# The Problem Bit

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

***Abstract*—Health monitoring is an integral service for the long-term operation of wireless sensor networks. Instead of actively adding probe or status traffic, recently proposed passive health monitoring systems infer system health solely from existing application traffic. Our results from extensive testbed experiments prove that the detection rate of an exemplary monitoring application is significantly improved, *i.e.*, by up to 28% to overall 94%, when a passive health estimation method is complemented with a minimally active component.**

**This paper presents *Hybrid Monitoring*, a novel health monitoring system that combines the advantages of passive and active health monitoring systems. Results from the analysis of passively reconstructed, inexact per-hop timing information are significantly improved when one bit of extra information is added to every packet. The resulting system is able to estimate a significant health metric, *i.e.*, the number of occurred failure events on a node, with a high confidence. The temporal resolution of the estimated signal is equal to the packet sampling interval, the performance of our system is not affected by large packet delays, *e.g.*, when a sensor node is disconnected for a long time.**

## I. Introduction

Wireless sensor networks (WSNs) are well-suited for long-term monitoring applications. Their ability to run for multiple years solely from battery power allows WSNs to be placed into environments that offer minimal infrastructure only. The use of wireless communication enables a non-intrusive integration into an environment. Exemplary applications are ecosystem monitoring [22], the monitoring of heritage buildings [3], and the monitoring of geophysical processes in hostile environments [11].

Looking at recent projects, we can currently observe a continuous growth in network size [22] and in the usage horizon of WSN deployments. Furthermore, anticipated future applications of unprecedented societal and economical interest, *e.g.*, the monitoring of natural hazards, will require those networks to fulfill highest reliability requirements. Networks of growing size and importance can no longer be monitored manually, but require automated systems that ideally warn a network operator even before a yet undetected, lingering misbehavior [12] would eventually result in a fatal incident, *e.g.*, a node completely stopping its service.

Sympathy [24] is a health monitoring system in which nodes are actively sending local status, *e.g.*, counter values and routing information, to the sink. The amount of monitoring traffic added is reduced when in-network aggregation is used [26], [18]. LD2 [21] reduces traffic towards the sink by running a local diagnosis process inside the network. In contrast to those active systems, PAD [19] introduces only a negligible overhead in terms of communication and computation on resource-constraint devices. While PAD must resort to analyzing unmodified application traffic only, this limits the detection accuracy and coverage of the approach. For instance, given an increase in observed packet delays, the connectivity history of a node must be known in order to distinguish intermittent connectivity with 100% loss from disconnected operation. However, information available to passive systems is not sufficient for deciding this problem.

In this paper, we propose *Hybrid Monitoring*, a novel WSN health monitoring system that utilizes passively reconstructed packet information while only adding minimal extra information required for mitigating the aforementioned problem to every packet. The design of this system is based on the observation that a number of failure events, *e.g.*, failed packet transmission attempts, a congested channel, or full packet queues towards the sink, share the common characteristic of affected packets to wait longer inside the network. Resulting delays range from a few seconds to several minutes. Initially being interested in inferring the actual number of failure events occurred solely from passively reconstructed per-hop timing information [13], we found that accuracy and coverage are significantly improved when asking for minimal collaboration from inside the network. Concretely, our measurements prove that adding one bit per packet is sufficient for resolving the majority of situations in which a completely passive system would heavily over or underestimate the metric analyzed.

Implementing the semantics of the so called "problem bit" requires only minimal modifications on the sensor nodes. *Hybrid Monitoring* does not inject additional packets into the network, but needs only one extra bit to be attached to every packet that is generated by the existing sensor network application. A sensor node sets the problem bit of a locally generated packet if at least one failure event has occurred since the generation of the previous packet. Here, a failure event refers to any behavior that requires a packet transmission to be repeated or postponed despite of the node being connected to the network. Subject to further refinement w.r.t. the concrete protocol used, failure events are unacknowledged packet transmissions, the radio not being ready for transmission, and the queue at the next hop being full.

Problem bit information from inside the network is then combined with passively reconstructed path and per-hop timing information after packets have been received at the sink. Here, the main principle is to analyze the times that individual packets spent at a node. The resulting health metric is the actual number of failure events that occurred between the generation of two subsequent data packets. Exemplary scenarios that contribute to a rise in this metric are a lossy channel, an unbalanced routing tree, timing problems on the node, and even hardware issues. The number of failure events is an early diagnosis metric, *i.e.*, failures can be detected even before a protocol eventually starts to drop packets.

The contribution of this paper is as follows:

- We propose *Hybrid Monitoring*, a novel health monitoring system that complements a passive health monitoring method, *i.e.*, the analysis of passively reconstructed timing information, with a minimally active component. Sensor nodes are only required to set the problem bit according to very simple semantics. Retrieved information are useful for identifying lossy links and traffic bottlenecks, and to generally monitor the performance of a system over time. Our approach minimizes communication and computation overheads. As it can be completely integrated into an existing application, *Hybrid Monitoring* is suited for continuous operation. If needed, *Hybrid Monitoring* can also be combined with heavier debugging components that may be automatically activated based on the number of detected failure events.
- The algorithmic framework of *Hybrid Monitoring* is presented based on a formal model of a data collection application.
- We implemented our system on real nodes on top of the Dozer [1] ultra low-power data collection protocol. Extensive experiments of testbeds on up to 96 nodes in size show that the number of occurred failure events can be estimated with a high confidence. More than 90% of the decisions are done correctly when the number of occurred failure events is used as the input of a runtime monitoring application that eventually triggers further action, *e.g.*, notifies a network operator.

The remainder of this paper is structured as follows: After presenting related work in Section II, the general problem that this work is trying to tackle is presented in Section III. Section IV gives a high-level description of the proposed system. Assumptions made are summarized in a formal model of a data collection system that is shown in Section V, a detailed description of algorithms used is presented in Section VI. The results of our evaluation on real hardware are shown in Section VII, Section VIII discusses the broader applicability and the limitations of the presented system. Section IX concludes this paper.

## II. Related Work

After initial lessons learned [27], [15], a growing tool support has helped to make WSNs nowadays being ready for long-term missions at scale. The debugging of a system whose state is distributed over many devices being a very challenging task, many efforts have especially focused on this particular problem. As a result, various methods, *e.g.*, simulation [16], [23], testbeds [10], [17], co-located sniffing hardware [4], in-network debugging facilities [28], [2], and in-network state loggers [20], [25], are now available for supporting developers during the whole development lifecycle of a WSN.

Once a system is ready for production, informed network operation again requires distributed state to be accessible. While debugging facilities that were used during the development are often too heavyweight or even not applicable at the final deployment site, only the most important runtime information is continuously collected after deployment. A common design found in many solutions, *e.g.*, [29], [24], [26], [18], [5], is to collect information within the network, and to offload the analysis process to the network sink. Variations include the usage of in-network aggregation techniques, *e.g.*, [29], [26], or to run parts of the analysis already inside the network [26], [21]. Actively sending state can be completely avoided when a higher uncertainty can be tolerated [19].

The system proposed in this paper extends multi-hop network tomography [13], a passive method for the reconstruction of the packet path, per-hop ordering and per-hop timing information of individual packets, with an as non-intrusive as possible active components running inside the network. Only very simple semantics need to be implemented on the nodes, only one bit of extra information is added to every packet.

When comparing this new approach to Sympathy [24], we first find a considerable amount of traffic being saved by the passive reconstruction of path information. While Sympathy is more flexible when it comes to freely choosing an reporting interval, *i.e.*, health data is not coupled with application traffic, our approach is automatically achieving a higher temporal resolution when the application is sending more data. In contrast, an active scheme such as Sympathy might even have to reduce its operation in that case.

Still, certain information that is reported by Sympathy, *e.g.*, the node uptime, is not covered by our approach. While not having this information certainly leaves more uncertainty in the analysis, there is also no other choice but active transmission when such information is required.

Agnostic Diagnosis [18] is another system that is very similar to Sympathy with the main difference of making use of so called correlation graphs instead of utilizing a simple decision tree. While transmitting more information, *i.e.*, periodically sampled counter values, certainly increases confidence, our solution is working on a comparable level of detail, *i.e.*, not only considering fatal errors, but already inefficient behavior, *e.g.*, packet retransmissions.

Having the purpose of efficiently transmitting spatial distributions of scalar values, *e.g.*, energy levels, eScan [29] can be considered rather orthogonal or even complementary to this work. If certain scalar metrics, *e.g.*, the node uptime, the current battery voltage, or the humidity within the enclosure, are required, using the presented in-network aggregation scheme would certainly help to reduce the overhead that is added to the network.

Probably closest to this work is PAD [19], a passive health monitoring approach. Here, the root causes for sensor and node failures are inferred using a probabilistic approach. While the whole decision process is based on only sporadically inserted path marks that are then used to generate decision graphs, the detection performance of this approach is limited.

*Hybrid Monitoring* distinguishes itself from PAD and other mentioned systems by using proven to be correct [13] path and per-hop timing information of individual packets as its input. While the detection of sensor faults, *e.g.*, [9], is orthogonal to this work, *Hybrid Monitoring* is also not focussing on node liveness, but on the detection of inefficient operation that already degrades system performance and may also lead to a fatal error if not discovered beforehand. Although technically

being an active system, the overhead of *Hybrid Monitoring* is closest to that of passive systems, *e.g.*, PAD [19].

## III. Inferring State from Minimal Information

The utility of a sensor network application is threatened when measurement locations are temporarily not served due to node failures. The timely replacement of a node after the fact can be difficult when long and expensive travel is needed. Instead, an early diagnosis can reduce maintenance costs when degraded but not yet failed components can already be replaced during scheduled maintenance periods.

In this paper, we want to explore the minimal amount of transferred information needed for making informed decisions with confidence. As the available computational power is an order of magnitude higher as well as cheaper after packets have been received at the sink, we are especially interested in solutions that ask for the minimum amount of communication and computation to take place inside the network. In this context, we want to find answers to the following questions: How much state is already known implicitly, *e.g.*, from the timing behavior of a system? How large is the gain from adding extra in-band information to an initially passive method?

## IV. Hybrid Monitoring

Various reasons can increase the waiting time of a packet at a node. Radio and wireless channel need to become available, sending over a lossy channel can require multiple transmission attempts. Exceptionally long waiting times can occur when a node is disconnected from the network. Mentioned events share the characteristic that any of their occurrences results in an increased end-to-end packet delay. Measuring the end-to-end delay for each individual packet is a common practice in data collection applications.

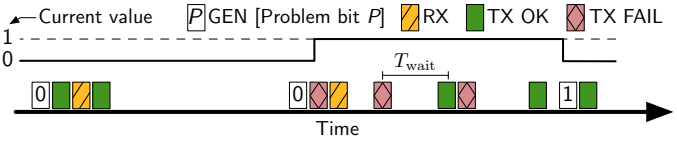



Figure 1. Exemplary scenario with several packets being generated, received from other nodes and eventually being transmitted to a parent. The local variable that corresponds to the problem bit is set to 1 on the first occurrence of a failure event. It is again cleared after its current value has been copied into the most recently generated packet.

Looking at the exemplary situation in Figure 1, we can see three packets that are generated at the node under observation. Two more packets are received from other nodes. If a transmission failed, the next transmission is assumed to happen a fixed time $T_{\text{wait}}$ later.

While we can see three transmission errors happening between the generation of the second and the third locally generated packet in Figure 1, the question now is how we can reconstruct the number of occurred failure events after packets have been received at the sink? The proposed solution for this problem is depicted in Figure 2. Assuming that we are able to split the measured end-to-end packet delay at the sink into individual contributions per node, this information can then be further decomposed and used for inferring the number of occurred failure events.

In practice, several challenges need to be solved before the number of failure events can be estimated with confidence. The end-to-end delay first of all being composed out of the contributions of all nodes along the packet path, the contribution of each single node can again be split into disjoint events. Delays caused by a node being disconnected from the network must be filtered out in order to not be labeled as failure events. This separation is in particularly made difficult by the fact that connected and disconnected operation can be arbitrarily interleaved.

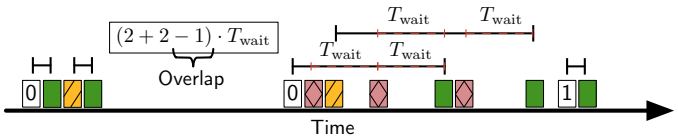



Figure 2. Packet timing on a node under analysis. Introduced delays are caused by packet transmissions errors (see Figure 1) that force subsequent packets to wait. Given access to this time information, we can reconstruct the number of occurred failure events by decomposing the sojourn time of each packet.

In order to solve this problem, we propose that sensor nodes have to set the so called problem bit on the occurrence of a failure event during connected operation. As illustrated in Figure 1, the current value of the corresponding variable gets copied and then cleared every time when a new packet is generated.

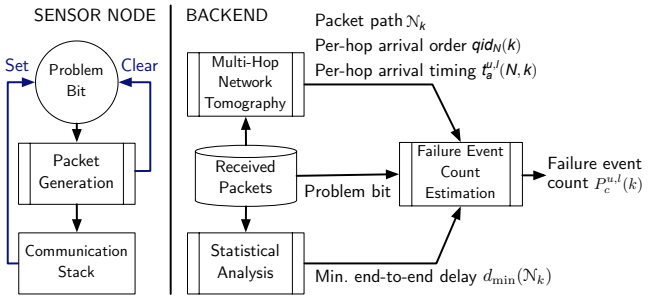



Figure 3. Information from minimal collaboration inside the network is augmented after packets have been received at the sink.

The overall design of *Hybrid Monitoring* is shown in Figure 3. Each node maintains a local variable that corresponds to the problem bit. As long as a node is connected to a parent, the problem bit is set on any occurrence of a failure event. Every time when a new packet is generated, the current value of the problem bit variable is written into that packet and then cleared. Apart from helping to not mistake disconnected operation for a high number of failure events, the semantics used also support the later analysis by mitigating the effects of imperfect timing information.

The simple semantics inside the sensor network are complemented by a set of algorithms that are executed outside the network after packets have been received at the sink. First, multi-hop network tomography [13] is used for reconstructing the packet path and per-hop timing information of individual packets. While obtained worst-case waiting times from this method can be very pessimistic, the final decision on the number of occurred failure events is made more robust by adding results from a statistical analysis of packet end-to-end delays as a second data source. Here, continuously updated

Table I. OVERVIEW OF SYSTEM MODEL VARIABLES

| | |
|---|---|
| **Packet application headers** | |
| $o(k)$ | Source node network address |
| $\tilde{t}_g(k)$ | Estimated packet generation time |
| $P(k)$ | Problem bit |
| **Added on arrival at the sink** | |
| $t_b(k)$ | Arrival time at the sink |
| **From post-processed packet headers** | |
| $id_N(k)$ | Packet generation index reflecting the correct order of generation for packets originating from a node $N$ |
| **From multi-hop network tomography** | |
| $\mathcal{N}_k$ | Packet path |
| $qid_N(k)$ | Queue index reflecting the order of arrival at the queue of node $N$ |
| $t_a^{u,l}(N,k)$ | Upper and lower bounds on the arrival time of $k$ at node $N$ |
| $t_d^{u,l}(N,k)$ | Upper and lower bounds on the departure time of $k$ from node $N$ |
| **From analysis** | |
| $\Delta^{u,l}(k)$ | Upper and lower bounds on the accuracy of the estimated packet generation time $\tilde{t}_g(k)$ |
| $t_g^{u,l}(k)$ | Upper and lower bounds on the unknown packet generation time $t_g(k)$ |
| $d_{\min}(\mathcal{N}_k)$ | Minimum end-to-end delay of path $\mathcal{N}_k$, partially constant |
| **Implementation-specific parameters** | |
| $T_{\text{wait}}$ | Waiting time before the next transmission attempt after a failure event |

estimates of the minimum end-to-end delay, *i.e.*, the sojourn time of a packet without waiting times due to disconnects or failure events, of every path are used for estimating the waiting time of delayed packets.

## V. SYSTEM MODEL

**Multi-hop data collection.** We assume a data collection application in which sensor nodes generate data that is then send to a sink. Sensor nodes and sink maintain a single FIFO queue that is used for both locally generated and forwarded packets. Communication is based on a tree-based multi-hop routing protocol. The network operation is subject to phenomena that are common to wireless sensor networks, *i.e.*, packet loss and packet duplication. The network sink is the only device that has access to a synchronized clock. Inside the network time is measured on clocks of limited resolution and with a bounded drift.

**System model variables.** Table I gives an overview of the variables used in this model. Packets are uniquely identified using a numerical index that reflects the order of arrival at the sink. For every received packet $k$, we first of all assume to have access to the source address $o(k)$, the problem bit $P(k)$, and the packet path $\mathcal{N}_k$. Further information can be grouped as follows:

- **Timing information:** We define $\tilde{t}_g(k)$ as the estimate of the unknown packet generation time $t_g(k)$. The error of this estimate is bounded by $t_g^l(k) = \tilde{t}_g(k) - \Delta^l(k) \leq t_g(k) \leq \tilde{t}_g(k) + \Delta^u(k) = t_g^u(k)$. Here, $\Delta^l(k)$ and $\Delta^u(k)$ denote the upper and lower bounds on the error of the time-stamping mechanism used. The arrival time at the sink $t_b(k)$ is assumed to be measured on a perfect clock. For each hop $N$ that the packet visited, *i.e.*, $N \in \mathcal{N}_k$, we define $t_a^u(N,k)$ and $t_a^l(N,k)$ as the upper and lower bounds on the unknown arrival time $t_a(N,k)$ of packet $k$ at node $N$. Likewise, $t_d^u(N,k)$ and $t_d^l(N,k)$ denote the upper and lower bounds on the not accurately known departure time $t_d(N,k)$ of packet $k$ at node $N$. It holds that $t_a^l(N,k) \leq t_a(N,k) \leq t_a^u(N,k)$ and $t_d^l(N,k) \leq t_d(N,k) \leq t_d^u(N,k)$, respectively.
- **Sequencing information:** Apart from the index $k$ that is used to identify packets in this model, we also assume the order of which packets at a node $N$ were generated to be reflected by the packet generation index $id_N(k)$. The packet generation index of a packet $k$ is assumed to be by one larger than the packet generation index of the packet that was generated at the same source $o(k)$ immediately before $k$. Similarly, the queue index $qid_N(k)$ reflects the order in which packets arrived at the queue of a node $N$. A packet $k$ that arrived at a node $N$ has a higher queue index than any other packet that arrived at $N$ before $k$. Please note that the packet generation index $id_N(k)$ not only allows to order packets from the same source $N$, but can also be used to detect packet loss. In contrast, the queue index $qid_N(k)$ allows to order packets from any source that visited node $N$, but can not be used for detecting missing packets.

**Passive reconstruction.** Packet path $\mathcal{N}_k$, queue index $qid_N(k)$, and per-hop timing information $t_a^{u,l}(N,k)$ and $t_d^{u,l}(N,k)$ are not implicitly transferred as part of every packet, but reconstructed using multi-hop network tomography [13]. For us being able to use this passive reconstruction method, our system must also fulfill the requirements of multi-hop network tomography. Most notably, multi-hop network tomography assumes that all sensor nodes are generating data. Furthermore, it is assumed that the address of the first receiver of a packet, *i.e.*, the current parent of the source node at the moment of transmission, is transmitted as part of the packet. Further assumptions made including case studies in which multi-hop network tomography is applied to systems running CTP [8] and Dozer [1] can be found in the corresponding paper [13].

**Protocol operation.** Lastly, we make the following further assumptions regarding the protocol operation:

- Sensor nodes are aware of their connectivity state. A sensor node is in a connected operation state as long as it can regularly communicate with its designated parent node, *e.g.*, successfully receive beacons and packet acknowledgements. The transition to a disconnected state is based on certain rules defined by the protocol, *e.g.*, the expiration of a timer or an error counter hitting a threshold. The phase of no communication prior to the disconnect decision still counts as connected operation.
- Certain errors, *e.g.*, a lost beacon message, a missed packet acknowledgement, or the radio being in the wrong state, can require the transmission of a packet to be postponed or repeated. In such a case, we assume that the next transmission attempt is made after waiting for a implementation-specific time $T_{\text{wait}}$. $T_{\text{wait}}$ can include a jitter $J \ll T_{\text{wait}}$.
- The minimum end-to-end delay $d_{\min}(\mathcal{N}_k)$ of a path $\mathcal{N}_k$ is defined as the time that a packet needs for traversing $\mathcal{N}_k$ in the absence of both transmission errors and

disconnected operation. $d_{\min}(\mathcal{N}_k)$ is assumed to be constant with a bounded deviation smaller than a defined $\delta$ for at least small numbers of subsequent transmissions along the same path.

## VI. From a Single Bit to a Scalar Value

For every received packet $k$ with a set problem bit, *i.e.*, $P(k) \equiv 1$, we want to determine the upper and lower bounds $P_c^u(k)$ and $P_c^l(k)$ on the unknown number of failure events that occurred at the source of packet $k$ between the generation of packet $k$ and the generation of the immediate predecessor of $k$. In contrast to the introductory example that was presented in Section IV, we cannot access perfect per-hop timing information in the real case. Instead, our algorithms are required to be robust to imperfect timing information.

Estimating the unknown number of occurred failure events involves two main steps. First, we need to determine the set $\mathcal{W}$ of all packets that may have been waiting for transmission at node $o(k)$ within the temporal scope of $P(k) \equiv 1$, *i.e.*, before the generation of packet $k$ and after the generation of the immediate predecessor of $k$. In other words, the set $\mathcal{W}$ contains all packets whose end-to-end delay may be affected by the failure events that caused the problem bit $P(k)$ to be set. The number of occurred failure events is then estimated by analyzing the timing information of all packets $w \in \mathcal{W}$.

The analysis of the waiting time of a single packet $k$ at a node $N$ is presented in Section VI-A. An approach that constructs the number of occurred failure events as the sum of non-overlapping, individual contributions of multiple packets is then shown in Section VI-B. Section VI-C presents the algorithm that is used for continuously estimating the current minimum end-to-end delay $d_{\min}(\mathcal{N}_k)$ of a path $\mathcal{N}_k$.

For clarity and brevity, it is assumed that packet path and per-hop timing information of all packets can be reconstructed. In practice, this information is missing for a small fraction of packets that are not "reliable" and can thus not be used in a multi-hop network tomography [13]. In consequence, this small amount of packets can not be used in the following analysis.

### A. Packet Waiting Time Analysis

Given a packet $k$ with a problem bit $P(k) \equiv 1$, the analysis starts with determining the set $\mathcal{W}$ of packets whose timing may be affected by the failure events that caused the problem bit $P(k)$ to be set. As the temporal scope of the problem bit is defined as the time period that starts after the generation of the previous packet and ends before the generation of the current packet $k$, we first need to find the predecessor $f$ of packet $k$. This packet $f$ has the two properties that it was generated at the same node as $k$ and that its packet generation index $id_{o(k)}(f)$ must be exactly one lower than the packet generation index $id_{o(k)}(k)$ of packet $k$:

$$f := \arg\max_x id_{o(k)}(x) \text{ for all } x : o(x) \equiv o(k) \\ \wedge\, id_{o(k)}(x) \equiv id_{o(k)}(k) - 1$$

The temporal scope of the problem bit can now be expressed by the precise packet generation times of $f$ and $k$: $t_{p,s}(k) = t_g(f) < t < t_g(k) = t_{p,e}(k)$. As precise packet generation times are not known in practice, we must resort to the usage of upper and lower bounds $t_g^{u,l}(f)$ and $t_g^{u,l}(k)$ that reflect the uncertainties of obtained packet generation times. We define the upper and lower bounds on the beginning and end of the time interval $]t_{p,s}(k), t_{p,e}(k)[$ for which the problem bit $P(k)$ is valid as follows:

$$t_{p,s}^l(k) := t_g^l(f) \qquad t_{p,e}^l(k) := t_g^l(k)$$
$$t_{p,s}^u(k) := t_g^u(f) \qquad t_{p,e}^u(k) := t_g^u(k)$$

The set $\mathcal{W}$ of all packets that may be affected by the failure events covered by $P(k)$ is then build by selecting all packets that (i) arrived at the node $N := o(k)$ before packet $k$ and (ii) may have departed after the generation of packet $f$:

$$\mathcal{W} := \{w \mid qid_N(w) < qid_N(k) \wedge t_d^u(N, w) > t_g^l(f)\}$$

For each packet $w \in \mathcal{W}$, we then determine the minimum and maximum number of $T_{\text{wait}}$ long intervals that $w$ can have spent at node $o(k)$ within $]t_{p,s}(k), t_{p,e}(k)[$. To achieve that, we first determine the shortest and longest time interval that packet $w$ can have spent at $o(k)$ within $]t_{p,s}(k), t_{p,e}(k)[$. The shortest time interval $s_{\min}(w)$ is based on the assumption that $w$ arrived as late as possible while again leaving as early as possible:

$$s_{\min}(w) := \min\left(t_d^l(w), t_{p,e}^l(k)\right) - \max\left(t_a^u(w), t_{p,s}^u(k)\right)$$

Likewise, the calculation of $s_{\max}(w)$ is based on the counter assumption of arriving as early as possible while leaving as late as possible:

$$s_{\max}(w) := \min\left(t_d^u(w), t_{p,e}^u(k)\right) - \max\left(t_a^l(w), t_{p,s}^l(k)\right)$$

As the longest possible time interval $s_{\max}(w)$ is often pessimistic, the final calculation of the number of $T_{\text{wait}}$ long intervals also includes an estimation of the maximum added delay that packet $w$ can have experienced along the complete path due to failure events. The maximum added delay due to failure events $e_{\max}(w)$ is the difference of the actual end-to-end delay of packet $w$ and the minimum end-to-end delay $d_{\min}(\mathcal{N}_w)$ along the path $\mathcal{N}_w$ that packet $w$ travelled along:

$$e_{\max}(w) := t_b(w) - \tilde{t}_g(w) - d_{\min}(\mathcal{N}_w)$$

The calculation of the minimum and maximum number of $T_{\text{wait}}$ long intervals that packet $w$ spent at node $o(k)$ finally also accounts for a jitter $J \ll T_{\text{wait}}$:

$$v_{\min} := \left\lfloor \frac{s_{\min}(w) + J}{T_{\text{wait}}} \right\rfloor, v_{\max} := \left\lfloor \frac{\min(s_{\max}(w) + J, e_{\max}(w))}{T_{\text{wait}}} \right\rfloor$$

### B. Failure Event Count Estimation

In order to estimate the total number of failure events that occurred at node $N := o(k)$ within the time interval $]t_{p,s}(k), t_{p,e}(k)[$, we must combine the waiting times of all packets $w \in \mathcal{W}$. Here, the challenge is that the time intervals in which different packets can have waited at node $N$ may partially or fully overlap, *e.g.*, when a single failure event causes multiple packets to wait.

The algorithm used for determining the upper and lower bounds $P_c^{u,l}(k)$ on the unknown number of occurred failure events at node $N$ is presented in Algorithm 1. Given a packet $k$, bounds are immediately set to zero if the problem bit of $k$ is not set (line 2). Else, the procedure starts with determining packet $f$ that was generated at node $N$ directly before packet $k$ (line 3). Next, the set of packets $\mathcal{W}$ that might be affected by a problem at $N$ during the time interval $]t_{p,s}(k), t_{p,e}(k)[$ is determined (line 5).

**Algorithm 1:** Algorithm for estimating the number of occurred failure events

**Input** : Packet $k$
**Output**: Upper and lower bounds $P_c^{u,l}(k)$ on the number of failure events at node $N := o(k)$ between the generation of packet $k$ and its immediate predecessor $f$

1 $P_c^u(k) \leftarrow 0$; $P_c^l(k) \leftarrow 0$; $\mathcal{S} \leftarrow \{\}$ ;
2 **if** $\neg P(k)$ **then return**;
3 $f \leftarrow \arg\max_x id_N(x)$ for all $x : o(x) \equiv N$
4 $\quad \wedge\ id_N(x) \equiv id_N(k) - 1$ ;
5 $\mathcal{W} \leftarrow \{w \mid qid_N(w) < qid_N(k) \wedge t_d^u(N,w) > t_g^l(f)\}$ ;
6 **if** $\mathcal{W} \equiv \emptyset$ **then return**;
7 $w \leftarrow \arg\min_x qid_N(x)$ for all $x : x \in \mathcal{W}$ ;
8 **if** $t_a^u(N,w) < t_g^l(f) \wedge\ t_d^l(N,w) > t_g^u(k)$ **then**
9 $\quad P_c^l(k) \leftarrow 1$; $P_c^u(k) \leftarrow \lfloor (t_g^u(k) - t_g^l(f))/T_{\text{wait}} \rfloor$ ;
10 $\quad$ **return**;
11 $r^l \leftarrow t_g^l(f)$; $r^u \leftarrow t_g^u(f)$ ;
12 **while** *true* **do**
13 $\quad w \leftarrow \arg\min_x qid_N(x)$ for all $x : x \in \mathcal{W} \setminus \mathcal{S}$ ;
14 $\quad r^l \leftarrow \max(t_a^l(N,w), r^l)$ ;
15 $\quad r^u \leftarrow \max(t_a^u(N,w), r^u)$ ;
16 $\quad v_{\min}(w)$, $v_{\max}(w) \leftarrow$ AnalyzePacket($w$, $r^l$, $r^u$, $k$)
17 $\quad P_c^l(k) \leftarrow P_c^l(k) + v_{\min}(w)$ ;
18 $\quad P_c^u(k) \leftarrow P_c^u(k) + v_{\max}(w)$ ;
19 $\quad r^l \leftarrow r^l + v_{\min}(w) \cdot T_{\text{wait}}$; $r^u \leftarrow r^u + v_{\max}(w) \cdot T_{\text{wait}}$ ;
20 $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{w\}$ ;
21 $\quad$ **if** $\mathcal{W} \setminus \mathcal{S} \equiv \emptyset$ **then break** ;
22 **end**
23 **if** $P_c^l(k) \equiv 0$ **then** $P_c^l(k) \leftarrow 1$;
24 **if** $P_c^u(k) \equiv 0$ **then** $P_c^u(k) \leftarrow 1$;

If the first analyzed packet arrived at $N$ before packet $f$ was generated and also departed from $N$ after packet $k$ was generated (line 8), the estimation of the number of occurred failure events is hindered by both the best-case and the worst-case waiting time being determined too pessimistic as the temporal distance of packets $f$ and $k$. In this case, we set the number of failure events to at least one and to at most the highest number of $T_{\text{wait}}$ long intervals that fit into the time interval between the generation of $f$ and $k$ (line 9).

Otherwise, all packets $w \in \mathcal{W}$ are processed in the order of arrival at node $N$ (line 13). As packet are assumed to leave a node in the same order as they arrived, packets that arrived earlier are analyzed before later arriving and thus also later departing packets.

Newly introduced variables $r^u$ and $r^l$ have the purpose of avoiding waiting times to be counted more than once. $r^u$ and $r^l$ are first initialized with the packet generation times of packet $f$ (line 11), and then continuously incremented when either the arrival of the next analyzed packet is not overlapping with the waiting times of the previous packet $w$ (lines 14 and 15), or when at least one new waiting interval of length $T_{\text{wait}}$ was detected (line 19).

The actual estimation of the minimum and maximum waiting time of the currently analyzed packet $w$ (line 16) is located in Algorithm 2. The beginning of the time interval in which this algorithm looks for waiting times is defined by $r^u$ and $r^l$, the end of the time interval is given by the generation time of packet $k$. The upper and lower bounds on the total number of occurred failure events $P_c^{u,l}(k)$ are determined by building the sum of the individual, non-overlapping contributions (lines 17 and 18)

Processing the next packet $w$ continues until all packets $w \in \mathcal{W}$ have been analyzed (line 21). Already analyzed packets are members of both the set $\mathcal{W}$ and the newly introduced set of already seen packets $\mathcal{S}$.

If all packets $w \in \mathcal{W}$ have been processed without any failure event being found, *e.g.*, because of packets being lost, the problem bit $P(k)$ being set allows us to infer that there must have been at least one occurrence of a failure problem (lines 23 and 24).

**Algorithm 2:** AnalyzePacket

**Input** : Analyzed packet $w$, start of analysis time window $r^{u,l}$, packet $k$
**Output**: Minimum and maximum number $v_{\min}(w)$ and $v_{\max}(w)$ of $T_{\text{wait}}$ long time intervals within $r^{u,l}$ and the generation of packet $k$

1 **begin**
2 $\quad s_{\min}(w) \leftarrow \min(t_d^l(N,w), t_g^l(k)) - r^u$ ;
3 $\quad v_{\min}(w) \leftarrow \lfloor (s_{\min}(w) + J)/T_{\text{wait}} \rfloor$ ;
4 $\quad s_{\max}(w) \leftarrow \min(t_d^u(N,w), t_g^u(k)) - r^l$ ;
5 $\quad e_{\max}(w) \leftarrow t_b(w) - \tilde{t}_g(w) - d_{\min}(\mathcal{N}_w)$ ;
6 $\quad v_{\max}(w) \leftarrow \lfloor \min(s_{\max}(w) + J, e_{\max}(w))/T_{\text{wait}} \rfloor$ ;
7 **end**

Depending on the concrete communication protocol used, implementation-specific details can require further refinements of the procedure that is described in Algorithm 1. For example, a sensor node running Dozer [1] needs approximately 60 sec for negotiating a connection with a new parent node. In consequence, estimation results are improved when those 60 sec are excluded from being accounted as waiting times due to a failure event.

### C. Estimation of Minimal End-to-End Packet Delay

So far, the minimum end-to-end delay $d_{\min}(\mathcal{N}_k)$ of the path $\mathcal{N}_k$ was assumed to be known. The algorithm that is used for continuously updating the estimates of the path-dependent minimum end-to-end delay is shown in Algorithm 3. Here, the main assumption is that the minimum end-to-end delay equals to the end-to-end delay of a packet that has not been delayed inside the network due to a failure event or a disconnect. While that information is not perfectly known, the two requirements are verified using a worst-case analysis and a heuristic approach.

Before the first packet of a path $\mathcal{N}_k$ has been received, the minimum end-to-end delay $d_{\min}(\mathcal{N}_k)$ of this path is initialized with a protocol-dependent parameter, *e.g.*, the product of the path length $|\mathcal{N}_k|$ and a defined maximum processing time of a packet $T_{\text{queue,max}}$.

Given a newly arrived packet $k$, the algorithm starts with traversing all nodes $N$ along the path $\mathcal{N}_k$ of packet $k$ (line 2). At each node $N$, it is then analyzed if $k$ may have been at node $N$ during a problem. This is done by first identifying all packets $\mathcal{P}$ that were generated at node $N$ during the maximum time that $k$ can have spent at $N$, and then determining if the problem bit of any of those packets was set (line 3). Packet $k$ can only be used to set a new minimum end-to-end delay $d_{\min}(\mathcal{N}_k)$ if all involved packets had the problem bit not set (line 7).

The exclusion of $k$ being delayed due to a disconnect is based on the comparison of the end-to-end delay of packet $k$ with the existing value of $d_{\min}(\mathcal{N}_k)$. Packet $k$ is assumed to not have been delayed by a disconnect if its end-to-end delay is at most by a $\delta$ higher than the former value (line 8).

**Algorithm 3:** UPDATEPATHDELAYESTIMATE

**Input** : Newly arrived packet $k$
**Output**: Updated estimate of the minimum end-to-end delay $d_{\min}(\mathcal{N}_k)$ on path $\mathcal{N}_k$

```
1  begin
2      foreach N ∈ N_k do
3          P ← {x | o(x) ≡ N ∧ t_g^u(x) > t_a^l(N,k)
4                 ∧ t_g^l(x) < t_d^u(N,k) ∧ P(x) ≡ 1} ;
5          if P ≢ ∅ then break ;
6      end
7      if P ≡ ∅ then
8          if t_b(k) − t̃_g(k) < d_min(N_k) + δ then
9              d_min(N_k) := t_b(k) − t̃_g(k) ;
10         end
11     end
12 end
```

## VII. MULTI-TESTBED EVALUATION

In this section, we present the results of extensive testbed experiments that we conducted in order to (i) quantify the gain in confidence when nodes are actively transmitting one extra bit per packet, and (ii) to evaluate the performance of *Hybrid Monitoring* when compared to ground truth.

All testbed experiments are based on an implementation of *Hybrid Monitoring* on top of the TinyOS implementation of Dozer [1]. Dozer is a state-of-the-art low-power data collection protocol that is used in multiple long-term deployments [14]. Apart from implementing the semantics of the problem bit and adding instrumentation code for outputting ground truth information over the serial port of the node, no further modifications were done to the standard implementation of Dozer.

Three sets of nodes are used in this evaluation: 96 Tmote Sky (MSP430+CC2420 radio) nodes of the TWIST [10] testbed allow us to study the performance of the *Hybrid Monitoring* in a large network. Tests with two kinds of radio hardware are possible on the FlockLab [17] testbed. Here, we are using 30 Tmote Sky nodes and 30 TinyNode184 (MSP430+SX1211) nodes. Because the 868 MHz radio of the TinyNode184 is operating on a different band than local WiFi networks, traces obtained using TinyNode hardware include less failure events due to packet loss.

The integration of *Hybrid Monitoring* into the Dozer data collection protocol is described in Section VII-A. The four test configurations used and the characteristics of the data obtained are presented in Section VII-B. In Section VII-C, the performance of *Hybrid Monitoring* is compared with a completely passive system that does not make use of the problem bit. Comparisons with ground truth show significant gains in confidence when the problem bit is used. The performance of *Hybrid Monitoring* in the context of an runtime application that triggers further action, *e.g.*, notifies a network operator, based on a pre-defined threshold value is presented in Section VII-D.

### A. Integration into the Dozer Data Collection Protocol

The operation of the Dozer protocol is divided into rounds of a fixed length $T_{\text{round}}$, a round can be further divided into a number of slots. The slot assignment is based on a local schedule that includes communication with the parent node, communication with child nodes, a phase in which new child nodes can join, and the execution of local processing tasks. The local schedule of a node is modified every time when the node switches to another parent, or when a new child node connected. Additionally, a sensor node can decide to shift the position of the local processing slots, *e.g.*, because the old position is likely to interfere with a communication slot.

Waiting packets are transmitted during the slots that are reserved for communication with the parent node. If a packet transmission fails, the next transmission is postponed to the following round. Thus, the waiting time $T_{\text{wait}}$ is $T_{\text{wait}} := T_{\text{round}}$. For the remainder of this evaluation, $T_{\text{round}}$ is set to 30 sec with a jitter of $\pm 2$ sec.

The problem bit is set on the occurrence of at least one of the following failure events that share the common characteristic of postponing the next packet transmission by $T_{\text{wait}} := 30$ sec plus jitter:

- **Missed beacon:** A beacon message from the parent did not arrive within the expected time. In consequence, Dozer is not trying to upload packets.
- **Missed acknowledgement:** A packet acknowledgement from the parent did not arrive within the expected time. In consequence, the packet at the head of the queue must be retransmitted.
- **Busy radio:** The radio was not ready for a packet transmission. This can happen when the radio is occupied by another task, *e.g.*, currently handling a connection request of another node.
- **Full parent queue:** None or not all locally queued packets could be transmitted because of a full queue at the parent node.

Apart from setting the problem bit, listed events also cause a log entry to be generated. Further log entries are generated on

Table II. CHARACTERIZATION OF DATA FROM TESTBED EXPERIMENTS BEFORE AND AFTER MULTI-HOP NETWORK TOMOGRAPHY

| | | | | | | Received (Ground Truth) | | | After Tomography | | Waiting Time Uncert. | | Min. Delay Error | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **N** | **D** | **IPI** | **H** | **DY** | **Total** | $P(k) \equiv 1$ | **F. events** | **Total** | $P(k) \equiv 1$ | $p_{0.9}$ | $p_{0.98}$ | **Mean** | $p_{0.85}$ | $p_{0.98}$ |
| | | | | | | | | *Tmote Sky* | | | | | | | |
| A) | 96 | 17 hours | 120 sec | 5 | 99.5% | 49,305 | 15,772 | 23,536 | 96.5% | 96.6% | 98 sec | 120 sec | 12 sec | 27 sec | 44 sec |
| B) | 26 | 14 hours | 120 sec | 4 | 99.7% | 10,654 | 3,768 | 6,823 | 92.8% | 93.2% | 120 sec | 178 sec | 11 sec | 27 sec | 44 sec |
| C) | 30 | 9 hours | 60 sec | 4 | 99.8% | 14,940 | 6,750 | 10,496 | 89.7% | 87.2% | 78 sec | 118 sec | 11 sec | 26 sec | 45 sec |
| | | | | | | | | *TinyNode* | | | | | | | |
| D) | 30 | 24 hours | 120 sec | 5 | 99.9% | 20,613 | 7,015 | 11,053 | 95.8% | 95.8% | 111 sec | 123 sec | 9 sec | 25 sec | 41 sec |

the generation of a packet, on the arrival of packet from another node, each time when the node tries to transmit a message to its parent, when the node disconnects from its parent, and when the node connected to a new parent. Log entries are immediately timestamped using the local node clock and then put into the memory of the node. The outputting of buffered log entries over the serial port is done at the next slot that has been assigned for local processing tasks. Likewise, the generation of new packets is also executed in such a slot.

### *B. Data Preparation Results*

The test configurations used and the characteristics of results obtained are shown in Table II. For each experiment, Table II lists the number of nodes (N), the duration (D) of the test in hours, the inter-packet interval (IPI) used, the height of the routing tree (H), the achieved data yield (DY), the total number of received packets, the number of received packets with a set problem bit, and the number of failure events that occurred during the experiment. Chosen test durations are mainly driven by the availability of testbed resources. Periodic packet generation is no requirement of our method, however, periodic packet generation allows us to better analyze the statistical properties of received timing information.

Furthermore, Table II also lists the amount of packets for which information can be reconstructed using multi-hop network tomography [13]. Numbers obtained are within the results of the original paper, packets are treated equally independent of the problem bit. The waiting time uncertainty is defined as the difference between the maximum (worst-case) and the minimum (best-case) time that a packet can have spent at a node. Results are described by the 90th and the 98th percentile. Found uncertainties are also in line with previous results.

The last three columns of Table II quantify the results of Algorithm 3 that is used for estimating the minimum end-to-end delay of individual packets paths. Results listed include the mean value, the 85th percentile and the 98th percentile. Here, mainly non-determinisms found in the Dozer protocol operation, *e.g.*, slots for local processing tasks being moved from the end to the beginning of a round, introduce an estimation error that is larger than $T_{\text{wait}}$ for approximately one fifth of the packets.

### *C. Comparison with Completely Passive System*

For being able to quantify the gain by adding one bit to every packet, we created completely passive versions of Algorithm 1 and Algorithm 3. The resulting system does not make use of the problem bit. While *Hybrid Monitoring* uses the problem bit for deciding if the number of failure events is definitely zero or at least one (Algorithm 1, lines 23 and 24), the passive system lacks of this information. In consequence, the passive system is executing the waiting time analysis (Algorithm 2) for all packets received.

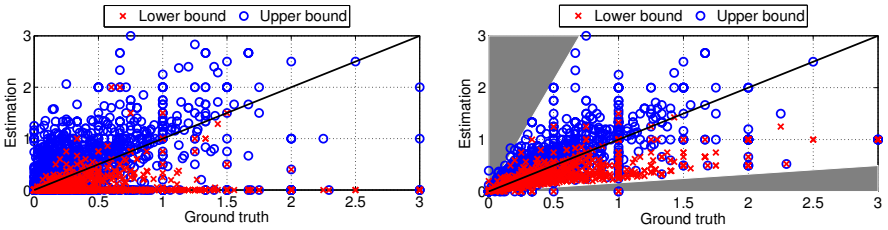
In the following, we compare obtained upper and lower bounds on the number of failure events $P_c^{u,l}(k)$ with ground truth. The relationship between estimated bounds and the actual number of failure events in experiment A) is shown in Figure 4. Shown results have been aggregated based on information from a connection table that has been obtained from ground truth. Each entry of this table corresponds to a connection between a child node and a parent node, each entry of the connection table is represented by a circle and a cross. Shown values are calculated by dividing the number of failure events by the number of packets that were successfully submitted during the corresponding time interval. Estimated values are plotted over ground truth, ideally all data points should be located on the diagonal line that represents the identity function.

First of all, we can see that worst-case and best-case estimations are separated by the identity function in both figures Figure 4(a) and Figure 4(b). However, solutions obtained from the passive system are significantly higher or lower, respectively, than ground truth in a large number of cases.

This observation can be further quantified using Pearson's correlation coefficient. In Figure 5, we can see that the correlation between estimated results and ground truth grows almost linearly with the fraction of packets for which problem bit information is used during the estimation process (Algorithm 1 and Algorithm 3). Packets are randomly selected, the experiment is repeated 30 times for each fraction. Shaded areas denote the range of results obtained. While we obtain values $\rho_{\text{upper}} = 0.92$ and $\rho_{\text{lower}} = 0.86$ when comparing ground truth with results from the *Hybrid Monitoring* system, significantly lower values $\rho_{\text{upper}} = 0.75$ and $\rho_{\text{lower}} = 0.56$ are retrieved when correlating ground truth with results from the completely passive system.

The gain of using the problem bit in *Hybrid Monitoring* can be further highlighted when using data from both systems as input for an anticipated alarming application. Here, we consider a runtime filter that triggers an action, *e.g.*, informs a network operator, if the number of failure events that occurred at a node within a moving time window of $\Delta$ length exceeds a pre-defined threshold value.

Raw results of generating one aggregated value per node every $\Delta := 30$ min are shown in Figure 6. Shaded areas denote the gain of *Hybrid Monitoring* compared to the completely passive system. Here, we can again see results of the completely passive system being significantly too high or too low, respectively, in many cases. In consequence, the performance




(a) Completely passive system

(b) Nodes transmit problem bit

Figure 4. Mean number of failure events per packet. Shown results have been aggregated based on information from a connection table that has been obtained from ground truth.

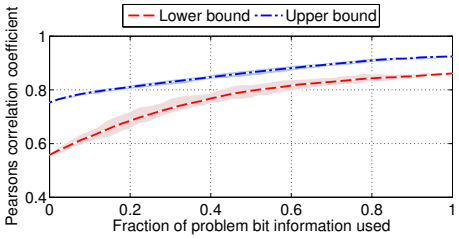



Figure 5. Correlation between the estimated mean number of failure events per packet and ground truth.

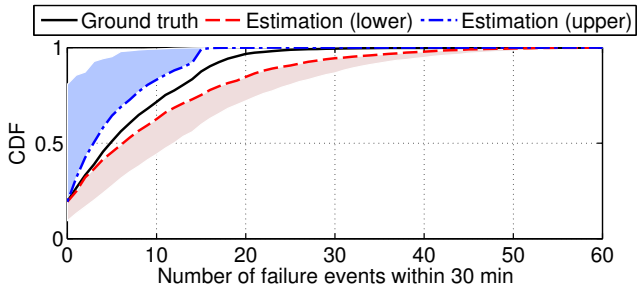



Figure 6. Sum of failure events in 30 min long intervals. Shaded areas denote the gain in estimation accuracy by using *Hybrid Monitoring* instead of a completely passive system.

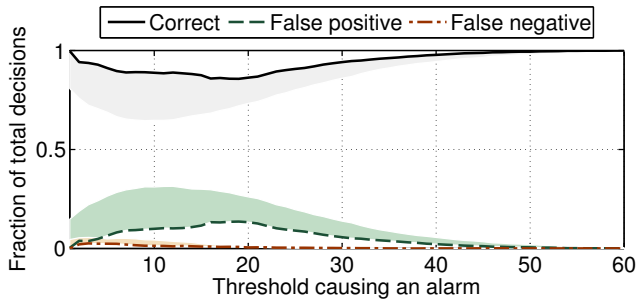



Figure 7. Evaluation of runtime monitoring performance. The triggering of an alarm is based on a pre-defined threshold value that is alternated up to 60.

of the final alarming application is also significantly lower when compared to the performance of *Hybrid Monitoring*. The fraction of correct decisions, *i.e.*, the inferred decision from estimated values is correct, false positives, *i.e.*, an alarm despite the correct value being below the set threshold, and false negative, *i.e.*, missed alarms, are shown in Figure 7. Again, shaded areas denote the improvement by adding one bit of extra information per packet.

### D. Runtime Monitoring

Results obtained from all four testbed experiments A) to D) are used to evaluate the quality of decisions made in the context of an runtime health monitoring scenario. Results for two configurations of a fixed evaluation interval $\Delta$ are shown in Table III and Table IV. Decisions are made based on the estimated upper bound of failure events that occurred within $\Delta$. Values obtained when basing decisions on estimated lower bounds are comparably well.

Decisions made are correct in $\geq 90.7\%$ of all cases. The quality of decisions made is the better for the more packets for which multi-hop network tomography could return results. Ranking testbed experiments either by the fraction of correct decisions made or the fraction of packets for which additional information could be reconstructed (see Table II) yields the exactly same order. This behavior is to be expected given that only packets that passed multi-hop network tomography are used as input for estimating the number of failure events occurred within $\Delta$.

Table III. RUNTIME MONITORING, $\Delta = 10$ min, $\leq 20$ failure events

| | # Decisions | Mean correct | Min. correct | Max. correct |
|---|---|---|---|---|
| A) | 9,841 | 93.5% | 84.4% | 99.7% |
| B) | 2,084 | 91.6% | 80.8% | 99.8% |
| C) | 1,485 | 90.7% | 84.0% | 99.8% |
| D) | 4,112 | 92.4% | 80.3% | 99.8% |

Table IV. RUNTIME MONITORING, $\Delta = 30$ min, $\leq 60$ failure events

| | # Decisions | Mean correct | Min. correct | Max. correct |
|---|---|---|---|---|
| A) | 3,319 | 94.1% | 85.7% | 99.9% |
| B) | 697 | 92.5% | 79.9% | 100.0% |
| C) | 505 | 91.0% | 82.7% | 99.4% |
| D) | 1,393 | 93.2% | 81.7% | 99.9% |

## VIII. BROADER APPLICABILITY AND LIMITATIONS

The presented approach is based on the assumption that certain events inside the network contribute a measurable, additional delay to the end-to-end delay of waiting packets. For example, Dozer [1] waits for approximately 30 sec before starting a new transmission attempt. A measurable penalty is potentially also added by other slotted approaches, *e.g.*, slotted programming [7], that may be set to postpone every retransmission attempt to the next slot. Here, it is to expect that slot lengths of a few seconds or even less are sufficient in combination with an accurate enough time-stamping mechanism. In a broader scope, existing data collection protocols may only require small modifications in order to increase the delay penalty that is added after a transmission failure. For instance, one possibility to adapt CTP [8] would be to increase the firing interval of the retransmission timer.

Apart from requiring transmission failures to cause a measurable delay, a system must also conform to the formal model of multi-hop network tomography [13]. Exemplary systems that have been verified to be conforming to this model are CTP Noe on top of LPL and Dozer. Systems that are not compatible to multi-hop network tomography are for instance protocols that randomly select the receiver of every packet.

Likewise, multi-hop network tomography and in consequence the presented approach in this paper are also not applicable to systems that use flooding as their communication primitive, *e.g.*, the low-power wireless bus [6].

## IX. Conclusions

We presented *Hybrid Monitoring*, a novel health monitoring system that complements a passive health monitoring method, *i.e.*, the analysis of passively reconstructed timing information, with a minimally active component. Adding one bit of extra information to every packet is already sufficient for mitigating the uncertainties that have the largest impact on the accuracy of the passive health monitoring method. As a result, the number of failure events that occurred between the generation of two subsequent packets can be estimated with a high confidence. Based on experiments on testbeds of up to 96 nodes, we find the amount of correct decisions made by a runtime monitoring application to be significantly improved when one more bit of information is available to the estimation process.

**Acknowledgements.** We want to thank the anonymous reviewers for their valuable feedback that helped us to improve this paper. Modifications needed to run Dozer on Tmote Sky nodes were provided by Marco Zimmerling. The work presented was supported by NCCR-MICS under SNSF grant #5005-67322, the Swiss Confederation and Nano-Tera.ch.

## References

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

[2] Q. Cao, T. Abdelzaher, J. Stankovic, K. Whitehouse, and L. Luo, "Declarative tracepoints: A programmable and application independent debugging system for wireless sensor networks," in *Proc. 6th ACM Conf. on Embedded Networked Sensor Systems (SenSys '08)*, 2008.

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

[4] B. Chen, G. Peterson, G. Mainland, and M. Welsh, "LiveNet: Using passive monitoring to reconstruct sensor network dynamics," *Distributed Computing in Sensor Systems*, vol. 5067, pp. 79–98, 2008.

[5] Y.-H. Chiang, M. Keller, R. Lim, P. Huang, and J. Beutel, "Lightweight network health monitoring," in *Proc. 11th Conf. on Information Processing in Sensor Networks (IPSN '12)*, 2012.

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

[7] R. Flury and R. Wattenhofer, "Slotted programming for sensor networks," in *Proc. 9th Conf. on Information Processing in Sensor Networks (IPSN '10)*, 2010.

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

[9] S. Guo, Z. Zhong, and T. He, "FIND: Faulty node detection for wireless sensor networks," in *Proc. 7th ACM Conf. on Embedded Networked Sensor Systems (SenSys '09)*, 2009.

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

[11] A. Hasler, I. Talzi, J. Beutel, C. Tschudin, and S. Gruber, "Wireless sensor networks in permafrost research - concept, requirements, implementation and challenges," in *Proc. 9th Conf. on Permafrost (NICOP '08)*, 2008.

[12] M. Keller, J. Beutel, A. Meier, R. Lim, and L. Thiele, "Learning from sensor network data," in *Proc. 7th ACM Conf. on Embedded Networked Sensor Systems (SenSys '09)*, 2009, pp. 383–384.

[13] M. Keller, J. Beutel, and L. Thiele, "How was your journey? Uncovering routing dynamics in deployed sensor networks with multi-hop network tomography," in *Proc. 10th ACM Conf. on Embedded Networked Sensor Systems (SenSys '12)*, 2012.

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

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

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire tinyos applications," in *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, 2003.

[17] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proc. 12th Conf. on Information Processing in Sensor Networks (IPSN '13)*, 2013.

[18] K. Liu, Q. Ma, X. Zhao, and Y. Liu, "Self-diagnosis for large scale wireless sensor networks," in *Proc. 30th IEEE Conf. on Computer Communications (INFOCOM '11)*, 2011.

[19] Y. Liu, K. Liu, and M. Li, "Passive diagnosis for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1132–1144, 2010.

[20] L. Luo, T. He, G. Zhou, L. Gu, T. F. Abdelzaher, and J. A. Stankovic, "Achieving repeatability of asynchronous events in wireless sensor networks with EnviroLog," in *Proc. 25th IEEE Conf. on Computer Communications (INFOCOM '06)*, 2006.

[21] Q. Ma, K. Liu, X. Miao, and Y. Liu, "Sherlock is around: Detecting network failures with local evidence fusion," in *Proc. 31st IEEE Conf. on Computer Communications (INFOCOM '12)*, 2012, pp. 792–800.

[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 *Proc. 7th ACM Conf. on Embedded Networked Sensor Systems (SenSys '09)*, 2009.

[23] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st Conf. Local Computer Networks (LCN '06)*, 2006.

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

[25] K. Römer and J. Ma, "PDA: passive distributed assertions for sensor networks," in *Proc. 10th ACM/IEEE Conf. on Information Processing in Sensor Networks (IPSN '09)*, 2009.

[26] S. Rost and H. Balakrishnan, "Memento: A health monitoring system for wireless sensor networks," in *Proc. 3rd Annual Conf. on Sensor and Ad Hoc Communications and Networks (SECON '06)*, 2006.

[27] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, 2004.

[28] J. Yang, M. L. Soffa, L. Selavo, and K. Whitehouse, "Clairvoyant: A comprehensive source-level debugger for wireless sensor networks," in *Proc. 5th ACM Conf. on Embedded Networked Sensor Systems (SenSys '07)*, 2007, pp. 189–203.

[29] Y. Zhao, R. Govindan, and D. Estrin, "Residual energy scan for monitoring sensor networks," in *Proc. Conf. on Wireless Communications and Networking (WCNC '02)*, 2002.