Dwarf: Delay-aWAre Robust Forwarding for Energy-Constrained Wireless Sensor Networks

Mario Strasser[1], Andreas Meier[1], Koen Langendoen[2], and Philipp Blum[3]

[1] ETH Zurich, Switzerland (equally contributing authors)
[2] Delft University of Technology, The Netherlands
[3] Siemens Building Technologies, Switzerland

Abstract. With the field of wireless sensor networks rapidly maturing, the focus shifts from “easy” deployments, like remote monitoring, to more difficult domains where applications impose strict, real-time constraints on performance. One such class of applications is safety critical systems, like fire and burglar alarms, where events detected by sensor nodes have to be reported reliably and timely to a sink node. A complicating factor is that systems must operate for years without manual intervention, which puts very strong demands on the energy efficiency of protocols running on current sensor-node platforms.
Since we are not aware of a solution that meets all requirements of safety-critical systems, i.e. provides reliable data delivery *and* low latency *and* low energy consumption, we present Dwarf, an energy-efficient, robust and dependable forwarding algorithm. The core idea is to use unicast-based partial flooding along with a delay-aware node selection strategy. Our analysis and extensive simulations of real-world scenarios show that Dwarf tolerates large fractions of link and node failures, yet is energy efficient enough to allow for an operational lifetime of several years.

1 Introduction

The state of the art in Wireless Sensor Networks (WSN) is rapidly changing. From the Smart Dust vision in 1999 [4], through the early Great Duck Island experiment with first-generation hardware in 2002 [9], to a host of (pilot) deployments in operation today [1, 11, 17]. Although the application domains vary, these deployments typically fall into the class of remote monitoring, where rather soft constraints on performance (e.g., latency, throughput, and lifetime) allow for straightforward engineering solutions. The experience gained with these pilots is being incorporated into a second-generation software that is better tuned, more robust, and offers the potential for enlarging the scope to more demanding applications.

Using WSN technology for implementing safety-critical applications such as fire and burglar alarm systems is a big challenge because of the real-time constraints imposed by their users. Typically, alarms detected by sensor nodes have to be reported reliably and within a few seconds to at least one sink node, even in case that some of the nodes and communication links fail. Additionally, safety-critical applications are required to observe the status of the network and report node failures within a specified time. A complicating factor is that maintenance costs have to be very low to make an application economically feasible. This requires energy-efficient operation of the sensor network, because batteries should not be replaced more often than once every two to three years.

With current generation sensor node hardware built out of COTS components, the radio consumes the most power, and the above lifetime requirement translates into a duty cycle of well below 1%. This, in turn, limits the data rate to about 1 message per second and leaves little room for re-transmissions in a multi-hop scenario. Multi-path routing proposed for ad-hoc networks is an alternative way of handling link and node failures, but solutions either compromise on latency or rely on broadcast for efficiency. A broadcast-based approach, however, is consuming way too much energy as it implies that nodes listen to *all* neighboring traffic including regular status-update messages for monitoring system integrity. Furthermore, broadcast causes channel contention problems and introduces synchronization overhead. The need to avoid broadcast is also a reason that standard data gathering algorithms for WSN networks, like TAG [8] and Synopsis Diffusion [12], cannot be used.

To jointly address the three fundamental requirements associated with safety-critical applications (reliable data delivery, low latency, low energy consumption) we advocate an integrated approach that cuts across the individual MAC, routing, and transport layers, to arrive at a working solution for commodity sensor nodes in use today. To this end, we present Dwarf, a Delay-aWAre Robust Forwarding algorithm that is based on the following observations and assumptions:

a) One of the most robust, yet simple forwarding algorithms is flooding because it ensures that a message will eventually reach its destination as long as the network remains connected.
b) Traditional flooding is very expensive (with regard to energy consumption and transfer costs) and does not consider the message delivery time at all.
c) Nodes duty-cycle their radio to increase network lifetime and spend most of their time in sleep mode. Also, to minimize overheads and reduce protocol complexity, nodes do not synchronize globally (as in TDMA-based systems [5, 7]) and wake up independently of each other.
d) The node-to-sink notification time is determined by the, relatively long, sleep periods of the destination nodes along the path. What is more, the transfer time of a message is much smaller than the time between two wake-ups (10 ms vs. 1000 ms in our alarm-system scenario, see Section 3).

The fundamental idea of Dwarf is to perform a unicast-based partial flooding towards the sink in combination with a (greedy) delay-aware node selection strategy to overcome the drawbacks mentioned above. More precisely, the number of neighbors k to which an alarm is forwarded determines the degree of introduced redundancy, thus making the algorithm more robust at the expense of an increase in the number of messages and the associated complexity in handling peak loads (e.g., collisions). The selection of the destination nodes according to their wake-up time and relative position aims at reducing the overall alarm notification time. That is, neighbors that wake up first and are closer to a sink are favored over nodes that wake up later or are not on the shortest path towards the sink. In order to maintain system integrity, status messages are exchanged between neighboring nodes on a regular basis. This enables to detect (temporary) link failures as well as (permanent) node failures, which must be reported to the sink so operators can take appropriate action (e.g., replace batteries). The status messages are also used to account for clock drifts in individual nodes and keep an up-to-date view on neighboring nodes' wake-up times.

Summarized, the main contributions of this paper are threefold:

1. It presents a novel, integrated algorithm (Dwarf) for the robust and timely delivery of alarm messages at the sink node in a energy-constrained multi-hop sensor network.
2. It provides a theoretical analysis of fundamental performance guarantees that Dwarf can achieve.
3. It shows through a set of detailed simulations that the Dwarf algorithm meets the requirements of an alarm system taken in a real-world scenario.

The remainder of this paper is organized as follows. Section 2 discusses related work, followed by a list of requirements and assumptions in Section 3, which set the boundary for the actual Dwarf algorithm presented in Section 4. The evaluation in Section 5 includes a formal analysis as well as an extensive set of simulations. Finally, Section 6 concludes the paper.

2 Related Work

The need for energy-efficient operation is at the core of WSN research and has received considerable attention. At the MAC layer, the excess channel capacity can be exploited by duty-cycling the radio; at the routing layer, the redundancy in the number of nodes can be exploited by rotating on/off duties. The latter approach, however, is not an option in an alarm system where all nodes are essential. WSN-specific MAC protocols generally save energy at the expense of an increase in (multi-hop) latency, but differ in their exact trade-off [6]. The class of Low-Power Listening protocols, such as B-MAC [13] or more sophisticated approaches like SCP-MAC [21] and WiseMAC [2] fit best because of the low power consumption doing idle listening and explicit control of the length of the interval between wake-ups. SCP-MAC is based on a global synchronization of the wake-up slots and hence introduces overhearing of the regular status-update messages. WiseMAC on the other hand is based on asynchronous wake-up slots, naturally minimizing message overhearing. Dwarf will be working with a slightly enhanced version of WiseMAC that is discussed in detail in Section 3.2.

Another corner stone of WSN research is the need to handle errors in the wireless channel (short-term packet loss, long-term link failures) and the possibilities of node failures. Surprisingly little research has been done on providing reliable end-to-end message delivery. The transport protocols that do address link failures (e.g., ESRT [15], RMST [16], and PSFQ [18]) include techniques like retransmissions and path diversity to overcome the errors on individual links, but do not provide end-to-end reliability because of the high costs and long (multi-hop) latencies involved. At best, advanced protocols like MMSPEED [3] provide probabilistic bounds on end-to-end delivery ratios[4]. Therefore, an application should always be prepared to tolerate (residual) packet loss.

An effective, but expensive, approach to handle communication errors is to use flooding. Quite often network-wide redundancy is not needed and partial flooding suffices. For example, the GRAB protocol [20] uses a credit mechanism to specify how many additional hops may be made to reach the destination, effectively creating a "wide-path". GRAB requires the set-up of a gradient field towards the destination, hence, is only applicable to a few, popular destinations like the sink(s) in an alarm system. The DFRF framework [10] generalizes this idea and allows for easy creation of tailor-made partial-flooding protocols. Unfortunately, DFRF does not integrate well with the MAC layer below making it difficult to control latency and energy consumption.

An issue specific to safety-critical systems is the importance of detecting failed nodes, which compromise the integrity of the system. A straightforward solution is to make use of heart-beat style failure detectors where nodes periodically send out a message notifying neighbors of their status. Wang and Kuo extend this idea to a two-phase gossiping protocol suited for ad-hoc networks [19]. Although very robust, information propagates slowly and at high cost. Recent work by Rost and Balakrishnan shows that more-advanced failure detectors help in reducing the message overhead [14], but latency remains an issue.

The lack of an integrated protocol that provides fast and robust delivery of alarm messages in a multi-hop WSN, combined with a low-overhead failure detector prompted us to design Dwarf.

3 Requirements and Assumptions

The design of the Dwarf forwarding and failure detection algorithms were driven, on the one hand, by the requirements from the safety-critical application that it should support and, on the other hand, by the functionality and configurability that the MAC layer below provides. As always in a design process, the boundary conditions were not crystal clear and subject to change. Hence, we had to make some assumptions, which are also detailed in this section.

3.1 Alarm-system Scenario

The concrete application scenario for which Dwarf was designed is a distributed indoor wireless alarm system. Each sensor node consists of a micro controller (ATMEL ATMega128), a communication unit (CC1000 transceiver), a power supply (2 AA batteries) and a sensor for detecting a specific alarm condition. All nodes are manually deployed at fixed locations in a building as with ordinary,

[4] MMSPEED in fact provides QoS guarantees on reliability and latency, but ignores energy consumption, rendering it unsuitable for our purposes.

wired sensors. In addition, there is at least one mains-powered sink node that is connected to a central control station. Domain specific regulations require that an alarm raised by a sensor is reported at the control station (sink) **within 10 seconds**, which leaves little room for per-hop delays in typical office buildings with long corridors and one control station per floor.

A second domain specific requirement is that failing nodes must be reported **within 5 minutes** at the control station. Since link errors caused by environmental interference are much more likely to occur than a node running out of energy or failing for some other reason, we assume that unreachable nodes, although technically still alive, must also be reported. This requires a periodic status observation of the nodes, which must send out at least one message per 5 minutes. The control station needs to be positively informed about the aliveness of each node, necessitating a collective, multi-hop forwarding scheme.

A failed node must be replaced, which is a costly operation due to the need of calling in a qualified technician asserting the integrity of the complete alarm system. Thus, it makes sense to replace the batteries of all nodes as soon as the first one runs out of energy. However, to reduce operational costs, such a grand replacement procedure should not occur more often than every two to three years. This consideration requires Dwarf to minimize and to equalize the power consumption for all sensor nodes.

3.2 MAC Protocol

A first requirement on the MAC protocol is that the effective duty-cycle must be well below 1 %, which follows from the minimum lifetime (2 years), the power consumption of the target radio in use (15 mA) and the battery capacity (2800 mAh). A further constraint follows from the maximum end-to-end latency of 10 s and the assumption that topologies with a depth of at least 5 hops must be supported, together bounding the maximum wake-up interval (T_w) to at most 2 s (careful staggering wake-up periods a la DMAC [7] would allow for even longer intervals, but runs the risk of excessive delays in the case of link errors).

A 1 % duty cycle and a 2 s wake-up interval allow for an active period of about 20 ms, which is enough to perform a carrier sense operation taking approximately 2.5 ms on a CC1000 radio [13]. Recall that Dwarf is based on partial flooding to overcome link and node failures, and needs to contact several nodes per hop. To avoid accumulating delays in doing so, it is essential that a MAC protocol provides an interface which allows for querying the wake-up schedule of neighboring nodes. Note further that, although most MAC protocols do not provide such functionality, only minor modifications are required to enhance them.

To allow for easy deployment and a high resilience to errors, a MAC protocol that requires no time synchronization between nodes is strongly preferred. This limits the choice to the class of low-power listening protocols, in which a sender pretends each message with a preamble that is slightly longer than T_w to ensure that the intended receiver will sense a busy channel and listen in on the complete transmission. These long preambles increase latency, but a sophisticated protocol like WiseMAC, which learns the wake-up schedules of its neighbors, can still use ordinary (short) preambles in most cases; the exact length of the preamble depends on the clock drift and the time passed since the last message was exchanged with the intended receiver. Thus, in order to use WiseMAC effectively, Dwarf should ensure that neighboring nodes are periodically contacted.

3.3 Definitions

Throughout this paper, we represent the sensor network by the graph $G := (V, E)$ consisting of the set of sink nodes $S \subset V$, the set of sensor nodes $V \setminus S$, and the set of edges E. All communication links are considered to be bidirectional and two nodes $u, v \in V$ can directly communicate with each other (i.e., are neighbors) if and only if $\{u, v\} \in E$. Furthermore, all sensor nodes are organized in rings according to their distance to the nearest sink; nodes with the same distance are said to be in the same ring:

Definition 1. *Let $d(u, v)$ be the distance (i.e., the length of the shortest path) between two nodes u and v. A node u is said to be in the i-th ring, or alternatively to be on level $l(u) = i$ with respect to the set of sink nodes S if and only if $\min\{d(u, s) : s \in S\} = i$. The set $R_i := \{u : u \in V \wedge l(u) = i\}$ contains all nodes on level i and the maximal level is denoted by $L := \max\{l(u) : u \in V\}$.*

Based on Definition 1, the neighbors of a node are divided into parents, peers, and children (see Figure 1(a)):

Definition 2. *We denote the set $N_u^- := \{v : \{u, v\} \in E \wedge l(v) = l(u) - 1\}$ as the parents of a node u, the set $N_u^0 := \{v : \{u, v\} \in E \wedge l(v) = l(u)\}$ as its peers, and the set $N_u^+ := \{v : \{u, v\} \in E \wedge l(v) = l(u) + 1\}$ as its children, respectively.*

As already mentioned, we assume that all nodes but the sinks sleep most of the time in order to save energy, and only wake up periodically. The wake-up period of node u is denoted by $T_w(u)$, its wake-up times by $\tau_{u,i}$.

Definition 3. *Let $\tau_{u,i}$, $u \in V \setminus S$ and $i \in \{0, 1, 2, \ldots\}$ be the wake-up times of node u, such that $\tau_{u,i+1} = \tau_{u,i} + T_w(u)$ and $0 < T_w(u) < \infty$. The duration until the upcoming wake-up time relative to the current time t is denoted by $\tau(u, t) := \min\{\tau_{u,i} : \tau_{u,i} > t\} - t$. Sink nodes are assumed to be always listening hence, $\forall s \in S : \tau(s, u) := 0$.*

Finally, we set $T_w := \max\{T_w(u) : u \in V \setminus S\}$, assume that there exists a constant upper bound $t_m \in O(1)$ on the transmission time of any message m, and use k to denote the (constant) upper bound on the number of neighbors to which a message is forwarded.

4 Algorithms

In this section, we present the proposed Dwarf algorithm which consists of two main tasks: alarm forwarding and node or link status observation.

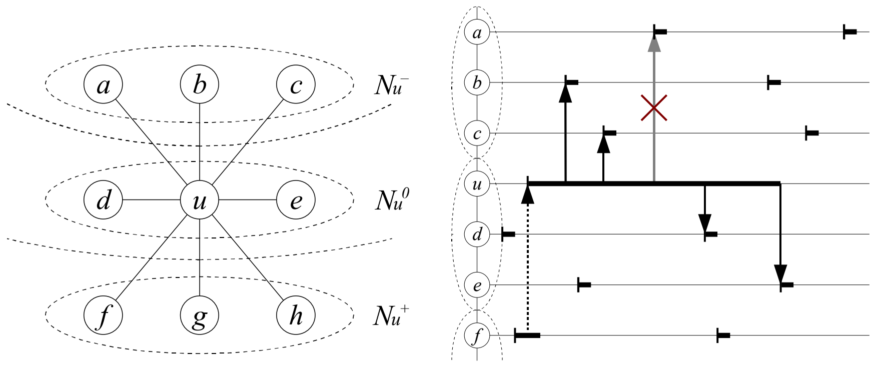


Fig. 1. On the left, the local neighborhood of node u with parents $N_u^- := \{a, b, c\}$, peers $N_u^0 := \{d, e\}$ and children $N_u^+ := \{f, g, h\}$ is depicted. A corresponding forwarding example with $k = 4$ and a failing transmission to node a is shown on the right.

4.1 Alarm Forwarding

When a node detects an alarm, it creates an appropriate alarm message m and calls the function *forwardAlarm()* which, in turn, forwards it to $\min(k, |N_u^-| + |N_u^0|)$ parents and peers (see Algorithm 1). To this end, a set of parent candidates $\widetilde{N}_{u,m}^-$ as well as a set of peer candidates $\widetilde{N}_{u,m}^0$ is maintained for each message. The actual selection of the nodes to which a message is forwarded is performed by the function *getNextHop()*. The function determines the parent candidate that wakes up next and, after removing it from the candidate set, returns it as the current destination. If there are no more parents to chose from (i.e., $\widetilde{N}_{u,m}^- = \emptyset$), the peer that wakes up next is returned instead. Once both sets are empty, they are reinitialized with $N_u^- \setminus \widehat{N}$ and $N_u^0 \setminus \widehat{N}$, respectively, with $\widehat{N}$ being the set of neighbors that successfully received the message (or forwarded it in the first place). In order to keep the induced traffic as low as possible and because there would be no additional gain otherwise, a node on level one aborts the forwarding process as soon as it has successfully delivered the message to at least one sink.

Upon reception of an alarm message m, a node first verifies that the message has not already been forwarded (i.e., $m \notin H$). New messages are appended[5] to the message history H and forwarded in the same manner as a newly generated alarm using the function *forwardAlarm()*.

Should an alarm message be dropped because of a send, receive, or transmission failure, it is retransmitted up to r_a times, resulting in maximal $k + r_a$ transmissions per message m. For each retransmission, however, a new destination is selected with *getNextHop()*, thus ensuring that retransmissions are also forwarded in the fastest way possible.

[5] Of course not a whole message m has to be stored but only an unique identifier such as the tuple (alarm originator, sequence number).

Algorithm 1 Alarm forwarding for node u

```
var $H \leftarrow \emptyset$

function initCandidates($m$)
    $\widetilde{N}^-_{u,m} \leftarrow N^-_u \setminus \widehat{N}_m$
    $\widetilde{N}^0_{u,m} \leftarrow N^0_u \setminus \widehat{N}_m$
end function

function getNextHop($m$)
    if $\widetilde{N}^-_{u,m} = \emptyset$ and $\widetilde{N}^0_{u,m} = \emptyset$ then
        initCandidates($m$)
    end if
    if $\widetilde{N}^-_{u,m} \neq \emptyset$ then
        select $v \in \widetilde{N}^-_{u,m}$ such that
          $\tau(v) = \min\{\tau(w) : w \in \widetilde{N}^-_{u,m}\}$
        $\widetilde{N}^-_{u,m} \leftarrow \widetilde{N}^-_{u,m} \setminus \{v\}$
        return $v$
    else if $\widetilde{N}^0_{u,m} \neq \emptyset$ then
        select $v \in \widetilde{N}^0_{u,m}$ such that
          $\tau(v) = \min\{\tau(w) : w \in \widetilde{N}^0_{u,m}\}$
        $\widetilde{N}^0_{u,m} \leftarrow \widetilde{N}^0_{u,m} \setminus \{v\}$
        return $v$
    else
        return $\perp$
    end if
end function

function forwardAlarm($m$)
    $H \leftarrow H \cup \{m\}$
    initCandidates($m$)
    $r_m \leftarrow 0$
    $i_m \leftarrow 1$
    $v \leftarrow$ getNextHop($m$)
    send alarm message $m$ to node $v$
end function

upon acknowledgment of alarm message $m$ sent to $w$
    if $i_m < \min(k, |N^-_u| + |N^0_u|)$ and $w \notin S$ then
        $\widehat{N}_m \leftarrow \widehat{N}_m \cup \{w\}$
        $i_m \leftarrow i_m + 1$
        $v \leftarrow$ getNextHop($m$)
        send alarm message $m$ to node $v$
    end if
end upon

upon drop of alarm message $m$ sent to $w$
    if $r_m < r_a$ then
        $r_m \leftarrow r_m + 1$
        $v \leftarrow$ getNextHop($m$)
        send alarm message $m$ to node $v$
    else if $i_m < \min(k, |N^-_u| + |N^0_u|)$ then
        $i_m \leftarrow i_m + 1$
        $v \leftarrow$ getNextHop($m$)
        send alarm message $m$ to node $v$
    end if
end upon

upon reception of alarm message $m$ from $v$
    if $m \notin H$ then
        $\widehat{N}_m \leftarrow \{v\}$
        forwardAlarm($m$)
    else if $v \notin \widehat{N}_m$ then
        $\widehat{N}_m \leftarrow \widehat{N}_m \cup \{v\}$
    end if
end upon

upon detection of an alarm
    create alarm message $m$
    $\widehat{N}_m \leftarrow \{\}$
    forwardAlarm($m$)
end upon
```

A forwarding example for the scenario depicted in Figure 1(a) and $k = 4$ is presented in Figure 1(b). After receiving an alarm message, node u forwards the message to the parents b, and c as well as to peers d and e, assuming that the transmission to parent a failed.

4.2 Node Status Observation

The purpose of the status messages is twofold: On the one hand, they are required in order to detect node or link failures, and on the other hand, they keep the mutual knowledge of neighboring nodes regarding their wake-up times up to date. Therefore, nodes send a status message to a peer as well as to a parent in a round robin fashion every interval T_s (see Algorithm 2). In the scenario depicted in Figure 1(a), for instance, node u would first send its status message to a and d, after interval T_s to b and e, then to c and d etc. Each status message contains a list of nodes which are known to be up and running; or put differently, present a node's (limited) view of the network. Whenever a node receives a status message, the included node status list X' is merged with its own list X (i.e., $X := X \dot{\cup} X'$);

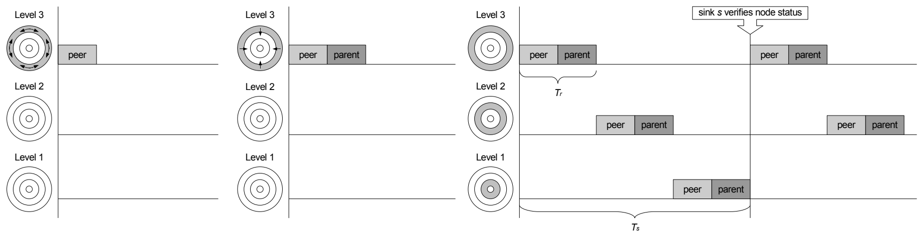


Fig. 2. Each node periodically sends a status message to first a peer and then a parent, containing a list of all running nodes it knows about. When a node receives such a status message it appends the mentioned nodes to its own status list. Nodes in the outer rings send first such that the sink will eventually receive the complete status of the network.

the list is cleared every interval T_s once it has been sent to a parent. Should a message to a peer (parent) be dropped, the next peer (parent) is chosen and the message retransmitted up to r_s times.

By having the nodes in the outer rings of the network send first (see Figure 2), node states are disseminated — and thereby gradually updated — in form of waves towards the sink which eventually receives a complete list of all running nodes every interval T_s[6]. Therefore, the status-update interval T_s is divided into L sub-intervals of length $T_r := T_s/L$, which, in turn, are halved into a peer and a parent slot. Furthermore, each node in the i-th ring is associated with i-th sub-interval and sends its status message to the selected peer and parent in the corresponding peer or parent slot, respectively. By scheduling the peer slot before the parent slot, some additional reliability is introduced as a node's status list is now forwarded by one of its peers as well. The exact sending time within a slot is independently and uniform-randomly chosen by each node and, if possible, rounded to the nearest regular wake-up time. It might be worth mentioning that, due to the relatively large delay between two subsequent parent slots within a wave, a loose time synchronization (i.e., in the order of seconds) is sufficient for the proposed algorithm.

4.3 Startup

The required knowledge of a node u consists of: (i) its level $l(u)$, (ii) its one-hop neighbors and their levels (i.e., N_u^-, N_u^0, and N_u^+), and (iii) the starting time for the status waves. The information regarding level and neighbors can easily be obtained as part of an enhanced neighborhood discovery algorithm that works as follows:

1. Initially, the level of all nodes but the sinks is set to infinite and they have no information about their neighborhood.

[6] If there is more than one sink, each of them receives only a partial list. However, we assume that all sink nodes are connected with each other, either directly or via a central control station, and thus can easily obtain all partial lists.

Algorithm 2 Status message exchange for node u

```
var next_parent ← 0
var next_peer ← 0
var $X \leftarrow \{u\}$
var $t_u \leftarrow$ uniform randomly out of $[0, T_r)$
start peer timer for $\Delta t = t_u + (L - l(u))T_r$
start parent timer for $\Delta t = t_u + ((L - l(u)) + \frac{1}{2})T_r$

function NEXTNODETOPROBE
    if use_parent = true then
        if next_parent < $|N_u^-| - 1$ then
            next_parent ← next_parent + 1
        else
            next_parent ← 0
        end if
        return $N_u^-$[next_parent]
    else
        if next_peer < $|N_u^0| - 1$ then
            next_peer ← next_peer + 1
        else
            next_peer ← 0
        end if
        return $N_u^0$[next_peer]
    end if
end function

upon timeout of peer timer
    use_parent ← false
    $r_m \leftarrow 0$
    $v \leftarrow$ NEXTNODETOPROBE()
    create status message $m$
    send status message $m$ to node $v$
    start peer timer for $\Delta t = T_s$
end upon

upon timeout of parent timer
    use_parent ← true
    $r_m \leftarrow 0$
    $v \leftarrow$ NEXTNODETOPROBE()
    create status message $m$
    send status message $m$ to node $v$
    $X \leftarrow \{u\}$
    start parent timer for $\Delta t = T_s$
end upon

upon drop of status message $m$
    if $r_m < r_s$ then
        $r_m \leftarrow r_m + 1$
        $v \leftarrow$ NEXTNODETOPROBE()
        send message $m$ to node $v$
    end if
end upon

upon reception of status message $m$
    merge received status $X'$ with $X$
end upon
```

2. The sinks initiate the algorithm by broadcasting their ID and level.
3. A node u which receives a message from a neighbor v with $l(v) < l(u) - 1$ sets its own level to $l(v) + 1$, updates the parent, peer, and children and (re)broadcasts its new level.

An accurate propagation of the starting time of the status waves could be achieved by flooding a relative starting time which is gradually updated by subtracting the (approximated) transfer times.

5 Evaluation

In this section, we provide an analytical evaluation of the proposed algorithms and present findings of simulations for scenarios derived from real-world data.

5.1 Analytical

In the following, we present proofs for the maximal number of link and node failures that can be tolerated, show an upper bound on the required hop count, and prove that the proposed destination selection algorithm is only a constant factor worse than the optimal solution.

Theorem 1. *The proposed alarm forwarding algorithm (Algorithm 1) can tolerate up to* $t_l := \min(k, \delta) - 1$ *link failures with* $\delta = \min\{|N_u^-| + |N_u^0| : u \in V\}$ *and* k *being the number of forwarding destinations.*

Proof. Let us assume that u is the first node on level $i > 0$ to receive or create an alarm which cannot be forwarded to a node on level $i-1$. Consequently, all links to u's parents must be broken. In addition, for $\geq \min(k, |N_u^0|)$ of u's peers either the links from u to them or from them to their parents must be broken. Per definition, each node has at least one parent. Thus, in total $\geq |N_u^-| + \min(k, |N_u^0|) \geq \min(k, \delta) > t_l$ links must be down, contradicting the assumption that there are at most t_l link failures. The threshold is tight for $k \geq \delta$ as a node u with $|N_u^-| + |N_u^0| = \delta$, which exists per definition, can be isolated from all its parents and peers if we allow $\geq \delta = t_l + 1$ link failures.

Theorem 2. *If at most t_l links fail (see Theorem 1), an alarm initiated by node u on level $l(u)$ will reach the nearest sink after at most $l(u)+\min(t_l, l(u)) \leq 2l(u)$ hops.*

Proof. In order to extend the number of required hops by one, the links to all parents of a node v must be broken. However, as a message is forwarded to $\min(k, |N_v^0|)$ peers, at least $\min(k, |N_v^0|) - (t_l - |N_v^-|) = \min(k + |N_v^-|, |N_v^0| + |N_v^-|) - t_l \geq \min(k, \delta) - t_l = 1$ of them are able to forward it to one of their parents. As a result, the number of required hops can be extended by only one for each level and requires that at least one link is down. The maximal number of additional hops is thus $\min(t_l, l(u))$.

Definition 4. *For a node u, we denote by $|N_u^Z|$ the maximal number of peers such that: (i) for each peer there exists a path to a node on level $l(u)-1$ that is not a parent of u; (ii) on each path, all but the last node are on level $l(u)$; and (iii) all paths are mutually node-disjoint. More precisely, $|N_u^Z| = \max\{|a| \mid a \subseteq N_u^0 \ \wedge \ \forall v \in a : \exists$ path $(v, v_1, v_2, v_3, \ldots, v_m)$ such that $\forall v_i,\ 1 \leq i < m : \ v_i \in R_{l(u)}$ and $v_m \in R_{l(u)-1} \setminus N_u^- \ \wedge \ \forall v, w \in a, v \neq w : \forall i, j : \ v_i \neq w_j\}$*

Theorem 3. *The proposed alarm forwarding algorithm (Algorithm 1) can tolerate up to $t_p := \min(k, \gamma) - 1$ node failures with $\gamma = \min\{|N_u^-| + |N_u^Z| : u \in V\}$ and k being the number of forwarding destinations.*

Proof. Let us assume that u is the first node on level $i > 0$ to receive or create an alarm which cannot be forwarded to a node on level $i-1$. Consequently, all parents of u must have failed. In addition, for $\geq \min(k, |N_u^Z|)$ of u's peers, they, a node on the corresponding node disjoint path, or the corresponding node in the next level must have failed. Thus, in total $\geq |N_u^-| + \min(k, |N_u^Z|) \geq \min(k, \gamma) > t_p$ nodes must be down, contradicting the assumption that there are at most t_p node failures. The threshold is tight for $k \geq \gamma$ as a node u with $|N_u^-| + |N_u^Z| = \gamma$, which exists per definition, can be isolated from all nodes in the next level if we allow $\geq \gamma = t_p + 1$ node failures.

Theorem 4. *If at most t_p nodes fail (see Theorem 3), an alarm initiated by node u on level $l(u)$ will reach the nearest sink after at most $l(u) + \beta t_p$ hops with $\beta = \max\{|N_u^+| : u \in V\}$.*

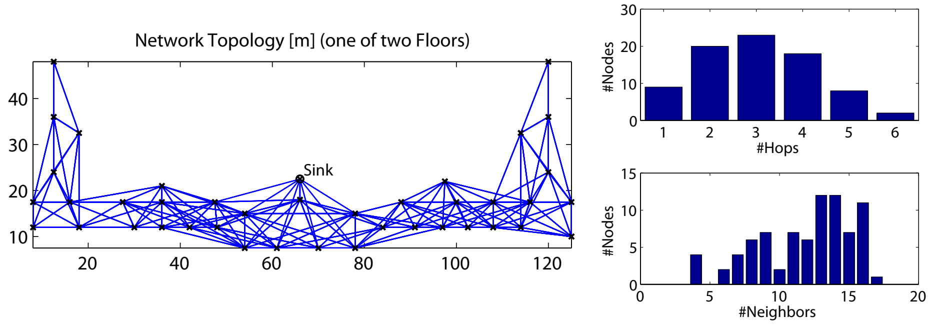


Fig. 3. 80 sensor nodes are positioned according to a real world, but wired deployment; connectivity is based on measured path-loss coefficients.

Proof. Given that at most t_p nodes fail, there exists a path $(u, u_1, u_2, u_3, \ldots, u_m)$ which connects a node u with a node u_m in the next lower level. A node v that fails has at most $|N_v^+|$ children and thus can prevent at most $|N_v^+|$ nodes on level i from forwarding an alarm to level $i-1$. Consequently, after at least $|N_v^+|$ hops in the same level a node with a different parent is reached. As a result, each failed node v can extend the number of required hops by at most $|N_v^+| \leq \beta$ and the maximal number of additional hops is bounded by βt_p.

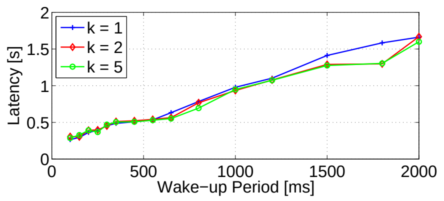
Theorem 5. *The presented (greedy) destination selection algorithm selects a route that is at most* $1 + \frac{T_w}{t_m} \in O(1)$ *times slower than the optimal route.*

Proof. If there are no link failures, an alarm m initiated by node u on level $l(u)$ will reach the nearest sink in time $t_b = l(u)t_m$ in the best case and in time $t_w = l(u)(T_w + t_m)$ in the worst case. Thus, if the algorithm prefers parent v over w because $\tau(v, t_0) = T < \tau(w, t_0) = T + \varepsilon$ we get a worst case ratio of

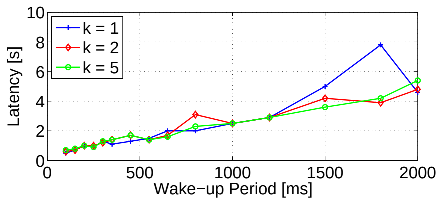
$$c = \frac{T + (l(u) - 1)(T_w + t_m)}{T + \varepsilon + (l(u) - 1)t_m} \leq \frac{T_w + t_m}{t_m} = 1 + \frac{T_w}{t_m}$$

5.2 Simulation

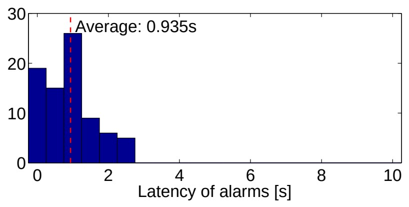
To evaluate Dwarf under appropriate and realistic conditions, we conducted a set of measurements in an existing (wired) real-world alarm system located in a large historic public building comprising 80 sensor nodes and one central control station (sink). In particular, the complete 80x80 path-loss matrix was recorded, capturing the link quality between any pair of nodes. In addition, we enhanced GloMoSim such that it can be directed to work with this recorded data. During simulation, a signal-to-interference-plus-noise-rate (SINR) model is used to compute the bit-error rate for each transmission individually. The packet-error rates are then calculated based on the packet length not assuming any bit-error correction. Figure 3 shows the topology and some basic characteristics in the case without interference; links are shown if the bit-error rate is below 0.1%.



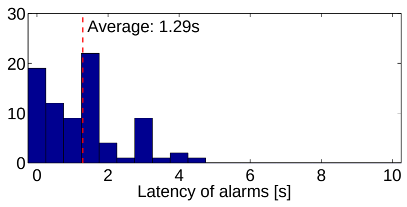
(a) Average latency



(b) 99-quantile latency



(c) $T_w = 1s$, $k = 2$



(d) $T_w = 1.5s$, $k = 2$

Fig. 4. Fire-alarm performance without link failures.

The realistic channel model is complemented with the original GloMoSim implementation of the WiseMAC protocol, enhanced by its authors to include an API for querying the wake-up times of contacted neighbors. This functionality was needed to implement Dwarf's parent selection for rapid alarm forwarding.

In the remainder of this section, we present the results of our simulations with respect to alarm notification time, message complexity, energy consumption, and robustness against link failures.

Alarm Notification Time Figure 4(b) presents the 99-quantile of the latency for different T_w and k in a first experiment without any node and link failures. Note that, since no absolute guarantees can be given in any wireless system, we report the worst case latency of all simulations except the 1% pathological cases. The first observation that can be made is that the maximum (6-hop) notification delay increases linearly with T_w due to the wake-up period dominating the actual message transfer time ($T_w \gg t_m$). In addition to the 99-quantile, the latency distribution of all alarm messages for $k = 2$ and $T_w = \{1s, 1.5s\}$ is depicted in Figures 4(c) and 4(d). This latency distribution shows that the average (3-hop) latency is just $0.94s$ for $T_w = 1s$ and $1.29s$ for $T_w = 1.5s$, which is even smaller than the nodes' wake-up time. This is a consequence of Dwarf's forwarding scheme that selects the next hop according to its level *and* wake-up time. The average hop delay is therefore much smaller than T_w. In contrast to the 99-quantile, the average latency increases less than linear with T_w. This behavior can be explained by the fact that an always listening sink has a much bigger impact on the average latency than on the 99-quantile. Since the outliers become more distinct with an increasing wake-up time, we will use $T_w = 1s$ from now on as the default setting for the remaining experiments.

Table 1. The number of messages Dwarf injects into the network as a function of trigger level and k. For comparison, a network-wide flood generates 479 messages.

Level	k=1	k=2	k=3	k=4	k=5
1	1.00	1.00	1.00	1.00	1.00
2	2.00	4.23	7.51	11.46	19.84
3	3.01	9.25	18.30	27.19	35.81
4	4.02	15.97	26.73	35.51	40.95
5	5.03	22.69	34.56	43.20	48.40
6	6.07	30.68	42.14	49.23	53.29

Figure 4(b) also shows the impact of the parameter k. One can observe that the smaller k, the more outliers occur. This is especially severe for $k = 1$ where such a distinct outlier can be noticed at $T_w = 1.8s$. Choosing $k > 1$ prevents them to occur, since a local jam can be bypassed on another route. Even though $k = 5$ seems to be more stable in terms of small outliers, $k = 2$ already delivers the messages well within the required $10s$.

Message Complexity The parameter k has not only an impact on the latency but also on the number of propagated alarm messages. Table 1 shows the average number of totally generated messages per alarm, depending on k and the level on which the alarm was triggered. For $k = 1$ the number of messages is just slightly larger than the level, showing that messages can usually be forwarded to parents. For $k = 2$ the number of messages increases in the order of 2^{level} up to the 4th level. For $k = 5$, on the other hand, the increase in the number of messages is far below 5^{level}, mainly because messages are not sent backwards to children and due to the topology of real deployments that enforce frequent unifications of alarm messages routed on different paths. As a result, increasing k from 2 to 5 for a level-6 alarm will not even double the number of generated messages. Furthermore, an alarm is considered to be a very rare event, allowing a certain message overhead in order to ensure robust operation.

Energy Consumption There are two main sources of energy consumption: First, the radio must be turned on regularly in order to check for a possible alarm message and second, sending and receiving status messages. The energy consumption of the former increases linearly with the wake-up frequency $1/T_w$, while the latter depends on the total number of sent messages. This number, in turn, heavily depends on the update interval T_s, which was chosen to be 150 seconds in order to ensure that node failures are reported in time ($2T_s \leq 5min$). Figure 5(a) depicts this partitioning of the maximal energy consumption for the usual idle state where no alarms are generated. It shows that the energy consumption of status messages is constant for $T_w \gtrsim 500ms$, but significantly increases for shorter wake-up times due to a more frequent overhearing of status messages. As already mentioned, the targeted duty cycle is required to be

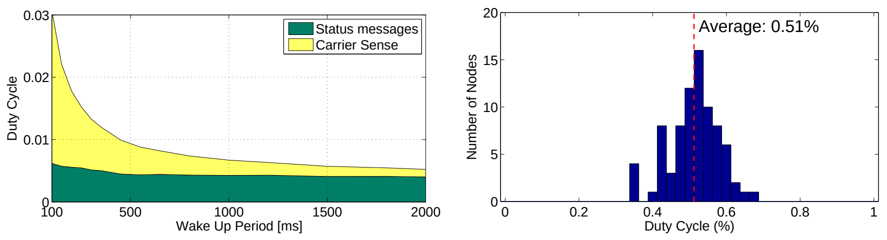


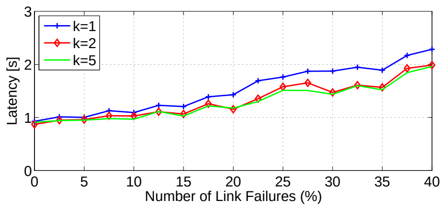
Fig. 5. Worst-case energy consumption for status message exchange and carrier sense. As shown on the right, the node's energy consumption well-balanced.

below 1%, which can be achieved with $T_w \gtrsim 500ms$. However, having a wake-up time of about $1s$ provides some additional flexibility and accounts for additional maintenance tasks as well as network initialization. Finally, Figure 5(b) shows that Dwarf provides the desired equalized energy consumption of all nodes; the maximum duty cycle is only about 25% higher than the average.

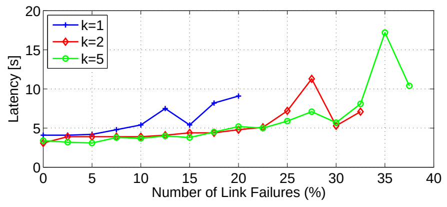
Robustness Analysis So far we considered a benign communication environment without any failing links. In order to analyze Dwarf's performance in a harsher environment, we added random but static link failures while ensuring that the network remains connected. Furthermore, we triggered the alarm messages synchronous to the link failures, in order to avoid that Dwarf adapts to the limited communication environment, and, as discussed before, chose T_w being equal to $1s$.

Figure 6(b) presents the 99-quantile of the latency with up to 40% link failures and different k's. Setting $k = 1$ clearly does not provide a lot of robustness. This can also be seen in Figure 6(c) showing the fraction of messages that are not delivered at all. For instance, there is a single alarm that is not reported with only 7.5% link failures. This can happen if the communication of a node towards the sink is blocked and no redundant messages are sent. In contrast to the failure-less case in Figure 4, having $k = 2$ or $k = 5$ makes a difference in the alarm performance. Especially the 99-quantile shows fewer outliers with bigger k's. The unreported alarms, on the other hand, show a similar trend for different $k \geq 2$. The explanation for this is that each $k > 2$ provides enough redundancy to find a way to the sink, except when the way towards the sink is completely blocked at the alarm-triggering node (i.e., neither parents nor peers are available).

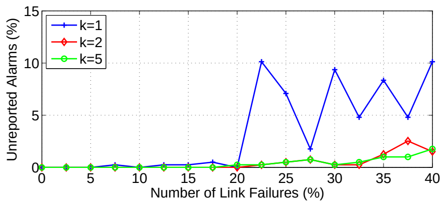
Figure 6(d) shows the distribution of the latency for 30% link failures with $T_w = 1s$ and $k = 2$. Compared to the case without link failures (cf. Figure 4), the alarms take about 50% longer to reach the sink, but still get there on time, except for one outlier. We determined that such outliers are caused by the increase in the number of status messages in response to the link failures, which effectively blocks the channel for alarm messages. There is little we can do about this because ongoing transmissions cannot be aborted.



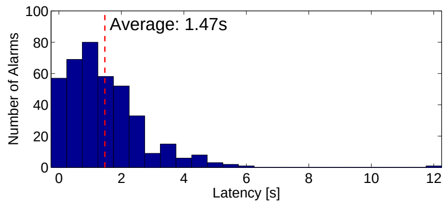
(a) Average latency with link failures



(b) 99-quantile latency with link failures. The lines end when more than 1% of the messages are not reported



(c) Unreported alarm messages



(d) Latency distribution for 30% link failures and $k = 2$

Fig. 6. Fire-alarm performance with link failures, based on 400 triggered alarm messages per sample point.

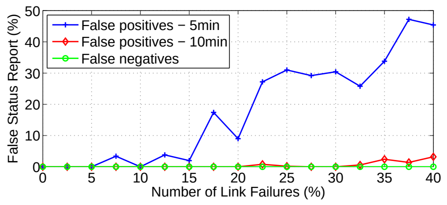


Fig. 7. Status message performance with link failures.

The impact of link failures on the average latency is shown in Figure 6(a). The main observation that can be made is that $k = 2$ shows a better average performance than $k = 1$, since alarm messages are bypassed on other routes.

Not only the alarm messages are affected by link failures, but also the status messages. Their performance in combination with link failures is shown in Figure 7. By design, Dwarf does not report any false negatives that is, Dwarf never reports nodes being alive which actually have failed. In contrast, the number of false positives (alive nodes reported missing) suffers severely from the bad communication environment as shown in Figure 7. The performance can be increased significantly, however, when the reporting time for node failures is increased and the results of several status monitoring intervals can be combined, as shown in Figure 7 for a doubled reporting time of 10min. Alternatively, T_s could be decreased, resulting in increased power consumption.

6 Conclusions

In this paper we presented Dwarf, an energy-efficient, robust and dependable forwarding algorithm for the accurate notification of alarm messages in safety-critical WSN applications. The fundamental idea of Dwarf is to perform a unicast-based partial flooding in combination with a (greedy) delay-aware node selection strategy. Our evaluation, based on a real-world scenario, shows that alarm messages are dependably reported at the sink, even if a substantial number of links in the network fail. On average alarms are delivered over multiple hops in less than a node's wake-up time T_w. For a T_w in the order of 1s, over 99% of the alarm messages are reported well within the required 10s, even if 30% of the links fail. The effective duty-cycle is always below 1%, yielding an operational lifetime of several years. Finally, Dwarf manages to reliably report failed nodes within the target interval of 5 minutes. Under poor conditions, i.e. when many links fail, alarm and status messages actually interfere showing that application scenarios should always be regarded as a whole. We firmly believe that alarm reporting, failure detection, and duty-cycling should be addressed in an integrated way, as Dwarf does.

Possible enhancements that are left for future work are to also consider the children of a weakly connected node (with a certain probability) and to change the number of forwarding destinations k (dynamically) per node, depending on either a node's local view of the network or the history of a message. Furthermore, the link-quality between nodes has been assumed to be static but is likely to vary on a long-term basis. This would require Dwarf to adapt its ring structure accordingly in order to ensure that only good-quality links are being used.

7 Acknowledgments

The work presented in this paper was supported by CTI grant number 8222.1 and the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. We would like to thank Bernhard Imfeld from Siemens Building Technologies for providing us with the real-world scenarios and Amre El-Hoiydi for adapting WiseMAC.

References

1. D. Culler, D. Estrin, and M. Srivastava, editors. *Special issue IEEE Computer on Wireless Sensor Networks*, August 2004.
2. A. El-Hoiydi and J.-D. Decotignie. WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In *First Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004), Lecture Notes in Computer Science, LNCS 3121*, pages 18–31. Springer-Verlag, July 2004.
3. E. Felemban, C.-G. Lee, and E. Ekici. MMSPEED: Multipath Multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks. *IEEE Trans. on Mobile Computing*, 5(6):738–754, 2006.

4. J. Kahn, R. Katz, and K. Pister. Next Century Challenges: Mobile Networking for ''Smart Dust''. In *5th ACM/IEEE Conf. on Mobile Computing and Networks (MobiCom '99)*, pages 271–278, Seatle, WA, August 1999.
5. A. Keshavarzian, H. Lee, and L. Venkatraman. Wakeup scheduling in wireless sensor networks. In *7th ACM symposium on Mobile ad hoc networking and computing (MobiHoc)*, pages 322–333, Florence, Italy, 2006.
6. K. Langendoen and G. Halkes. Energy-efficient medium access control. In R. Zurawski, editor, *Embedded Systems Handbook*, pages 34.1 – 34.29. CRC press, 2005.
7. G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks. In *Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, NM, April 2004.
8. S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.
9. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM Workshop on Wireless Sensor Networks and Application (WSNA)*, pages 88–97, Atlanta, GA, September 2002.
10. M. Maroti. Directed flood-routing framework for wireless sensor networks. In *5th ACM/IFIP/USENIX Conf. on Middleware*, pages 99–114, 2004.
11. P.J. Marrón, T. Voigt, C. Rohner, and B. Ahlgren, editors. *2nd ACM Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Uppsala, Sweden, June 2006.
12. S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *2nd ACM Conf. on Embedded Networked Sensor Systems*, pages 250–262, Baltimore, MD, November 2004.
13. J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *2nd ACM Conf. on Embedded Networked Sensor Systems*, pages 95–107, Baltimore, MD, November 2004.
14. S. Rost and H. Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In *IEEE SECON*, Reston, VA, September 2006.
15. Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-sink reliable transport in wireless sensor networks. In *4th ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc)*, pages 177–188, June 2003.
16. F. Stann and J. Heidemann. RMST: Reliable data transport in sensor networks. In *First IEEE Workshop on Sensor Net Protocols and Applications*, pages 102–112, Anchorage, AK, April 2003.
17. T. Voigt and C. Rohner, editors. *Workshop on Real-World Wireless Sensor Networks (REALWSN)*, Stockholm, Sweden, June 2005.
18. C.-Y. Wan, A. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *ACM Workshop on Wireless Sensor Networks and Application (WSNA)*, pages 1–11, Atlanta, GA, September 2002.
19. S.-C. Wang and S.-Y. Kuo. Communication strategies for heartbeat-style failure detectors in wireless ad hoc networks. In *Conf. od Dependable Systems and Networks*, pages 361–370, San Francisco, CA, June 2003.
20. F. Ye, G. Zhong, S. Lu, and L. Zhang. GRAdient Broadcast: A robust data delivery protocol for large scale sensor networks. *Wireless Networks*, 11(3):285–298, 2005.
21. W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *4th ACM Conf. on Embedded Networked Sensor Systems (SenSys 2006)*, pages 321–334, Boulder, CO, November 2006.