# How Was Your Journey? Uncovering Routing Dynamics in Deployed Sensor Networks with Multi-hop Network Tomography

Matthias Keller, Jan Beutel, and Lothar Thiele
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
{kellmatt,beutel,thiele}@tik.ee.ethz.ch

## Abstract

In the context of wireless data collection, a common application class in wireless sensor networks, this paper presents a novel, non-intrusive algorithm for the precise reconstruction of the packet path, the per-hop arrival order and the per-hop arrival times of individual packets from partial in-band information at runtime. Information is reconstructed outside the network immediately after a packet is received at the sink. After establishing the correctness of our proposed algorithm, we evaluate its performance in testbed experiments using CTP and Dozer, two well-known data collection protocols. Foremost interested in obtaining a better understanding of the performance of long-term real-world deployments, *Multi-hop Network Tomography (MNT)* is applied to in total more than 140 million packets that have been obtained from three multi-year WSN deployments of the PermaSense project. The capabilities of the performance analysis of deployed systems using the proposed algorithm and methodology are demonstrated in a case study.

## Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

## General Terms

Performance, Design, Measurement

*Keywords*

Performance analysis, data collection, wireless sensor networks, network tomography, multi-hop

## 1 Introduction

Wireless sensor networks (WSNs) have proven their applicability in many scenarios, *e.g.*, ecosystem management [22], monitoring of buildings [5], and monitoring of natural hazards [2]. When preparing the initial installation, a considerable research and development effort is usually spent customizing and optimizing a WSN implementation for a given application scenario. Different methods and tools such as abstractions [9], simulators [18], testbeds [11] and diagnostics [33] have been developed for supporting and facilitating all phases of the life-cycle up to long-term operation.

However, appropriate tools are largely missing to support detailed performance analysis of deployed systems. When for example the network size, the sensing modalities or the environment, *e.g.*, due to the addition of new interferers like 4G/LTE equipment in the vicinity, are changing over the years of deployment, it is typically very hard to assess at which point in time the system must undergo minor, *e.g.*, new parametrization, or major, *e.g.*, addition or removal of features, modifications. While commonly available performance metrics, *i.e.*, data yield, end-to-end packet delays, radio duty-cycle, and link quality measurements are well-suited for giving an overview and estimating the general health of a deployed system, the root causes of an observed drop in the system performance remain mostly hidden.

Additional information are usually not available due to the costs and risks attached to their retrieval. For instance, transmitting extra information to aid such analysis in-band would increase the required bandwidth and is not always feasible. Furthermore, if a behavior is only observed after successful deployment, an addition of further data to be collected and transmitted is only possible with an update of all installed systems, either using a manual process, or by in-network reprogramming facilities. The latter of which is often regarded as an extra risk as it can easily render a system unusable [31]. Duplication of a similar system in a testbed setting and using this "replica" for analysis purposes in most cases suffers from economic feasibility and an inherent mismatch between production and test environments.

It is therefore desirable to devise comprehensive analysis capabilities based solely on the in-band information transmitted through a wireless sensor network. As a stepping stone for facilitating a passive but accurate health and performance monitoring of WSNs, this paper presents multi-hop network tomography (MNT), a novel, non-intrusive algorithm for reconstructing the travelled path, the per-hop arrival order, and the per-hop arrival times of individual packets at runtime. Information is reconstructed outside the network immediately after a packet has been received at the



*SenSys'12*, November 6–9, 2012, Toronto, ON, Canada.

sink. Concretely, we exploit the fact that packets are transferred through the network in a first-in first-out (FIFO) fashion. We find that the order in which packets arrive at the sink allows us create correspondences between packets that originate from independent sources but travel along similar paths. The apparent challenge lies foremost in the dynamics found in wireless multi-hop networks, *e.g.*, topology changes, packet reordering, and lost packets.

Specifically, multi-hop network tomography addresses the following problems: First, transferring path, per-hop arrival order, and per-hop arrival time information in-band does not scale for large networks as the amount of information grows linearly with the path length. As a result, the overall network performance may decrease due to congestion and higher packet loss rates [29] when sending more and larger packets further aggravating the capability to understand network behavior in detail. Second, active methods for extracting desired information would also require to modify the software that is running on the sensor nodes. Not only introducing the effort of reconfiguring a deployed system, valuable path, per-hop arrival order and per-hop arrival time information from historic data would remain unknown. While this work primarily targets low-power WSNs, the method as such can be applied to any multi-hop network. It is clear however that the most benefit is achieved in resource constrained scenarios commonly found in WSNs.

Apart from the analysis of deployed systems, MNT offers advantages in potentially any scenario in which no other communication channels, *e.g.*, serial ports, for extracting performance data are usable. For example, adding load on the serial interface for outputting additional performance data is also unfavorable in certain testbed settings, *e.g.*, when the execution timing must not be changed.

The contribution of this paper is as follows:

- We present the MNT algorithm for reconstructing the packet path, the per-hop arrival order, and the per-hop arrival timing of individual packets.
- Based on a formal model of a real system, we proof the correctness of results obtained, *i.e.*, that extracted path, order and timing information match with ground truth.
- We validate the correctness of our implementation in extensive experiments with two well-known communication protocols, *i.e.*, CTP [10], and Dozer [4], on testbeds of up to 90 nodes size. Here, testbed infrastructure allows us to extract ground truth without the need for congesting the in-band communication resource under investigation. We evaluate the performance of our algorithm in terms of the fraction of packets that can be reconstructed in various settings. The scalability of the approach is discussed using simulation.
- The application of the MNT algorithm to more than 140 million packets from three deployed systems is presented in a case study.

Related work including a discussion of the novelty of this work is presented in Section 2. Section 3 motivates the underlying problem of reconstructing data from partial information, the core principles used in multi-hop network tomography are presented in Section 4. Assumptions made when designing the MNT algorithm are introduced in Section 5, the full algorithm is presented in Section 6. Section 7 presents our validation done on real hardware and in simulation, our case study based on data from real-world deployments is presented in Section 8. The broader applicability and limitations of the MNT algorithm are discussed in Section 9, Section 10 concludes this paper.

## 2 Related Work

*A. Performance of Wireless Sensor Networks*

The performance of wireless sensor networks has been intensively studied at several layers. For instance, [34] and [30] study the link-layer performance in numerous configurations and environments. Various protocol papers, *e.g.*, CTP [10] and the low-power wireless bus [8], discuss the performance of the routing layer. The end-to-end application performance is subject of several deployment reports, *e.g.*, [17, 1, 12]. An extensive cross-layer performance study is provided by [24]. Located on the intersection between the routing and the application layer, the purpose of this work is to passively reconstruct hidden network performance data.

*B. Network Health Monitoring*

For dealing with the inherent challenges found in the long-term operation of WSNs, *e.g.*, hardware failures, several solutions for the run-time monitoring of deployed systems have been proposed [28, 20]. While the problem of how to combine measured information for inferring a root cause is orthogonal to this work, such systems could potentially benefit from per-packet path and timing information that is provided by the MNT algorithm. Ideally, we expect reconstructed per-hop timing information to even facilitate the development of health monitoring systems that are also able to automatically report small variations of the system performance before a major incident happened.

*C. Wired Network Tomography*

Network tomography is an important tool for network monitoring in wired IP networks. Without the need for cooperation of involved components, *e.g.*, routers, network structure and link-level performance characteristics, *e.g.*, delay or packet loss, are measured based on the travel of actively inserted probes. The problem of network tomography in wired networks has been well-studied, an extensive overview of available methods is given in [6]. In most cases, the studied problem is either to reconstruct the network structure only, or to determine link-level performance measurements for an a priori known network topology. Regarding our goal of reconstructing both the network structure and link-level characteristics, we found the work of Rabbat *et al.* [27] as the possibly earliest work that already covers both dimensions in the wired scenario.

*D. WSN Network Tomography*

Network tomography algorithms for wireless sensor networks are restricted by WSNs supporting much less probing traffic than wired networks. Nguyen *et al.* [25] propose the application of statistical methods, *i.e.*, Maximum likelihood and Bayesian approaches, for the identification of lossy links. While the network topology is assumed to be known, the method of Nguyen *et al.* also allows for multiple, dynamic topologies by splitting a trace into so called "routing time slots" in which the topology is assumed to be stable.

Being interested in learning an unknown network topology, Liu *et al.* [20] propose an active marking scheme for the reconstruction of topology information at the sink. Here, extracted path information is not guaranteed to be correct.

Without adding extra probing traffic to the network, the MNT algorithm extracts all information from already existing application traffic. To the best of our knowledge, this is the first work that aims for the reconstruction of detailed per-packet information while also giving guarantees on the correctness of extracted information.

## 3 Exploiting Information Implicitly Given

Measurements taken inside a sensor network provide additional detail for the understanding of an observed end-to-end system performance. For example, an observed end-to-end packet delay might have several causes, *e.g.*, transmission failures, back-pressure, or unfair resource allocation. However, the amount of information that can be transferred in-band is limited as additional load potentially threatens the performance of the system under investigation.

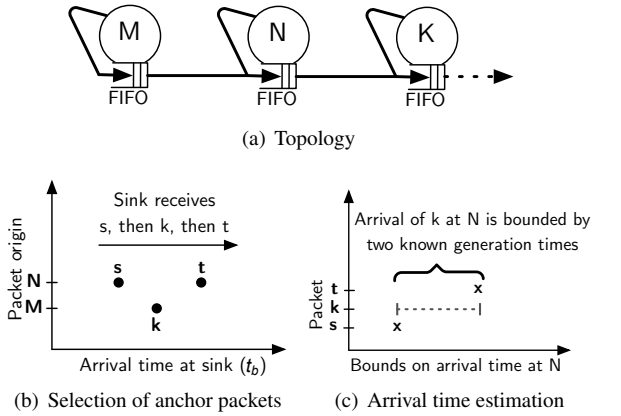
For enabling detailed performance analyses of sensor networks in spite of resource constraints this paper wants to answer the following questions: Is it sufficient to transmit only partial information in-band in order to reconstruct missing information outside the network? What information is commonly transferred and thus already available in sensor network applications? Which useful information is implicitly given by the structure and behavior a-priori known from a given WSN application, and thus does not need to be transmitted over the precious communication resource? How much information of which detail and accuracy can be reconstructed outside the sensor network? Are corresponding methods applicable to a broader set of applications?

As one concrete example, this paper presents multi-hop network tomography (MNT) for the reconstruction of the path, the per-hop arrival order, and the per-hop arrival time of individual packets from partial information.

## 4 Wireless Multi-Hop Tomography

In multi-hop data collection applications, sensor nodes have the dual functionality of (i) generating packets and (ii) forwarding packets of other nodes that are more distant to the sink. Generated packets typically include certain application header information, *e.g.*, the source address and a packet generation timestamp. For reconstructing information at the sink, multi-hop network tomography exploits the fact that state-of-the-art protocols like CTP maintain a single packet queue to which both locally generated and forwarded packets are added while traveling through the network. In the common case correspondences created between forwarded and locally generated packets are still visible at the sink, *i.e.*, packets arrive at the sink in the same order as they left a node within the network. The path, the per-hop arrival order and the per-hop arrival timing of individual packets are reconstructed by a per-hop correlation of information from both, locally generated and forwarded packets.

In the exemplary situation in Figure 1, packets $s$ and $t$ were consecutively generated at a node $N$. In contrast, packet $k$ was generated at another node $M$. Minimal topology information is provided by the address of the first-hop receiver,




(a) Topology

(b) Selection of anchor packets

(c) Arrival time estimation

**Figure 1. Reconstruction of packet path and arrival time bounds of packet $k$ based on information from packets $s$ and $t$.** Packets $s$ and $t$ were subsequently generated at node $N$. Packet $k$ from node $M$ arrived at $N$ after the generation of $s$, but before the generation of $t$.

*e.g.*, packet $k$ reports node $N$, packets $s$ and $t$ report node $K$, that is transmitted with every packet. Packet $k$ arrived at $N$ after the generation of $s$, but before the generation of packet $t$. Given that none of the packets was either lost, duplicated or reordered in this example, the observable order of arrival at the sink, see Figure 1(b), matches with the non-observable order of arrival at the intermediate node $N$, see Figure 1(c).

Packets $s$ and $t$ are selected as so called "anchor packets" and used for reconstructing (i) at which time packet $k$ arrived at node $N$, and (ii) to which node packet $k$ was forwarded after leaving node $N$. Concretely, information is inferred from (i) the known packet generation time of packets $s$ and $t$, see Figure 1(c), and (ii) the first-hop receiver reported by packets $s$ and $t$. For every packet, this procedure is repetitively applied until the packet has been traced up to the sink.

Due to phenomena common to WSNs, *e.g.*, packet loss, packet duplication and packet reordering, observations made at the sink might not match with the reality. Therefore, the MNT algorithm has to assess for each packet if information can safely be reconstructed, or if there is the risk of obtaining incorrect information. Based on the formal model of a system that is presented in the next Section 5, the description of the full MNT algorithm is situated in Section 6.

## 5 System Model

This section summarizes assumptions made and variables used by the MNT algorithm. Details on the adaptation of this generic model to more complex systems are presented in Section 5.1, *i.e.*, multiple sinks and systems that include multi-layered storage architectures.

We assume a multi-hop wireless sensor network that consists of a number of static sensor nodes and a sink. Communication is based on a tree-based routing protocol. The network operation is subject to phenomena common to wireless sensor networks, *i.e.*, topology changes, packet loss, packet duplication and packet reordering. All nodes produce and relay data in the sense of a data collection application. Nodes do not have access to global timing information and rely on a local clock.

| | |
|---|---|
| **Packet application headers** | |
| $o(k)$ | Source node network address |
| $\tilde{t}_g(k)$ | Estimated packet generation time |
| $p(k)$ | Network address of the current parent |
| **Added on arrival at the sink** | |
| $t_b(k)$ | Arrival time at the sink |
| **From analysis, packet headers, or post-processing** | |
| $\Delta^{u,l}(k)$ | Upper and lower bounds on the accuracy of the estimated packet generation time $\tilde{t}_g(k)$ |
| $id_N(k)$ | Packet index reflecting the correct order of generation for packets originating from a node $N$ |

**Table 1. Overview of system model variables**

**FIFO send queue.** All sensor nodes have the dual functionality of (i) generating data locally and (ii) forwarding packets received from other nodes. Sensor nodes maintain a finite FIFO queue for all outgoing packets. A packet is immediately added to this queue after generation in the local application or arrival on the radio. If connected, a sensor nodes transmits the contents of its send queue to the next hop. If single-hop communication fails, a sensor nodes keeps retransmitting the currently selected packet until this packet was successfully acknowledged by the parent node, or if the maximum number of transmission attempts is reached.

**Delay-tolerant data generation.** Sensor nodes are generating data, *i.e.*, recording status information, that is transmitted to the sink. The timing of data generation is unspecified. Sensor nodes may also generate data while disconnected from the network. In this case, packets are buffered on the sensor node until connectivity is reestablished.

We further assume that the information listed in Table 1 is known for each packet $k$ that has been received at the sink.

**Timing information.** The arrival time at the sink $t_b(k)$ is measured on a perfect clock and known for any packet $k$. To allow for packet time-stamping mechanisms with inaccuracies, we assume that only an estimate $\tilde{t}_g(k)$ of the packet generation $t_g(k)$ is accessible for all packets. The error of this estimate is bounded by $\tilde{t}_g(k) - \Delta^l(k) \leq t_g(k) \leq \tilde{t}_g(k) + \Delta^u(k)$. Here, $\Delta^u(k)$ and $\Delta^l(k)$ denote the upper and lower bounds on the error of the time-stamping mechanism used. $\Delta^u(k)$ and $\Delta^l(k)$ are not transmitted as part of a packet $k$, but assumed to be derived from an analysis, *e.g.*, an analysis on the accuracy of the clock synchronization scheme used. As packets are immediately added to the send queue after generation, $\tilde{t}_g(k)$ is a valid proxy for the arrival time of a packet $k$ at the send queue of its source.

**Sequencing information.** We assume the existence of a unique index $id_N(k)$ that yields the correct order of packet generation for packets originating from an individual node $N$: $id_N(u) > id_N(v)$ iff $t_g(u) > t_g(v)$. Packet loss is an artifact that is common to wireless multi-hop networks and therefore cannot be avoided. We assume that $id_N(k)$ also allows us to detect packet loss: Given a packet $v$ generated at $N$ in direct succession of a packet $u$, it must also hold that $id_N(v) \equiv id_N(u) + 1$. Practically, $id_N(k)$ can be obtained by submitting $id_N(k)$ as part of each packet, or by using post-processing algorithms that can reconstruct $id_N(k)$, *e.g.*, [14].

**Piggy-backed topology information.** Each packet $k$ carries a source address $o(k)$ and a first-hop receiver address $p(k)$. For retrieving up-to-date information, $p(k)$ is updated immediately before each try of transmitting the packet to the current parent. While transferring parent information is a common best practice in many real applications, *e.g.*, for being able to generate snapshots of the network topology, certain protocols, *e.g.*, [32], even require this information to be transmitted for their operation, *e.g.*, for enabling passive neighbor discovery. For applications that do not yet transfer first-hop receiver information, a single field, we follow the argumentation of Liu *et al.* [20] while considering the introduced traffic overhead to be negligible.

For the MNT algorithm to be able to trace packets based on the topology information described, we need to add an assumption concerning the observability of parent changes: For our further argumentation (see Section 6.3.1), it must hold that the parent of $N$ cannot have changed between the successful transmission of consecutively, locally generated packets $u$ and $v$, if we observe $p(u) \equiv p(v)$. Therefore, we can only allow for up to one parent change between the successful transmission of locally generated packets $u$ and $v$. While this is satisfied in the common case, properly handling the rare situation of more than one consecutive parent change between the transmission of two locally generated packets would only ask for a small modification of the protocol operation, *e.g.*, letting nodes locally count the number of parent changes since the most recently transmitted topology information and mark packets that were forwarded after the second, consecutive change.

## 5.1 Modeling More Complex Systems

None or only little extra information is required for adapting our model to significantly more complex systems:

**Multi-layered storage architecture.** For supporting high sampling rates and disconnected operation, recent system designs envision sensor nodes to be equipped with extra hardware for bulk storage, *e.g.*, FRAM [5] or SD memory cards [2]. Instead of maintaining a single packet queue only, locally generated packets are firstly added to a second queue that is situated on the added storage. Extra information is needed for supporting multi-layered storage architectures, namely the time spent in other queues before a packet was ultimately added to the send queue.

**Multiple sinks.** Sensor nodes may concurrently transmit packets to multiple sinks, *e.g.*, [23]. Here, sensor nodes are at the same time part of multiple concurrent tree topologies. A concrete multi-sink system conforms to our system model if its operation can be abstracted as the concurrent operation of multiple single-sink data collection trees that individually conform to our system model. Data received at each sink is then analyzed separately.

# 6 Safe Information Reconstruction

The MNT algorithm for reconstructing the travelled path, the per-hop arrival order, and the per-hop arrival times of individual packets is based on three core principles: First, path information is reconstructed by a per-hop correlation

of locally generated and forwarded packets. Here, we exploit that locally generated packets include the address of the first-hop receiver. Second, the per-hop arrival order of both locally generated and forwarded packets is inferred from the observed order of packet arrival at the sink. Third, packet generation time information of locally generated packets is used to bound the per-hop arrival time of forwarded packets that arrived at a respective node immediately before or after packet generation.

The correctness of information inferred is threatened by phenomena common to WSNs, namely topology changes, packet loss, and packet reordering. In the context of the MNT algorithm, we need to address the following two problems: Firstly, observed and real packet paths of individual packets must match. Therefore, we can only argue about packets for which we can guarantee that those packets in any case can only have travelled along exactly one path. Likewise, per-hop order information inferred from the observed order of arrival at the sink must also match with the real packet order at packet queues within the network. Path changes are the single source of packet reordering in multi-hop networks. Thus, we must ensure that a packet can not have been reordered due to a path change before we are allowed to reconstruct information of this packet or to use this packet for reconstructing information of other packets.

Given a trace $\mathcal{P}$ of received packets, the first step of the MNT algorithm is to determine the set $\mathcal{R}$ of so called "reliable" packets. For packets within this set, we introduce the concept of "anchor packets" for reconstructing packet path, per-hop order, and per-hop arrival times of individual packets at the sink: Given a forwarded packet $k$ that was forwarded to a node $N$, "anchor packets" $s$ and $t$ correspond to locally generated packets that were generated at node $N$ immediately before and after the arrival of $k$ at $N$. Generation time information and first-hop receiver information of both $s$ and $t$ are used to firstly bound the time of arrival of $k$ at $N$, and secondly to deduce the next hop to which $k$ travelled after $N$.

In the following, we will first describe the concept of information reconstruction using "anchor packets" in Section 6.1. The correctness of the results obtained is threatened by artifacts of path changes. After specifying the concrete impact on our problem in Section 6.2, the following Section 6.3 describes the properties of "reliable" packets and how a set $\mathcal{R}$ of "reliable" packets can be determined given a trace of received packets. Further extensions for improving extracted information using forward and backward reasoning are presented in Section 6.4. Reconstructed timing information is often too pessimistic and can be improved by correlating information of multiple packets.

## 6.1 Packet Correlation using Anchor Packets

This section formally describes the most integral concept of information reconstruction using "anchor packets". For clarity and brevity, the following description assumes that all involved packets are members of the corresponding set of "reliable" packets $\mathcal{R}$, and therefore reconstructed information is correct. The construction of a set of "reliable" packets will be described afterwards in Section 6.3.

Given a packet $k$, we want to reconstruct the following information:

- **Packet path** $\mathcal{N}_k$: Starting at the packet source $o(k)$, the ordered set $\mathcal{N}_k$ contains all nodes that packet $k$ visited until arriving at the sink node $S$. The order of items in $\mathcal{N}_k$ reflects the order of visited nodes.
- **Queue index** $qid_N(k)$: For all nodes $N$ that $k$ visited, *i.e.*, $\forall N \in \mathcal{N}_k$, we want to build a queue index $qid_N$ so that $qid_N$ reflects the order of packet arrivals at $N$: The queue index is larger, *i.e.*, $qid_N(m) > qid_N(n)$, iff packet $m$ arrived at $N$ after another packet $n$, *i.e.*, $t_a(N,m) > t_a(N,n)$. In contrast to the already known packet index $id_N(k)$, the queue index $qid_N(k)$ provides not only the sequence of locally generated packets, but also that of forwarded packets.
- **Bounds on queue arrival time** $t_a^{u,l}(N,k)$: For all nodes $N$ that $k$ visited, *i.e.*, $\forall N \in \mathcal{N}_k$, we want to bound the unknown queue arrival time $t_a(N,k)$ so that $t_a^l(N,k) \leq t_a(N,k) \leq t_a^u(N,k)$.

The reconstruction process for any packet $k$ starts at the source node $o(k)$ where we are immediately able to assign the queue index $qid_{o(k)}(k)$, the arrival time bounds $t_a^u(o(k),k)$ and $t_a^l(o(k),k)$, and the first two entries of the packet path $\mathcal{N}_k$. Concretely, the queue index $qid_{o(k)}(k)$ is initialized with a multiple of the known packet index $id_{o(k)}(k)$, *i.e.*, $qid_{o(k)}(k) := id_{o(k)}(k) \cdot c$ with $c > 1$. By multiplying the packet index $id_N(k)$, we give room for adding forwarded packets that arrived in between locally generated packets. Therefore, the multiplier $c$ must be larger than the maximum number of forwarded packets that can arrive in between two consecutively generated packets.

Next, arrival time bounds are initialized using upper and lower bounds on the packet generation time:

$$t_a^l(o(k),k) := \tilde{t}_g(k) - \Delta^l(k) \quad (1)$$

$$t_a^u(o(k),k) := \tilde{t}_g(k) + \Delta^u(k) \quad (2)$$

Likewise, the arrival time $t_a^{u,l}(S,k)$ at the sink corresponds to the known time of arrival at the sink $t_b(k)$: $t_a(S,k)^{u,l} := t_b(k)$. The packet path $\mathcal{N}_k$ is initialized with $\mathcal{N}_k := \{o(k)\}$. The next hop corresponds to the known first-hop receiver $p(k)$, we initialize $N^* := p(k)$, and start searching for anchor packets $s$ and $t$ at $N^*$:

$$s := \arg\max_x t_b(x) \text{ for all } x : o(x) \equiv N^* \wedge t_b(x) < t_b(k) \quad (3)$$

$$t := \arg\min_x t_b(x) \text{ for all } x : o(x) \equiv N^* \wedge t_b(x) > t_b(k) \quad (4)$$

Regarding all packets that were generated at $N^*$, $s$ is the packet that arrived at the sink latest before $k$. Likewise, packet $t$ arrived at the sink earliest after $k$. While we can only observe the order in which $s$, $t$ and $k$ arrived at the sink, (3) assumes that if $t_b(s) < t_b(k)$, it also holds that $t_a(N,s) < t_a(N,k)$. Likewise, (4) assumes that if $t_b(t) > t_b(k)$, it also holds that $t_a(N,t) > t_a(N,k)$. We will show in Section 6.3 that this assumption is backed by packets $s$, $k$ and $t$ being members of the corresponding set $\mathcal{R}$ of "reliable" packets.

The complete packet tracing algorithm is shown in Algorithm 1. The anchor packet selection is situated between lines 5 and 8. Tracing must firstly stop, if we cannot find

anchor packets $s$ and $t$ (line 9), if found packets $s$ and $t$ were not consecutively generated (line 10), *i.e.*, not all relevant packets are also part of the set of "reliable" packets, or if we cannot safely determine the next hop (line 11). The lowest free queue index $qid_{N^*}(k)$ that is smaller than the queue index $qid_{N^*}(t)$ of the anchor packet $t$ is determined in line 13.

**Algorithm 1:** Reconstruction of the path, the per-hop arrival order, and per-hop arrival times of a packet $k$

**input**: Packet $k$ with origin $o(k)$, first-hop receiver $p(k)$ and arrival time at the sink $t_b(k)$. $k \in \mathcal{R}$

1 **begin**
2 $\quad t_a^l(o(k),k) \longleftarrow \tilde{t}_g(k) - \Delta^l(k)$ ;
$\quad t_a^u(o(k),k) \longleftarrow \tilde{t}_g(k) + \Delta^u(k)$ ;
3 $\quad \mathcal{N}_k \longleftarrow \{o(k)\}$ ; $N^* \longleftarrow p(k)$ ;
4 $\quad$ **while** $N^* \not\equiv S$ **do**
5 $\quad\quad s \leftarrow \arg\max_x t_b(x)$
6 $\quad\quad\quad$ for all $x : x \in \mathcal{R} \wedge o(x) \equiv N^* \wedge t_b(x) < t_b(k)$ ;
7 $\quad\quad t \leftarrow \arg\min_x t_b(x)$
8 $\quad\quad\quad$ for all $x : x \in \mathcal{R} \wedge o(x) \equiv N^* \wedge t_b(x) > t_b(k)$ ;
9 $\quad\quad$ **if** $s \equiv \{\}$ ***or*** $t \equiv \{\}$ **then** break ;
10 $\quad\quad$ **if** $id_{N^*}(s) \not\equiv id_{N^*}(t) - 1$ **then** break ;
11 $\quad\quad$ **if** $p(s) \not\equiv p(t)$ **then** break ;
12 $\quad\quad \mathcal{N}_k \longleftarrow \mathcal{N}_k \cup \{N^*\}$ ;
13 $\quad\quad qid_{N^*}(k) \longleftarrow 1 + \max_{qid_{N^*}} qid_{N^*} < qid_{N^*}(t)$ ;
14 $\quad\quad t_a^l(N^*,k) \longleftarrow \tilde{t}_g(s) - \Delta^l(s)$ ;
$\quad\quad t_a^u(N^*,k) \longleftarrow \tilde{t}_g(t) + \Delta^u(t)$ ;
15 $\quad\quad N^* \longleftarrow p(s)$ ;
16 $\quad$ **end**
17 $\quad t_a^l(S,k) \longleftarrow t_b(k)$ ; $t_a^u(S,k) \longleftarrow t_b(k)$ ;
18 **end**

## 6.2 The Problem with Path Changes

Regarding our scheme of inferring information from observations made at the sink, path changes in the network can introduce two kinds of difficulties: (i) Observations may yield more than one possible path along which a packet $k$ may have travelled. In this case, it is not further decidable which of those multiple paths corresponds to the correct path. (ii) Packets can get reordered, and thus arrive at the sink in a different order than they arrived at individual queues within the network. In both cases, inferred information is no longer guaranteed to be correct.

Let us outline those two problems in the following brief example of a parent change: In Figure 2, we see that the parent of a node $N$ changes from node $K$ to node $L$ at a time $t_x$. Let us assume that packets $n_{K1}$, $n_{K2}$ and $n_L$ were generated at node $N$. Additionally, there is a packet $k$ that was forwarded from a node $M$ to node $N$ in between the generation of $n_{K1}$ and $n_{K2}$. While packets $n_{K1}$, $k$ and $n_{K2}$ were still forwarded to node $K$, packet $n_L$ was the first packet that went to the new parent $L$. Although both paths individually forward packets in the correct order, a larger delay for packets traveling along the old path can lead to packets arriving out of order when packets from both paths join at the sink. For example, we now assume that packet $n_L$ arrived at the sink before packets $k$ and $n_{K2}$, thus out of order. This can lead to

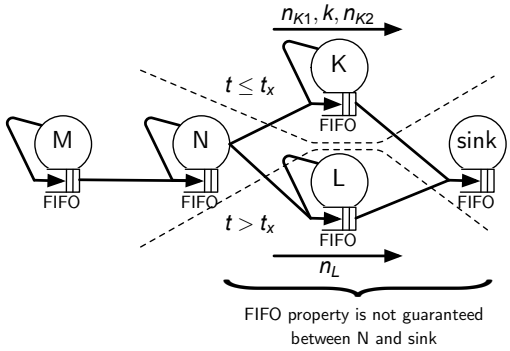



**Figure 2. Possible FIFO violation.** In this example, the parent of node $N$ changes from node $K$ to node $L$ at time $t_x$. Since packets on both paths may experience arbitrary delays, packets generated just before and after the topology change might not arrive at the sink in the order of generation anymore. This also affects packets being generated at more distant nodes, *i.e.*, node $M$.

the following two problems: (i) Observations from the sink no longer suggest that $k$ can only have travelled along its real path $\mathcal{N}_k := \{M,N,K,S\}$, but also along the new path over node $L$. (ii) Although $n_{K1}$ and $n_{K2}$ are the real "anchor packets" of $k$, the order of arrival at the sink would suggest to wrongly select $n_L$ and $n_{K2}$.

## 6.3 Finding a Set of Reliable Packets

With the goal of ensuring that packet correlation using anchor packets is only applied when this procedure is safe, we propose to limit information reconstruction to a subset of the set $\mathcal{P}$ of received packets, namely to a set $\mathcal{R} \subseteq \mathcal{P}$ of "reliable" packets.

*A packet $k$ is reliable,* i.e.*, $k \in \mathcal{R}$, if it fulfills two properties: From our observations at the sink, we can guarantee that (i) packet $k$ can only have travelled along exactly one path $\mathcal{N}_k$, and that (ii) the order relation between packet $k$ and any other packet $m \in \mathcal{R}$ is consistent along all packet queues in the network including the sink.*

Regarding the first condition, the underlying problem is that our observations made at the sink can yield multiple, ambiguous paths when it is not decidable if a packet $k$ left a node $N$ before or after node $N$ switching to another parent. Needed timing and order information for deciding this problem are yet unknown. While multiple choices might lead to selecting another, but the real path, we can only reason about packets that can only have travelled along exactly one path.

The second condition allows us to reason about the packet arrival order at packet queues within the network from observations at the sink. While attached packet sequencing information $id_N(k)$ yields the sequence of packet generation at a node $N$, the condition for reliable packets is stronger as it covers both locally generated and forwarded packets at an arbitrary node $N$. Formally, the second condition asks for the following: *For any packets $m,n \in \mathcal{R}$, it holds that* $\forall N \in \mathcal{N}_m \cap \mathcal{N}_n : t_a(N,m) > t_a(N,n)$ *iff* $t_b(m) > t_b(n)$. Here, $t_a(N,m)$ and $t_a(N,n)$ are the unknown arrival times of packets $m$ and $n$ at node $N$. $\mathcal{N}_m \cap \mathcal{N}_n$ denotes the set of nodes that both $m$ and $n$ visited, and therefore the set of nodes for which an order relation between packets $m$ and $n$ exists.

We will now present how the MNT algorithm consecutively solves those two problems. First, we present a worst-case analysis that decides if a packet can only have travelled along exactly one path, or if there are ambiguities. Second, we will show in Section 6.3.2 how available packet sequencing information $id_N(k)$ of locally generated packets can be used for solving the second problem. The complete algorithm for finding a set of "reliable" packets is finally presented in Section 6.3.3.

### 6.3.1 Per-hop Worst-Case Path Analysis

In the following, we will describe how we can test whether a packet $k$ can only have left a node $N$ to a unique next hop, or whether there are ambiguities. Starting at the known first-hop receiver $p(k)$, the following procedure is carried out per-hop until either the sink is reached, or we must conclude that we cannot decide along which path packet $k$ travelled along.

Let us assume a packet $k$ that was forwarded to a node $N$. We now want to determine whether we can guarantee that $k$ can only have left node $N$ to exactly one next hop, and if yes, to which next hop. In this two-part worst-case analysis, we first determine the set $\mathcal{W}$ of locally generated packets in between which $k$ may have arrived at $N$. Secondly, we analyze whether all packets in $\mathcal{W}$ were forwarded to the same next hop. Here, we consider both observable parent changes, *i.e.*, packets reporting different first-hop receivers, and potential, hidden parent changes, *i.e.*, a lost packet.

We start with determining the set $\mathcal{W}$ of locally generated packets in between which $k$ may have arrived at $N$. As the arrival time of packet $k$ at node $N$ is yet unknown, we must resort to bounding the arrival $t_a(N,k)$ of packet $k$ at node $N$ by the lower bound on the generation time, *i.e.*, $\tilde{t}_g(k)-\Delta^l(k)$, and the arrival time at the sink $t_b(k)$, *i.e.*, $t_g^l(k) \leq t_a(N,k) \leq t_b(k)$. Based on those bounds, we determine packets $u$ and $v$ that were generated at $N$ immediately before and after the earliest and latest arrival of $k$ at $N$, respectively. Based on the packet indexes $id_N(u)$ and $id_N(v)$, we then determine the set $\mathcal{W}$ of packets in between which $k$ can have arrived at $N$. Here, $\mathcal{W}$ includes $u$, $v$, and all received packets that were generated after and before $u$ and $v$ at node $N$, respectively.

$$u := \arg\max_x \tilde{t}_g(x)+\Delta^u(x) \text{ for all } x : o(x) \equiv N \\ \wedge \tilde{t}_g(x)+\Delta^u(x) < \tilde{t}_g(k)-\Delta^l(k) \quad (5)$$

$$v := \arg\min_x \tilde{t}_g(x)-\Delta^l(x) \text{ for all } x : o(x) \equiv N \\ \wedge \tilde{t}_g(x)-\Delta^l(x) > t_b(k) \quad (6)$$

$$\mathcal{W} := \Big\{ w \mid o(w) \equiv N \\ \wedge id_N(u) \leq id_N(w) \leq id_N(v) \Big\} \quad (7)$$

THEOREM 1. *We defined $\mathcal{W}$ as the set of all packets generated at a node $N$ in between which a forwarded packet $k$ must have arrived at $N$. Given a packet $k$ that arrived at a node $N$ in between two locally generated packets $m,n \in \mathcal{W}$, we can guarantee that $k$ was forwarded to a single possible next hop if (1) all packets $m \in \mathcal{W}$ were received at the sink,* i.e.*, no packet of $\mathcal{W}$ was lost, and (2) all packets $m \in \mathcal{W}$ carry the same first-hop receiver.*

PROOF. For the proof, we now go back to our formal model where we require that the first-hop receiver can not change more than once between the successful transmission of two consecutively, locally generated packets. Concretely, for any parent change, there must be at least one transmitted packet $m$ that carries the new parent as its first-hop receiver $p(m)$. Based on this assumption, there can be only one possible next hop if (1) all packets $m \in \mathcal{W}$ were received, and (2) all packets $m,n \in \mathcal{W}$ were forwarded to the same first-hop receiver, *i.e.*, $\forall m,n \in \mathcal{W} : p(m) \equiv p(n)$. □

Practically, packet loss is detected using the packet index $id_N$: No packets between $u$ and $v$ were lost, iff the number of elements $|\mathcal{W}|$ of the duplicate-free set $\mathcal{W}$ equals the difference of $id_N(v)$ and $id_N(u)$ plus one, *i.e.*, $|\mathcal{W}| \equiv id_N(v)-id_N(u)+1$. If packet loss is detected, we cannot add packet $k$ to the set $\mathcal{R}$ of reliable packets.

### 6.3.2 Exclusion of Packet Reordering

If a packet $k$ is guaranteed to have travelled along exactly one path, we can add $k$ to the set $\mathcal{R}$ of "reliable" packets, if we can also guarantee that the order relation between packet $k$ and any other packet $m \in \mathcal{R}$ is consistent along all packet queues in the network including the sink.

Here, our approach is to solve this problem at its source, namely parent changes within the network. Concretely, we want to analyze per hop if packets were reordered due to a parent change at this node. In the following, we present how this is done using sequencing information that is provided for locally generated packets.

We define $\mathcal{C}_N$ as the set of conflict-free packets originating from a node $N$. Concretely, it holds that $\forall m,n \in \mathcal{C}_N$ : $t_b(m) > t_b(n)$ iff $id_N(m) > id_N(n)$, which essentially means that all packets that are in $\mathcal{C}_N$ arrived at the sink in the same order as they were generated at node $N$. While the number of possible subsets $\mathcal{C}_N \subseteq \mathcal{P}$ is arbitrarily large, we propose to maximize the size of $\mathcal{C}_N$ by mapping the problem of finding $\mathcal{C}_N$ to solving the maximum independent set problem [21]. Therefore, we construct a graph in which each packet that originates from a node $N$ under investigation is represented by a vertex. While the maximum independent set problem is about finding the largest subset of independent elements, we connect two vertices with an edge, if the requirement $t_b(m) > t_b(n)$ iff $id_N(m) > id_N(n)$ is violated between the corresponding packets $m$ and $n$.

THEOREM 2. *We defined $\mathcal{W}$ as the set of all packets generated at a node $N$ in between which a packet $k$ must have arrived at $N$. For any packet $k$ that arrived at a node $N$ in between two consecutively, locally generated packets $m,n \in \mathcal{W}$, it holds that packet $k$ cannot have been reordered due to a parent change at $N$, if (1) $k$ can only have left $N$ to a single possible next hop, and (2) all packets $m$ that are part of $\mathcal{W}$ are also part of the set of conflict-free packets $\mathcal{C}_N$, thus $\forall m \in \mathcal{W} : m \in \mathcal{C}_N$.*

PROOF. Let us go back to the situation in Figure 2 and consider a node $N$ that generated packets $n_{K1}$, $n_{K2}$ and $n_L$. While $n_{K1}$ and $n_{K2}$ were forwarded to a node $L$, packet $n_K$ was forwarded to another parent $K$. We further assume that all three packets are in $\mathcal{W}$, *i.e.*, $\mathcal{W} := \{n_{K1}, n_{K2}, n_L\}$. In the follow-

ing, we will now discuss three possible cases concerning a forwarded packet $k$ arriving in between two packets of $\mathcal{W}$.

**Case 1:** Packet $k$ arrived in between $n_{K2}$ and $n_L$: As packets $n_{K2}$ and $n_L$ were forwarded to different next hops and therefore must carry distinct first-hop receivers, $k$ is already excluded from being a member of the set $\mathcal{R}$ of reliable packets due to ambiguities in the observable path.

**Case 2A:** Packet $k$ arrived in between $n_{K1}$ and $n_{K2}$, all packets $n_{K1}$, $n_{K2}$ and $n_L$ arrive at the sink in the same order as they arrived at $N$: Since all locally generated packets arrived in order, all three packets $n_1$, $n_K$ and $n_L$ are part of the conflict-free set of packets. In fact, packet $k$ has not been reordered due to a parent change at $N$. Analyzing packet $k$ continues until the sink $S$ is reached.

**Case 2B:** Packet $k$ arrived in between $n_{K1}$ and $n_{K2}$, packets arrive at the sink in the inconsistent order $n_{K1} \rightarrow n_L \rightarrow k \rightarrow n_{K2}$: Packet $k$ is no longer arriving at the sink in between its correct "anchor packets" $n_{K1}$ and $n_{K2}$, but in between $n_L$ and $n_{K2}$. It becomes apparent, that $n_L$ arriving at the sink before $n_{K2}$ is a conflict, which means that at most one of those two packets can still be in the set of conflict-free packets $\mathcal{C}_N$. Therefore, not all packets that are part of $\mathcal{W}$ are now also part of $\mathcal{C}_N$ anymore. In consequence, $k$ is no longer guaranteed to not have been reordered, and therefore not added to the set of "reliable" packets $\mathcal{R}$. □

### 6.3.3 Algorithm for Finding a Reliable Set

In the following presentation of the complete algorithm for constructing a set of "reliable" packets $\mathcal{R}$, we will now combine our previous findings. Here, we construct $\mathcal{R}$ by deciding for each packet $k \in \mathcal{P}$, if $k$ is also a member of $\mathcal{R}$. While the corresponding sets of conflict-free packets $\mathcal{C}_N$ is not part of Algorithm 2, those sets are determined before executing Algorithm 2 by solving the corresponding maximum independent set problem (see previous Section 6.3.2).

We start with firstly determining if packet $k$ arrived at the sink out of order w.r.t. packets originating from the same source (line 2). Then, we start with our analysis at the first-hop receiver $N^* := p(k)$. We can safely assume that $k$ must have arrived at $N^*$ later than it was generated, but earlier than it arrived at the sink, *i.e.*, $\tilde{t}_g(k) - \Delta^l(k) < t_a(N^*,k) < t_b(k)$. Thus, $k$ can have arrived at $N^*$ in between any of two locally generated packets $m,n \in \mathcal{W}$. Here, Equations (5) to (7) for determining $\mathcal{W}$ are reflected by lines 7 to 13 in Algorithm 2.

This analysis is a worst-case analysis, because we must prematurely stop for safety reasons, if one of the following conditions is met: First, we must stop, if $k$ might have been forwarded to not only one, but alternative other nodes. Evidence for this is found if we find evidence for packet loss in $\mathcal{W}$ (lines 14 to 15), as well as when not all packets in $\mathcal{W}$ were forwarded to the same first-hop receiver (lines 16 to 17). Second, we must also stop, if we find evidence that a packet $w \in \mathcal{W}$ arrived at the sink out of order w.r.t. another locally generated packet $v$ (line 18). In the good case, we continue analyzing $k$ in the context of the next hop, *i.e.*, $N^* := p(u)$ (line 19). Packet $k$ is added to $\mathcal{R}$, if we can safely trace the packet until the sink is reached (line 21). At the end of this algorithm, we constructed a set $\mathcal{R} \subseteq \mathcal{P}$ so that packet correlation using anchor packets is safe for all packets $k \in \mathcal{R}$.

**Algorithm 2:** Algorithm for deciding if a packet $k \in \mathcal{P}$ is also a member of the set of "reliable" packets $\mathcal{R}$

**input**: Packet $k$

1 **begin**
2  **if** $k \notin \mathcal{C}_{o(k)}$ **then**
3  return ;
4  **end**
5  $N^* \longleftarrow p(k)$ ;
6  **while** $N^* \not\equiv S$ **do**
7  $u \leftarrow \arg\max_x \tilde{t}_g(x) + \Delta^u(x)$ for all $x : o(x) \equiv N^*$
8  $\wedge\, \tilde{t}_g(x) + \Delta^u(x) < \tilde{t}_g(k) - \Delta^l(k)$ ;
9  $v \leftarrow \arg\min_x \tilde{t}_g(x) - \Delta^l(x)$ for all $x : o(x) \equiv N^*$
10  $\wedge\, \tilde{t}_g(x) - \Delta^l(x) > t_b(k)$ ;
11  **if** $u \equiv \{\}$ ***or*** $v \equiv \{\}$ **then** break ;
12  $\mathcal{W} \longleftarrow \{w \mid o(w) \equiv N^*$
13  $\wedge id_{N^*}(u) \leq id_{N^*}(w) \leq id_{N^*}(v)\}$
14  $\mathcal{I} \longleftarrow \{id_{N^*}(w) \mid w \in \mathcal{W}\}$ ;
15  **if** $|\mathcal{I}| \not\equiv \max\mathcal{I} - \min\mathcal{I} + 1$ **then** break ;
16  $\mathcal{P} \longleftarrow \{p(w) \mid w \in \mathcal{W}\}$ ;
17  **if** $|\mathcal{P}| > 1$ **then** break ;
18  **if** $\exists m \in \mathcal{W} : m \notin \mathcal{C}_{N^*}$ **then** break ;
19  $N^* \longleftarrow p(u)$ ;
20  **end**
21  **if** $N^* \equiv S$ **then** $\mathcal{R} \longleftarrow \mathcal{R} \cup \{k\}$ ;
22 **end**

## 6.4 Forward and Backward Reasoning

Initially set bounds on packet arrival times $t_a(N,k)$ are often pessimistic. As a first improvement, a packet can apparently not have arrived earlier than it was generated:

$$\forall N \in \mathcal{N}_k : t_a^l(N,k) := \max\left(\tilde{t}_g(k) - \Delta^l(k), t_a^l(N,k)\right)$$

Likewise, the arrival time at the sink $t_b(k)$ of a packet $k$ marks the largest upper bound:

$$\forall N \in \mathcal{N}_k : t_a^u(N,k) := \min\left(t_b(k), t_a^u(N,k)\right)$$

### 6.4.1 Forward and Backward Queue Traversal

For achieving further improvements of the arrival time of a packet $k$ at a node $N$, the timing information of packet $k$ can be correlated with information of other packets that also arrived at node $N$.

Given a packet $k$ that was arriving at the queue of a node $N$, thus $N \in \mathcal{N}_k$, the following equations state, that the observable order of arrival at node $N$ must also be reflected by the upper and lower bounds on the arrival time $t_a(N,k)$:

$$\forall u,N \in \mathcal{N}_u, t_b(u) < t_b(k) : t_a^l(N,k) := \max\left(t_a^l(N,k), t_a^l(N,u)\right)$$

$$\forall v,N \in \mathcal{N}_v, t_b(v) > t_b(k) : t_a^u(N,k) := \min\left(t_a^u(N,k), t_a^u(N,v)\right)$$

As we are restricting us to packets $k \in \mathcal{R}$, the order of packet arrivals at the packet queue of node $N$ matches with the observed order of packet arrivals at the sink. Similarly, bounds can also be improved by correlating information from different nodes that a particular packet visited.

## 7 Multi-Protocol Testbed Evaluation

In this section, we validate and evaluate our implementation of the MNT algorithm based on experiments with two well-known state-of-the-art communication stacks used for data collection, namely CTP [10] and Dozer [4]. After describing the experimental setup, *i.e.*, the protocol selection, the packet time-stamping scheme used, and the infrastructure used for extracting ground-truth information, results obtained are validated by comparing them to ground truth. The performance of the implementation used is evaluated in terms of the fraction of packets for which information reconstruction succeeded, the time passed until results are available, and the accuracy of calculated arrival time bounds.

### 7.1 Experimental Setup

The results presented originate from three test environments: CTP Noe is executed on top of the standard low-power listening (LPL) MAC that is provided with version 2.1 of TinyOS. Those tests are carried out on up to 92 Tmote Sky (TI MSP430, CC2420 radio) nodes that are part of the TWIST testbed [11]. Twenty-five TinyNode184 (TI MSP430, Semtech SX1211 radio) nodes of the also public FlockLab testbed [19] are used for measurements that involve Dozer. For larger scaling tests on a 100-hop line topology, we are furthermore running experiments with an implementation of CTP [7] in the Castalia/OMNeT++ [3] network simulator.

The purpose of the protocol selection and configuration used is to cover the following design aspect of routing protocols: While nodes are configured to turn on their duty-cycled radios every 62 ms and 250 ms, respectively, when running CTP Noe on top of the standard low-power listening [26] MAC found in TinyOS 2.1, communication only takes place every 15 or 30 seconds, respectively, when running Dozer with a corresponding set of parameters. While the design space for routing protocols is definitely broader, *e.g.*, includes energy considerations, the tradeoff between reactivity and latency is of highest relevance in the context of this work. All protocols have only been modified to transmit per-packet information assumed by our formal system model. This requires only changes on the application-level, other layers remain untouched.

#### 7.1.1 Packet Time-stamping

Providing simple integration, we decided to obtain packet generation timestamps using elapsed time on arrival [16]. We define $t_s(k)$ as the accumulated sojourn time that a packet $k$ spent within the network. Ideally, the packet generation time $t_g(k)$ of a packet $k$ is retrieved by subtracting the packet sojourn time $t_s(k)$ from the arrival time of the packet at the sink, thus $t_g(k) := t_b(k) - t_s(k)$. Here, it is assumed that $t_b(k)$ and $t_s(k)$ are measured on perfect clocks.

In the context of real clocks, we cannot measure $t_s(k)$, but the estimated packet sojourn time $\tilde{t}_s(k)$ that includes artifacts caused by measuring time on clocks with a low resolution and drift. After being initialized with $\tilde{t}_s(k) := 0$ on packet generation, this additional packet header is successively updated while a packet travels through the network. The inaccuracy of the resulting packet generation time estimate $\tilde{t}_g(k) := t_b(k) - \tilde{t}_s(k)$ of a packet $k$ is as follows:

$$\Delta^u(k) := \frac{\tilde{t}_s(k)}{1+\hat{\rho}}, \quad \Delta^l(k) := \frac{\tilde{t}_s(k) + |\mathcal{N}_k| \cdot \hat{t}_u}{1-\hat{\rho}}$$

Here, we assume a bounded clock drift $\rho \in [-\hat{\rho}; \hat{\rho}]$, *e.g.*, $\hat{\rho} := \pm 60$ ppm, and a clock resolution of $\hat{t}_u$, *e.g.*, $\hat{t}_u := 1$ sec. As an error of $[0, +\hat{t}_u)$ can be introduced by each separate measurement, we must multiply $\hat{t}_u$ with the length $|\mathcal{N}_k|$ of the packet path $\mathcal{N}_k$.

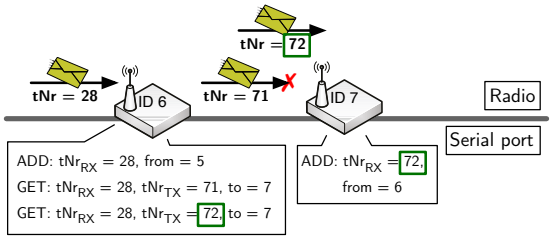



**Figure 3. Extraction of ground truth.** For our validation tests, we instrumented code to make the travel of individual packets observable. This information is not transferred in-band, but over the serial port of the sensor node. Sensor nodes firstly generate a log message when a packet is added to the send queue (ADD), and secondly immediately before a packet is handed over to the radio for transmission (GET). The value of the local transmission counter *tNr* is incremented before any transmission attempt, and logged on both sender and receiver sides.

#### 7.1.2 Extraction of Ground Truth

For the validation of the MNT algorithm, we are interested in the ground truth w.r.t. the path, the per-hop arrival order, and the per-hop arrival time of individual packets. Therefore, we instrumented existing protocol code for outputting this information over the serial port of the sensor node. An observer device, *e.g.*, a more powerful, networked PC, is connected to the serial port of any sensor node, received messages are logged to a file. Compared to sending ground truth information in-band over the radio, transmitting ground truth information over the serial port is very reliable, does not influence the packet stream under investigation, and also scales well for larger networks. Additionally, messages can be timestamped using the accurate, synchronized clock of the observer device. Ground truth is eventually reconstructed by correlating information from individual logs.

In more detail, sensor nodes log the following two events: Firstly, a log message is generated when a packet is added to the packet queue (ADD). This can be triggered by the application generating a new message, by the reception of a forwarded packet, or when a packet is copied from a secondary node storage, *e.g.*, a SD card. Secondly, a log message is also generated immediately before every attempt of transmitting a message over the radio (GET). Therefore, there can be multiple occurrences of GET events for a single packet. The occurrences of both events are timestamped on the local node clocks, the time difference between corresponding ADD and GET events yields the local packet sojourn time of a packet on a particular node. The sink is simply forwarding

| | N | D | IPI | H | PC | DY | Packets | | | | Proc. Delay | | Uncertainty | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Received | Reliable | Full path | Correct | $p_{0.9}$ | $p_{0.98}$ | $p_{0.9}$ | $p_{0.98}$ |
| *CTP Noe/LPL* | | | | | | | | | | | | | | |
| A) | 92 | 10 h | 15 s | 4 | 274 | 99.0% | 217,078 | 99.0% | 98.9% | 100.0% | 17 s | 20 s | <1 s | <1 s |
| B) | 85 | 9 h | 30 s | 3 | 122 | 98.3% | 88,541 | 98.9% | 98.9% | 100.0% | 32 s | 40 s | <1 s | 1 s |
| C) | 91 | 9 h | 120 s | 4 | 416 | 99.2% | 54,143 | 98.4% | 98.2% | 100.0% | 62 s | 75 s | 1 s | 1 s |
| *CTP (Simulation)* | | | | | | | | | | | | | | |
| D) | 100 | 5 h | 30 s | 100 | 0 | 99.9% | 60,150 | 96.5% | 93.2% | 100.0% | 30 s | 31 s | <1 s | <1 s |
| E) | 100 | 20 h | 120 s | 100 | 0 | 99.9% | 60,112 | 97.0% | 94.3% | 100.0% | 120 s | 121 s | <1 s | <1 s |
| *Dozer* | | | | | | | | | | | | | | |
| F) | 25 | 12 h | 15 s | 3 | 548 | 99.8% | 48,321 | 91.3% | 91.2% | 100.0% | 537 s | 1363 s | 15 s | 90 s |
| G) | 25 | 16 h | 30 s | 4 | 193 | 99.9% | 35,220 | 92.4% | 92.3% | 100.0% | 1024 s | 2201 s | 30 s | 120 s |
| H) | 25 | 24 h | 120 s | 4 | 196 | 99.9% | 18,216 | 98.5% | 98.4% | 100.0% | 120 s | 180 s | 84 s | 114 s |
| I) | 10 | 60 h | *120 s | 3 | 37 | 99.5% | 84,563 | 97.7% | 97.7% | 100.0% | 122 s | 212 s | 74 s | 118 s |

**Table 2. Validation and evaluation based on testbed experiments and simulation.** Shown are the number of sensor nodes (N), the duration of the test (D), the inter-packet interval (IPI), the height of the data collection tree in hops (H), the number of observed parent changes (PC), and the data yield (DY). The fractions of packets that were part of the "reliable" set and the fractions of packets that could be fully reconstructed up to the sink are specified based on the number of packets received. The distribution of the processing delay is given by the 90th and 98th percentiles. Similar, the per-hop arrival time bounds uncertainty, *i.e.*, the difference of upper and lower bounds, is also given using percentiles. (*) Deployment configuration in which five packets are generated every 120 s.

any received packet to the serial port.

Packet duplications can lead to multiple copies of a single packet being simultaneously traveling through the network. For being able to distinguish between multiple instances of a single packet, each packet transmission is made uniquely detectable by adding a transmission counter. Each sensor node is individually counting its local transmission attempts, the current counter value is added to each packet just before the packet is passed over to the radio. Sending the counter value in-band allows to log the respective value at both sides and finally to match the traces of the sending and the receiving node. An illustrating example of this mechanism is shown in Figure 3.

## 7.2 Validation and Evaluation Results

Validation and evaluation results from nine different test configurations are shown in Table 2. Results from simulation are in line with results obtained from experiments on real hardware. Varying test durations and network sizes are a result of varying availabilities of testbed resources. Apart from available time slots, tests are also limited to sensor nodes for which serial logging turned out to be successful in a pre-test.

For evaluating the sensitivity of the MNT algorithm w.r.t. the inter-packet interval (IPI), tests are repeated using varying packet generation rates. While using a periodic packet generation scheme allows us to make the sensitivity w.r.t. the IPI more visible, the MNT algorithm neither requires packet generation to be periodic, nor asks for all sensor nodes being using the same scheme (see Section 5). For being able to evaluate the performance for both very high and artificially lowered data yields, nodes are programmed to only generate new packets when there is free space in the local send queue. While this avoids packets being dropped due to queue overflows, this can result in the effective IPI being lower than the configured IPI.

### *7.2.1 Evaluation Methodology*

The number of parent changes is computed from collected ground truth information. The number of missing packets, and thus the data yield, is calculated from included packet sequencing information $id_N(k)$. Here, a packet is not only missing if it was not received at the sink, but also when ground truth information is missing. As we generally find serial port communication very reliable, the amount of packets missing due to missing ground truth information is not significant.

Reconstructed information of a packet is only counted as correct if all three reconstructed components, *i.e.*, packet path, per-hop arrival order and per-hop arrival times, are correct. The reconstructed path $\mathcal{N}_k$ of a packet $k$ is correct if found nodes including their order matches with ground truth. The reconstructed path can prematurely end if packet $k$ could not be fully reconstructed, but cannot contain additional elements or gaps. Reconstructed arrival order information is correct if the order of the queue index $qid_N(k)$ matches with ground truth for any node $N \in \mathcal{N}_k$. Lastly, extracted arrival time bounds are correct if it holds for all packets $k \in \mathcal{R}$ that the real arrival time $t_a(N,k)$ is within the reconstructed bounds, *i.e.*, $\forall N \in \mathcal{N}_k : t_a^l(N,k) \leq t_a(N,k) \leq t_a^u(N,k)$. As both reconstructed bounds and ground truth are measured on clocks of varying resolution and drift, this comparison allows for a bounded measurement error.

As presented in Section 6, the MNT algorithm also requires information of packets that arrived at the sink later than a packet $k$ under investigation. We define the processing delay as the time distance between the arrival time $t_b(k)$ of packet $k$ at the sink and the arrival time $t_b(v)$ of packet $v$ at the sink, *i.e.*, $t_b(v) - t_b(l)$. Here, $v$ is the latest arriving packet that is required for the analysis of packet $k$.

The uncertainty of reconstructed per-hop arrival time bounds is defined as the difference between upper and lower bounds, *i.e.*, $t_a^u(N,k) - t_a^l(N,k)$.

### *7.2.2 Discussion*

This section details on how the IPI, the reactivity of the communication protocol used, the length of the routing paths, the time packets spent in the network and the amount

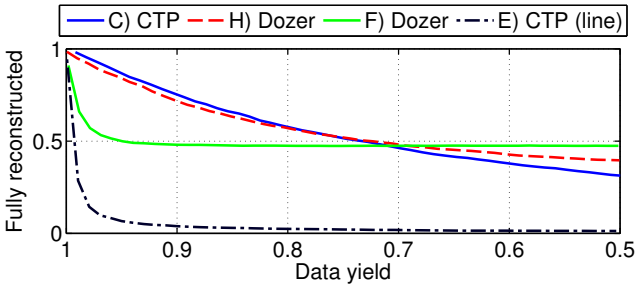



**Figure 4. Sensitivity of the reconstruction performance to a decreasing data yield.** The fraction of received packets for which information can be safely reconstructed converges towards the fraction of packets that originate from single-hop neighbors of the sink.

of lost packets influence the performance of the MNT algorithm. After firstly discussing the fraction of reconstructed packets, we will also detail on the processing delay and the uncertainty of reconstructed per-hop arrival time bounds.

While the core principle of the MNT algorithm is to reconstruct information from created correspondences between multiple packets of different sources, an integral metric for understanding the reconstruction performance is the number of correspondences that are needed for being able to safely reconstruct the information of a packet. The number of packets that need to be considered during a worst-case path analysis (see Section 6.3.1) grows with the length of the routing path and the time a packet spent in the network. Similarly, the number of packets whose reconstruction relies on a particular packet also grows for longer routing paths and larger packet sojourn times. In consequence, the number of packets whose reconstruction might be affected by a particular parent change or a lost packet is also increasing.

Recent deployment reports, *e.g.*, [5, 15], confirm that a data yield of $\geq 99.5\%$ is achievable even in very challenging environments. A problem within the network must not necessarily be reflected by a large portion of packets being lost, but can also cause packets arriving at the sink with a larger delay. For instance, the Dozer implementation used in the PermaSense project does not deliberately drop packets, but retransmits each single packet until its reception was eventually acknowledged by the next hop.

Nevertheless, artificially lowering the data yield down to 50% allows us to study the fundamental limits of the MNT algorithm. Starting with the original traces obtained from tests C), E), F) and H), the data yield is consecutively lowered by randomly removing packets. The resulting reconstruction performance is shown in Figure 4. Here, we see the fraction of fully reconstructed and received packets converging towards a stable value which is the fraction of packets originating from one-hop neighbors of the sink. For example, only 1% of the packets in test E) originate from the single one-hop neighbor in the simulated 100-hop line topology. The decrease is almost linear for cases C) and H), the curves for the tests E) and F) show a steep decay at the beginning of the curve. Large routing paths in test E) and large packet sojourn times of up to four hours in test F) result in large numbers of correspondences needed for deciding if information can be reconstructed safely. In consequence, already a small number of lost packets can cause the information needed for the reconstruction of multiple other packets being lost in those extreme situations.

Regarding the time needed until all related packets also arrived at the sink, the results of tests A) to E) show a correlation between the processing delay and the chosen IPI. While almost all packets reached the sink as fast as possible in those five tests, the results presented for tests F) and G) include the effects of a fraction of packets staying in the network for several hours.

The uncertainty of reconstructed per-hop arrival time bounds is upper bound by the largest IPI used along the routing path. Uncertainties are significantly reduced when a reactive communication stack, *i.e.*, CTP Noe/LPL, is used.

Overall, the MNT algorithm has proven to achieve high reconstruction rates $\geq 91.2\%$ in various configurations based on well-known CTP and Dozer protocols. During normal operation without congestion, information can be reconstructed quickly after a packet has been received at the sink. The uncertainty of reconstructed per-hop arrival time bounds is not affected by large packet sojourn times. Comparison with ground truth showed reconstructed information to be correct in all cases.

# 8 Making Real Network Dynamics Visible

This case study presents the application of the MNT algorithm to large data sets that originate from three productive real-world deployments of the PermaSense project. After the initial deployment of the first system in 2008, the principle operation of the Dozer protocol used and the definition of packet application headers transmitted have not been modified ever since. This enables a coherent analysis of the complete data sets using the MNT algorithm.

Section 8.1 presents the reconstruction of information required by the MNT algorithm. The purpose of this step is to map the output of the specific system implementation to the inputs of the implementation-independent, generic system model (see Section 5). The accuracy of this transformation step is evaluated in testbed experiments. Results obtained from applying the MNT algorithm to the resulting traces are presented and discussed in Section 8.3. Section 8.4 describes the targeted usage scenario of reconstructed data.

## 8.1 Data Preparation Methodology

Input data required by the MNT algorithm is not implicitly included in PermaSense data sets and must therefore be reconstructed from other information during a multi-stage pre-processing step, see Figure 5:

- **Packet index** $id_N(k)$: Sequencing information provided by a sequence number transmitted that is reset to zero every 20 days in average must be converted to a monotonically increasing packet index $id_N(k)$. The algorithm used is able to address both intended, *i.e.*, the maximum counter value is reached, and unintended resets, *e.g.*, a node failure, of the sequence number.
- **SD card store time**: Most sensor nodes used in the PermaSense project are equipped with an SD card that buffers locally generated packets when the node is dis-

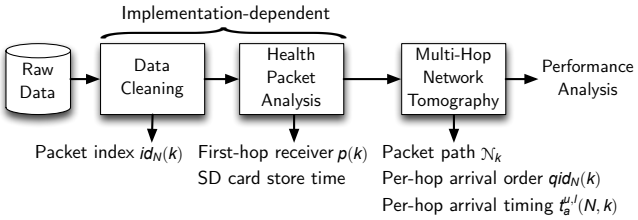



**Figure 5. Implementation-specific data preparation.** In order to conform with the inputs required by multi-hop network tomography, historic data from PermaSense deployments must be prepared using algorithms that are specific to the PermaSense system implementation.

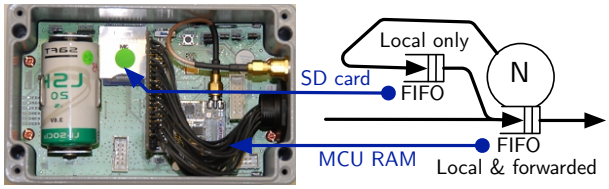



**Figure 6. PermaSense sensor node with two packet queues.** Locally generated packets are looped through a second queue that is situated on a SD memory card. The packet generation time is no longer a valid proxy for the send queue arrival time, the MNT algorithm thus requires the duration for which a packet was stored on the SD card to be known.

connected. Independent of the connection state, all locally generated packets are looped through this extra queue, see Figure 6. Received packet sequencing and periodically sampled queue size information are used for replaying queue operations and eventually reconstructing the SD card store time of individual packets.

- **First-hop receiver** $p(k)$: Nodes do not transmit the first-hop receiver of individual packets, but periodically sample the address of the current parent node. This requires the reconstruction of the time when a locally generated packet left the send queue, and thus was successfully transmitted over the radio. The estimation of this information is also based on known properties of the queue implementations used.

The accuracy of the deployment data preparation test is verified in two testbed experiments of 110 hours duration in total. Similar to a deployed setting, these experiments involve six sensor nodes with attached SD cards, four sensor nodes that can only buffer messages in the RAM, and one sink node. Sensor nodes generate five packets every 2 minutes. Queue size counters and parent information are transmitted with every fifth packet. In contrast to the deployed setting, ground truth SD card store time and first-hop receiver are also transmitted with every packet. For forcing sensor nodes to buffer messages, the sink node is eight times switched off for a duration of 3 hours each. From 157,645 unique packets received, 99.4% of the packets pass the first stage of assigning a unique packet index $id_N(k)$. SD card store time and first-hop receiver information are reconstructed for 154,870 packets, remaining packets cannot be reconstructed due to missing context at the end of the trace. Reconstructed first-hop receiver information is correct in 99.96% of the cases. Subject to actually buffered packets only, the absolute mean error of the estimated SD card store time is 50 s. This is to be expected given that queue sizes are only sampled every two minutes. Please notice that the logical ordering in which packets arrive is unaffected from this uncertainty, and that this uncertainty is specific to historic data originating from PermaSense deployments only.

| | **Matterhorn** | **Jungfraujoch** | **Dirruhorn** |
|---|---|---|---|
| *Deployment characteristics* | | | |
| Network size | 26 nodes | 15-28 nodes | 28 nodes |
| Years of operation | 3.3 | 3.1 | 1.6 |
| *Received packets* | | | |
| Unique | 78,023,336 | 46,101,139 | 19,500,270 |
| Duplicates | 548,796 | 435,542 | 132,027 |
| *Packets after Data Cleaning* | | | |
| Unique | 99.0% | 98.5% | 99.3% |
| *Packets after Health Packet Analysis* | | | |
| Total unique | 98.4% | 86.2% | 98.3% |
| Per node, min | 97.0% | 33.6% | 94.0% |
| Per node, max | 99.9% | 99.9% | 99.4% |
| *Input of Multi-Hop Network Tomography* | | | |
| Total | 76,869,053 | 39,814,741 | 19,207,962 |
| % of unique packets | 98.5% | 86.4% | 98.5% |

**Table 3. Results of PermaSense-specific data preparation.** Large portions of the input data have been reconstructed and are therefore ready for the multi-hop network tomography. Results of the Jungfraujoch deployment vary due to gaps in the traces used.

## 8.2 Multi-Year Deployment Data Preparation

Multi-year deployment data originating from different hardware and software releases is not perfect, *e.g.*, still contains artifacts of problems that have been fixed over time. Therefore, the analysis of such data requires highest attention and care. Results of each intermediate step are separately verified by automated and manual sanity checks. The results of the implementation-specific data preparation for more than 140 million packets that originate from three sensor network deployments are shown in Table 3. More than 98.5% of the data from Matterhorn and Dirruhorn deployments have been reconstructed and are therefore ready for multi-hop network tomography. The results for the Jungfraujoch deployment originate from truncated traces that are possibly caused by unintended manual deletion in the data repository. While the ends of the three incomplete traces show ca. 100,000 buffered packets each after a long phase of no connectivity, corresponding health packets for reconstructing how queues were flushed are missing.

## 8.3 Multi-Deployment Network Tomography

The execution of the MNT algorithm requires the combined processing of the traces of all sensor nodes. To deal with the amounts of data found in this case study, data sets are split into week-long slices with approximately 300,000 packets per slice. Each slice contains packets that were generated during the corresponding week. Depending on the time packets spent in the network also packets of following weeks must be loaded for providing required context to the worst-case path analysis that is part of the MNT algorithm

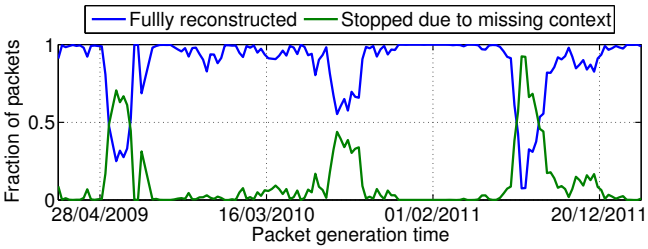



**Figure 7. Weekly reconstruction performance of Jungfraujoch data set.** Lower performance is caused by gaps in the data repository that were unintentionally introduced during the handling of data received.

| | Matterhorn | Jungfraujoch | Dirruhorn |
|---|---|---|---|
| *Total packets including packet duplicates* | | | |
| Input | 76,869,053 | 39,814,741 | 19,207,962 |
| Reliable | 99.5% | 93.4% | 97.8% |
| Full path | 99.5% | 91.5% | 93.7% |
| *Fully reconstructed packets per week-long slice* | | | |
| Min | 92.5% | 7.5% | 67.8% |
| Max | 100.0% | 100.0% | 100.0% |

**Table 4. Results of multi-hop network tomography.** The MNT algorithm is able to trace both original packets and independently traveling packet duplicates. Since it is not known at which hop a packet was actually duplicated, all packet instances are assumed to originate from the source. Duplicates are generally flagged and removed if this inaccuracy might harm the result of a particular analysis.

(see Section 6.3.1). For example, up to 10 times more data must be loaded when processing packets that were buffered in the network for multiple months at the Jungfraujoch site.

Multi-hop network tomography results are presented in Table 4 and Figure 7. More than 91.5% of the packets passed to the MNT algorithm were fully reconstructed. Compared to the amount of total received packets from the network before the data preparation, this accounts to 97.3%, 78.3%, and 91.6% of packets being fully reconstructed.

Lower results for individual weeks are caused by the MNT algorithm not being able to reconstruct all packets due to missing context, *i.e.*, human errors while handling data causing significant amounts of packets to be missing in the data repository. This is especially visible in Figure 7 which shows the weekly amounts of fully reconstructed packets and raised errors due to missing context. The number of errors virtually mirrors the number of packets that could not be reconstructed. Other potential problems, *e.g.*, sporadically lost packets and packet reordering, are only causing insignificant amounts of packets to be not reconstructable.

## 8.4 Performance Analysis Inside the Network

Multiple phenomena contribute to the delay that a packet experiences inside the network. Apart from the baseline set by the configuration of the protocol used, the packet delay is also influenced by intermittent routing problems, *e.g.*, problems with the wireless channel or no free buffer capacity at the next hop towards the sink. For example, understanding that large amounts of packets are delayed due to missing fairness within the network could be a valuable indicator for improving the network beyond the scope of protocol

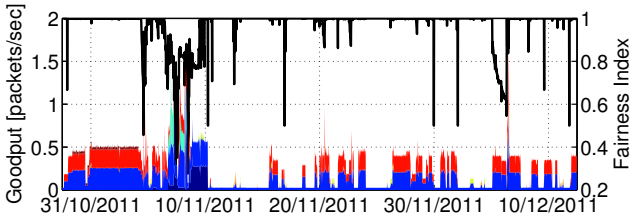



**Figure 8. Goodput (left-hand side axis) and fairness index (right-hand side axis) at node position 3 of the Dirruhorn deployment.** This node is on average two hops away from the sink. It generates own data and receives packets from seven other nodes during the time period shown. Different packet sources are shown using distinct colors. The allocation of slots in the packet queue is no longer fair when the goodput is higher than 0.5 packets per second.

self-recovery mechanisms, *e.g.*, by modifying statically set protocol parameters.

Combining reconstructed packet path and per-hop arrival times for deriving bounds on per-hop packet sojourn times is probably the most promising application of reconstructed data. Supported by the fact that the uncertainty of reconstructed arrival times is not affected by large packet sojourn times, our vision is to infer detailed network state, *e.g.*, links suffering from bad connectivity, from reconstructed per-hop packet sojourn times.

While the automated processing and interpretation of reconstructed information is outside the scope of this paper, the following example highlights how reconstructed information can be used for analyzing fairness within the network. Following the definition of Jain *et al.* [13], we calculate the Fairness Index that is defined as the fraction of users being treated fair. An allocation is fair, if all users receive a share that is fair in terms of their own demand and the demand of other users. Results obtained for 20 minutes long slices are shown in Figure 8. While the demand is defined as the number of packets that are currently buffered in both queues of a sensor node, the allocated share is defined as the number of packets that a particular child node transmitted during a slice of 20 minutes length.

In our analysis, we find the sink node to always be fair to its children. This is to be expected as the sink node is not duty-cycled and also does not suffer from queue size constraints when data is immediately forwarded to the base station PC. However, fairness is no longer given at intermediate nodes when the amount of received packets is higher than 0.5 packets per second over a longer time period. It is to investigate if this behavior needs to be improved, *e.g.*, by modifying the current queue allocation scheme.

# 9 Broader Applicability and Limitations

With tree-based routing protocols being very popular in real-world deployments, the MAC-independent MNT algorithm can potentially be used in many scenarios. Still, the assumption of all nodes generating data might not fit to some applications. While sensor nodes that only forward packets could be modified to cooperate, *i.e.*, modify passing first-hop receiver information to hide themselves, the existence of sole forwarders will unavoidably introduce inaccuracies. Addi-

tionally, the assumption of routing paths being rather stable might render the MNT algorithm not applicable for systems in which path changes occur very frequent, *e.g.*, because of nodes selecting the next hop in a round-robin fashion.

## 10 Conclusions

This paper presented multi-hop network tomography (MNT), a novel, non-intrusive algorithm for the reconstruction of the path, the per-hop arrival order and the per-hop arrival time of individual packets. The results of extensive testbed runs of two state-of-the-art data collection protocols, *i.e.*, CTP and Dozer, verified that information can be reconstructed with a great confidence. The application in a deployment context has been proven to be feasible in a case study that involved more than 140 million packets from three real-world sensor network deployments.

**Acknowledgements.** We want to thank our shepherd Kamin Whitehouse and the anonymous reviewers for their valuable feedback on earlier versions of this paper. The work presented was supported by NCCR-MICS under SNSF grant #5005-67322, Nano-Tera.ch, and the HFSJG.

## 11 References

[1] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. The hitchhiker's guide to successful wireless sensor network deployments. In *6th ACM Conf. on Embedded Networked Sensor Systems (SenSys '08)*, 2008.

[2] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel. PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes. In *7th Int'l Conf. on Information Processing in Sensor Networks (IPSN '09)*, 2009.

[3] A. Boulis et al. Castalia: A simulator for wireless sensor networks. http://castalia.npc.nicta.com.au/.

[4] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *6th Int'l Conf. on Information Processing in Sensor Networks (IPSN '07)*, 2007.

[5] 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 *8th Int'l Conf. on Information Processing in Sensor Networks (IPSN '09)*, 2009.

[6] A. Coates, A. Hero III, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazine*, 19(3):47–65, 2002.

[7] U. Colesanti and S. Santini. The Collection Tree Protocol for the Castalia wireless sensor networks simulator. Technical Report 729, Department of Computer Science, ETH Zurich, 2011.

[8] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *10th ACM Conf. on Embedded Networked Sensor Systems (SenSys '12)*, 2012.

[9] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *ACM SIGPLAN 2003 Conf. on Programming Language Design and Implementation (PLDI '03)*, 2003.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *7th ACM Conf. on Embedded Networked Sensor Systems (SenSys '09)*, 2009.

[11] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *2nd Int'l Workshop on Multi-hop Ad Hoc Networks (REALMAN '06)*, 2006.

[12] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse. The hitchhiker's guide to successful residential sensing deployments. In *9th ACM Conf. on Embedded Networked Sensor Systems (SenSys '11)*, 2011.

[13] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report 301, DEC, 1984.

[14] M. Keller, L. Thiele, and J. Beutel. Reconstruction of the correct temporal order of sensor network data. In *10th Int'l Conf. on Information Processing in Sensor Networks (IPSN '11)*, 2011.

[15] M. Keller, M. Woehrle, R. Lim, J. Beutel, and L. Thiele. Comparative performance analysis of the PermaDozer protocol in diverse deployments. In *6th IEEE Int'l Workshop on Practical Issues in Building Sensor Network Applications (SenseApp '11)*, 2011.

[16] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler. Elapsed time on arrival: A simple and versatile primitive for canonical time synchronisation services. *Int'l Journal of Ad Hoc and Ubiquitous Computing*, 1(4):239–251, 2006.

[17] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *20th Int'l Parallel and Distributed Processing Symposium (IPDPS '06)*, 2006.

[18] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, 2003.

[19] R. Lim, C. Walser, F. Ferrari, M. Zimmerling, and J. Beutel. Distributed and synchronized measurements with FlockLab. In *10th ACM Conf. on Embedded Networked Sensor Systems (SenSys '12)*, 2012.

[20] Y. Liu, K. Liu, and M. Li. Passive diagnosis for wireless sensor networks. *IEEE/ACM Trans. on Networking*, 18(4):1132–1144, 2010.

[21] M. Luby. A simple parallel algorithm for the maximal independent set problem. In *17th Annual ACM Symposium on Theory of Computing (STOC '85)*, 1985.

[22] L. Mo, Y. He, Y. Liu, J. Zhao, S.-J. Tang, X.-Y. Li, and G. Dai. Canopy closure estimates with GreenOrbs: sustainable sensing in the forest. In *7th ACM Conf. on Embedded Networked Sensor Systems (SenSys '09)*, 2009.

[23] L. Mottola and G. Picco. MUSTER: adaptive energy-aware multi-sink routing in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 10(12):1694–1709, 2011.

[24] L. Mottola, G. P. Picco, M. Ceriotti, c. Gună, and A. L. Murphy. Not all wireless sensor networks are created equal: A comparative study on tunnels. *ACM Trans. Sen. Netw.*, 7:15:1–15:33, September 2010.

[25] H. Nguyen and P. Thiran. Using end-to-end data to infer lossy links in sensor networks. In *25th Int'l Conf. on Computer Communications (INFOCOM '06)*, 2006.

[26] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *2nd Int'l Conf. on Embedded Networked Sensor Systems (SenSys '04)*, 2004.

[27] M. Rabbat, R. Nowak, and M. Coates. Multiple source, multiple destination network tomography. In *23th Int'l Conf. on Computer Communications (INFOCOM '04)*, volume 3, 2004.

[28] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *3rd ACM Conf. on Embedded Networked Sensor Systems (SenSys '05)*, 2005.

[29] Y. Sankarasubramaniam, I. Akyildiz, and S. McLaughlin. Energy efficiency based packet size optimization in wireless sensor networks. In *1st Int'l Workshop on Sensor Network Protocols and Applications (SNPA '03)*, 2003.

[30] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *ACM Trans. Sen. Netw.*, 6:16:1–16:49, 2010.

[31] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *7th Symp. on Operating Systems Design and Implementation (OSDI '06)*, 2006.

[32] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, 2003.

[33] J. Yang, M. L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: a comprehensive source-level debugger for wireless sensor networks. In *5th ACM Conf. on Embedded Networked Sensor Systems (SenSys '07)*, 2007.

[34] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, 2003.