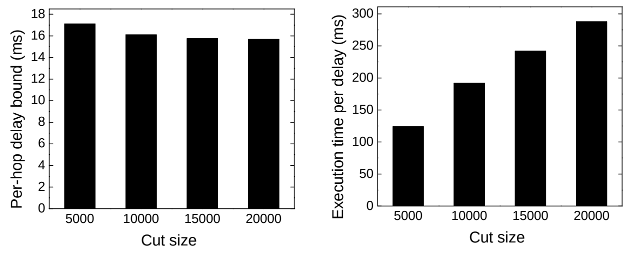
(a) error



(b) execution time

Figure 9. Impact of effective time window ratio. (a) The average errors of the reconstructed per-hop delays. (b) The average execution time per delay.

Impact of graph cut size. Another important parameter is the graph cut size as mentioned in Sectoin IV.C. We tune the graph cut size from 5000 to 20000. This parameter affects the accuracy of bounds and the reconstruction speed of Domo. Figure 10(a) shows the bound accuracy of Domo when the graph cut size varies. A lager graph cut size can increase the bound accuracy (i.e., tighter bound). Figure 10(b) shows the execution time per bound. From these two figures, we can see that Domo achieves high bound accuracy when the graph cut size is sufficiently large. In the current implementation, we configure the graph cut size to be 10000 and the average execution time per bound is 192ms.



(a) bound (b) execution time

Figure 10. Impact of graph cut size. (a) The average bounds of the calculated per-hop delay bounds. (b) The average execution time per delay bound.

VII. Conclusion

In this paper, we propose Domo, a passive, lightweight and accurate delay tomography approach to decomposing the packet end-to-end delay into each hop. Domo reconstructs per-hop per-packet delays by solving a set of optimization problems. SDP relaxation technique is used to improve the efficiency of Domo. Further, an improved time window method and a graph cut method are used to deal with large number of unknowns in the optimization problem. We implement Domo in TinyOS and evaluate its accuracy in TOSSIM. Evaluation results show that Domo achieves much higher accuracy compared with two related methods.

VIII. Acknowledgement

This work is supported by the National Science Foundation of China Grants No. 61202402, and No. 61370087, the Fundamental Research Funds for the Central Universities, the Research Fund for the Doctoral Program of Higher Education of China (20120101120179), Demonstration of Digital Medical Service and Technology in Destined Region, and the Open Research Fund of Zhejiang Provincial Key Lab of Data Storage and Transmission Technology, Hangzhou Dianzi University (No. 201304).

References

[1] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai, "Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest," in *Proceedings of SenSys*, 2009.

[2] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, "Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment," in *Proceedings of IPSN*, 2009.

[3] X. Mao, X. Miao, Y. He, T. Zhu, J. Wang, W. Dong, X. Li, and Y. Liu, "Citysee: Urban CO2 monitoring with sensors," in *Proceedings of INFOCOM*, 2012.

[4] O. Chipara, C. Lu, T. C. Bailey, and G. catalin Roman, "Reliable clinical monitoring using wireless sensor networks: Experiences in a step-down hospital unit," in *Proceedings of SenSys*, 2010.

[5] M. Patel and J. Wang, "Applications, challenges, and prospective in emerging body area networking technologies," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 80–88, 2010.

[6] M. Keller, L. Thiele, and J. Beutel, "Reconstruction of the correct temporal order of sensor network data," in *Proceedings of IPSN*, 2011.

[7] J. Wang, W. Dong, Z. Cao, and Y. Liu, "On the delay performance analysis in a large-scale wireless sensor network," in *Proceedings of RTSS*, 2012.

[8] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *Proceedings of SenSys*, 2007.

[9] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Efficient identification of additive link metrics via network tomography," in *Proceedings of ICDCS*, 2013.

[10] P. Sommer and B. Kusy, "Minerva: distributed tracing and debugging in wireless sensor networks," in *Proceedings of ACM SenSys*, 2013.

[11] 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.

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

[13] M. Lee, S. Goldberg, R. R. Kompella, and G. Varghese, "Fine-grained latency and loss measurements in the presence of reordering," in *Proceedings of the ACM SIGMETRICS*, 2011.

[14] M. Keller, J. Beutel, and L. Thiele, "How was your journey?: uncovering routing dynamics in deployed sensor networks with multi-hop network tomography," in *Proceedings of SenSys*, 2012.

[15] V. Sundaram and P. Eugster, "Lightweight message tracing for debugging wireless sensor networks," in *IEEE/IFIP DSN*, 2013.

[16] K. Papagiannaki, S. B. Moon, C. Fraleigh, P. Thiran, and C. Diot, "Measurement and analysis of single-hop delay on an ip backbone network." *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 908–921, 2003.

[17] M. Keller, L. Thiele, and J. Beutel, "Reconstruction of the correct temporal order of sensor network data," in *Proceedings of IPSN*, 2011.

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

[19] Y. Gao, W. Dong, C. Chen, J. Bu, G. Guan, X. Zhang, and X. Liu, "Pathfinder: Robust Path Reconstruction in Large Scale Sensor Networks with Lossy Links," in *Proceedings of ICNP*, 2013.

[20] X. Lu, D. Dong, Y. Liu, X. Liao, and L. Shanshan, "Pathzip: Packet path tracing in wireless sensor networks," in *Proceedings of MASS*, 2012.

[21] A. d'Aspremont and S. Boyd, "Relaxations and Randomized Methods for Nonconvex QCQPs," EE392o, Stanford University, 2003.

[22] J. Ugander and L. Backstrom, "Balanced Label Propagation for Partitioning Massive Graphs," in *Proceedings of ACM WSDM*, 2013.

[23] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proceedings of SenSys*, 2003.