Secondis: An Adaptive Dissemination Protocol for Synchronizing Wireless Sensor Networks

Federico Ferrari, Andreas Meier, and Lothar Thiele
Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland
Email: {ferrari, a.meier, thiele}@tik.ee.ethz.ch

***Abstract*—Reliability and predictability of the timing behavior have shown to be major issues for wireless sensor network deployments. Real-time requirements presented by several applications can be fulfilled by implementing communication schemes that lower possible sources of non-determinism of the timing behavior, assuming that the nodes are synchronized. The predictability of current synchronization protocols, however, cannot be verified, due to potential interferences with other activities.**

In this paper we propose Secondis, a dissemination protocol that periodically synchronizes and orchestrates activities in the network, providing three main benefits. (1) The synchronization task is performed in short time windows, where no interferences can occur, independently of any available communication structure. (2) The synchronization is energy-efficient, and (3) robust against link and node failures. Secondis provides probabilistic bounds on its predictability using probabilistic model checking analysis. It proposes a novel adaptive flooding scheme based on the observation that only a subset of the nodes is important for the dissemination. We analyze the behavior of Secondis in simulation, using realistic models of the wireless channel and hardware clocks.

I. Introduction

Wireless sensor networks (WSNs) are emerging systems for applications such as industrial automation, health-care, surveillance and safety monitoring. WSNs may present real-time requirements, such as bounded end-to-end delay for the data collection. Many sources of non-determinism may however affect the timing behavior, most importantly the inherent unreliability of the wireless channel and possible collisions that occur when two or more sensor nodes simultaneously transmit to a common neighbor. In real-time WSN scenarios it is of fundamental importance to achieve highly predictable communication schemes to minimize interference, and fault containment policies to guarantee a quick recovery of the network after a failure.

Global time synchronization is an important requirement in real-time WSNs, where nodes have to take time-coordinated actions in order to provide the desired reliability. Interference-free communication schemes assume that nodes are synchronized; their predictability strongly depends on that of the underlying time synchronization protocol. Several protocols for synchronizing clocks in WSNs [1]–[4] provide sufficient accuracy in the order of a few microseconds, and require the sensor nodes to periodically exchange messages with time information in order to synchronize their clocks to a common reference time. These protocols, however, provide no guarantees on their own predictability, i.e. on the fraction of the nodes that are synchronized at a certain point in time. This is due to the fact that they either rely on an available structure (i.e. network topology) or require to build an ad-hoc tree. In both cases, the degree of non-determinism is too high to verify the reliability for forwarding the time information. In particular, the synchronization task can interfere with other activities that are executed both *locally* at a node level (e.g. computation, sensing, storage) and *globally* at the network level (e.g. communication). Unreliability of time synchronization protocols has shown to be an important issue for WSN deployments and may compromise the functionality of the overall system [5].

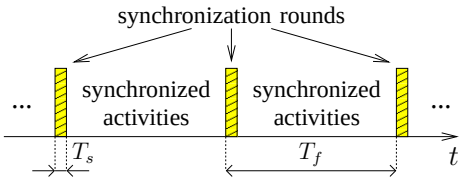


Fig. 1. Synchronization takes place exclusively within time slots of length T_s, and repeats periodically every T_f.

In this paper we propose Secondis, a protocol that propagates the timestamped messages required by standard synchronization protocols. It achieves its predictability by decoupling the synchronization task from other activities: communication required to synchronize the nodes is confined to short synchronization rounds that repeat periodically, as shown in Figure 1. No other activities are allowed to execute within these rounds, so that local and global interferences are minimized. Secondis is designed to perform a quick dissemination of small amounts of data from the root node to the rest of the network, without relying on any existing network topology. It propagates information that can be used not only to synchronize the clocks of the nodes, but also to coordinate activities over time in networks that make use of potentially collision-free communication schemes such as TDMA.

As a major contribution compared to existing dissemination protocols, Secondis provides probabilistic guarantees on the dissemination length for simple network scenarios. A designer can thus choose the desired trade-off between efficiency and predictability for the important synchronization task.

Secondis is a probabilistic protocol that exploits the property that not all nodes are equally important for the dissemination and that the overall dissemination speed is maximized if non-

important nodes send messages rarely compared to important ones. Secondis features an online, distributed algorithm that adapts to the current network topology by estimating the importance of a node for the dissemination.

The rest of the paper is organized as follows. Section II provides background information and related work. Section III introduces the protocol requirements, whereas Section IV discusses the approach to achieve a fast and reliable dissemination. Section V presents Secondis and Section VI outlines its implementation. Section VII analyzes Secondis' online heuristic and Section VIII evaluates Secondis in simulation. Section IX concludes the paper.

II. Background and Related Work

Sensor nodes are usually equipped with cheap oscillators that may introduce drifts on the local clocks from the nominal frequency of up to ± 100 parts per million, i.e. their local clocks may exhibit a skew of up to 12 ms per minute. Even though the clock drift can be estimated, it changes over time [6] (due to, e.g. changing temperature and oscillator aging) and hence frequent resynchronizations are required. Standard client-server clock synchronization algorithms are not suited for WSNs, due to the latter's multi-hop nature and energy constraints.

Over the last years, several algorithms have been proposed to synchronize clocks in WSNs. RBS [1] synchronizes a set of receivers in a cluster to a common reference node. Multi-hop synchronization is achieved by timestamp conversions performed by nodes that belong to different clusters. FTSP [3] achieves a high accuracy by using a linear regression table for drift estimation. A root node is elected and an ad-hoc tree structure is built to propagate the time information. GTSP [4] is a distributed algorithm that optimizes the synchronization accuracy between neighboring nodes. The reported accuracy of FTSP is 0.5 $\mu\mathrm{s}$ per hop, whereas GTSP achieves an optimal accuracy between neighboring nodes. These accuracies are sufficient to organize synchronized activities in WSNs. The protocols do however not provide guarantees on their reliability. They usually rely on an existing network topology or, as in the case of FTSP, they build their own ad-hoc tree to exchange time information. Since other activities may concurrently run, the synchronization messages have an increased probability to be lost, for instance due to a local change in the network topology or to interference with data traffic. No policies are usually implemented to prevent this problem or to guarantee a quick resynchronization of a node. For these reasons, it is very difficult to provide guarantees on the fraction of the nodes that are synchronized in the network.

The class of TDMA protocols for WSNs, such as RT-Link [7], TRAMA [8] and FLAMA [9], potentially achieves collision-free and thus time-predictable communication by assigning dedicated communication slots to each node. The protocols however assume that the nodes are already synchronized and rely on the use of one of the previously discussed synchronization protocols. This means that the predictability of the communication itself strongly depends on the one of the synchronization protocol. If for any reason some nodes are not synchronized, they may decide either to suspend the communication or to keep communicating which leads to potential collisions. In both cases the predictability of the network is greatly decreased.

A solution to increase reliability and predictability of the synchronization task is to confine it to dedicated time windows, decoupled from any other activity (see Figure 1). This requires to fit all necessary communication for the synchronization task into these short time windows. FTSP and other synchronization protocols are however not designed to propagate the time information in a short dedicated time. The obvious solution is to combine a synchronization protocol with a dissemination protocol that allows to rapidly flood the time information, together with possible additional information to coordinate network activities.

Several dissemination protocols for WSNs exist. Their goal is to deliver data from a source node to the rest of the network within a short time period. They try to avoid simple broadcast retransmissions that lead to the broadcast storm problem [10]. SPIN [11] is a family of adaptive protocols that optimize the propagation by performing metadata negotiation. Trickle [12] is the most common dissemination algorithm for WSNs; it is designed for propagating and maintaining code updates and uses "polite gossiping" to broadcast code updates, sending first a summary of the code. Such gossiping and meta-data negotiation approaches are however not well suited for the dissemination of small data, since they introduce overhead of similar size than the data itself. The Flash flooding protocol [13] exploits the capture effect of FM receivers, whereas PBBF [14] provides a probabilistic broadcast protocol whose adaptivity is however not tested under realistic network scenarios. These dissemination protocols cannot be easily optimized for a fast periodic dissemination of only a few bytes within a short time window (e.g. due to metadata exchange).

Secondis differs from these dissemination protocols in several points. It is specifically designed to propagate short messages with a constant length, like synchronization messages. Secondis provides a probabilistic dissemination scheme that requires neither gossiping nor negotiation policies. It exploits the fact that the nodes get synchronized during the dissemination by using a modified slotted Aloha scheme to access the wireless medium, with the important addition that transmission probabilities adapt to changing network conditions.

Due to its simple design and the lack of interference from other activities (i.e. other tasks running on the node and data communication from other nodes), its behavior can be analyzed with a probabilistic model checker, which provides probabilistic bounds on the fraction of the nodes that receive a synchronization message, under some specified settings.

III. Protocol Requirements

Global synchronization. Nodes synchronize to a common reference time, provided by the clock of the root node.

Accuracy. The synchronization accuracy should be comparable to that of existing protocols such as FTSP [3], i.e. in the

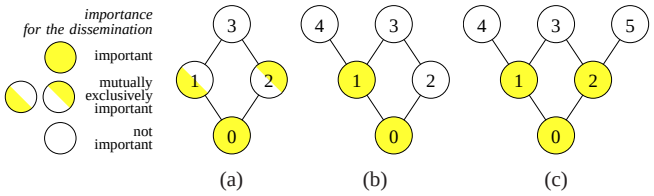


Fig. 2. Three sample topologies.

order of a few microseconds per hop.

Round-based synchronization. The nodes are synchronized regularly in rounds, i.e. there is a frame structure of length T_f that is split up in a synchronization round of length $T_s (\ll T_f)$ and an active round of length $T_f - T_s$ where data exchange takes place (see Figure 1). At the end of each synchronization round, a node must be capable of determining whether it has successfully updated its own clock.

Fast dissemination. The nodes should synchronize as fast as possible, to shorten the length of the synchronization rounds.

Reliability. A large fraction of the nodes should receive the updated time information during a synchronization round.

Predictability. The protocol should be able to provide guarantees on the probability that all nodes receive the message, given some specific network conditions.

Energy efficiency. The radio device is the main consumer of energy in the system, for both transceiving data and idle listening. It is therefore important to minimize the radio duty cycle induced by the synchronization.

Self-containment. The protocol must neither rely on an already existing topology in order to synchronize to the nodes nor on information from the running application. The behavior of the wireless channel changes over time and new nodes may join the network: the scheme should be able to adapt to changes in the network.

IV. Fast and Reliable Dissemination

In order to achieve a fast and reliable dissemination scheme it is important to notice that nodes are not equally important for the propagation. Depending on its position, a node can be considered redundant: its contribution to the propagation process is marginal compared to that of other nodes. Moreover, transmissions from a redundant node might collide with those from nodes that are instead important for the dissemination. Three examples with elementary network topologies help clarifying this issue.

In the 4-nodes network shown in Figure 2(a), time information has to be propagated from node 0, the root of the network, to the remaining three nodes. Once nodes 1 and 2 receive the message, they forward it to node 3. However, if both nodes 1 and 2 transmit the message right after the reception, these transmissions are likely to collide. Only one of nodes 1 and 2 should transmit, whereas the other node is redundant. Since the network is symmetric, it does not matter which node transmits. In the network in Figure 2(b), there is no symmetry: only node 1 should forward the message, so that nodes 3 and 4 receive without collisions. The network in Figure 2(c) is again symmetric, but in this case both nodes 1 and 2 are important for the propagation: only node 1 can communicate with node 4 and only node 2 can communicate with node 5. As discussed before, nodes 1 and 2 should not transmit at the same time, since this would generate a collision at node 3.

A node that receives a message should thus be able to detect: 1) *whether* it is important for the dissemination (i.e. it is required to forward the message); 2) depending on its importance, *when* it should transmit the message. Considering the network in Figure 2(c), a possible solution would require node 1 to send the message, immediately followed by node 2 (or vice versa). This schedule leads to an optimal dissemination length of 3 messages (counting also the root node's message), if an ideal channel is assumed. However, it requires to arbitrate nodes 1 and 2, which cannot directly communicate with each other. The problem becomes very complex when considering topologies with more nodes, due to its analogies with a 2-hop graph coloring problem. Each node should decide when to transmit depending on other nodes choices, leading to the requirement of a global organization. Several distributed coloring algorithms exist [15], but their communication and computation overhead is very expensive. The resulting schedule is also very fragile to (even small) changes in the topology, making such an approach not self-contained. Most importantly, these large-scale structures are too complex for providing guarantees on their reliability.

A. Probabilistic Transmissions

An approach that fulfills our requirements for reliability, predictability, and self-containment makes use of probabilistic transmissions. In the following we assume a slotted Aloha communication scheme during the synchronization rounds, which can easily be achieved since the message length is constant and originates from a common root node.

Let us suppose for simplicity of notation that a node i has received the first message in slot 0. The node then starts repeating Bernoulli trials in each of the following slots $\{1, 2, 3, \ldots\}$ with a probability of transmission x_i, until the outcome of a trial is a success, and the node eventually sends the message and stops the trials. The probability $p_{i,j} = \Pr[i, 1+j]^{TX}$ that node i transmits its message in slot $(1+j)$, with $j \geq 0$ is:

$$p_{i,j}(x) = x_i(1-x_i)^j = x_i \bar{x_i}^j \tag{1}$$

Assuming that links between neighboring nodes are perfect (no failures) and that collisions occur with probability 1 if two or more nodes send a message to the same node in the same slot, the predictability of the network in Figure 2(c) is given by $P^{(N)}_{3,4,5}(x, y)$, i.e. by the probability that all nodes 3, 4, and 5 receive at least one message within $N > 0$ slots.

$$P^{(N)}_{3,4,5}(x_1, x_2) = x_1 x_2 \sum_{j=0}^{N-1} \bar{x_1}^j \left[-\bar{x_2}^j + \sum_{n=0}^{N-1} \bar{x_2}^n \right] \tag{2}$$

It can be shown that $P^{(N)}_{3,4,5}$ is maximized if $x_1 = x_2 = \hat{x}$ (i.e. nodes 1 and 2 have the same transmission probabilities)

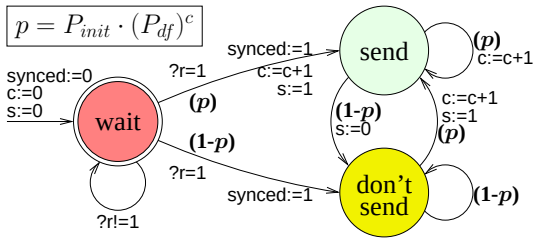


Fig. 3. Discrete-time model of a node, for a single synchronization round.

and that the optimal value $\hat{x}$ monotonically decreases when N increases. For $\hat{x} \neq 0$ and high values of N we have:

$$\lim_{N\to\infty} P^{(N)}_{3,4,5}(\hat{x},\hat{x}) = \frac{2-2\hat{x}}{2-\hat{x}} \tag{3}$$

The function on the right side of (3) only converges to 1 if $\hat{x} \to 0$. Therefore, only very low transmission probabilities allow to achieve high reliability, but at the expense of a very long dissemination phase. A higher transmission probability on the other hand increases the propagation speed, which is traded off by a reduced reliability.

B. Multiple Transmissions

We see from (3) that it is not possible to achieve both reliability and speed if a node sends only one message. However, it has to be determined which are the optimal transmission probabilities, and whether a constant probability or a probability decreasing over time is to be favored. Answering these questions analytically is very complex, and prompted us to make use of the probabilistic model checker PRISM [16].

Figure 3 shows the state graph of a discrete-time model that is executed by every node. Transitions between the states and variable updates are fully synchronized with the slots: they occur with zero delay during the time instant in which a slot ends and the next one begins. The time interval between the beginning of the transmission and the end of the reception of a message lies entirely within a single slot. A node starts its execution in the "wait" state, and remains in that state as long as no messages are received (signaled by $\mathsf{r} \neq 1$) without transmitting any packet. Once a message is received, it keeps making transitions either to state "send" (with probability p) where it sends a packet, or to state "don't send" (with probability $1-p$) where no transmissions are performed. The transmission probability of a node is given by $p = P_{init} \cdot (P_{df})^c$, where P_{init} is the transmission probability for the first message and P_{df} the decreasing factor for subsequent transmissions. For modeling a constant transmission probability, P_{df} is set to 1, otherwise it is set to a value $0 < P_{df} < 1$.

Communications between different nodes (modules) are abstracted as follows. A boolean variable s is set to 1 only during the slots in which the node is sending a message. The read-only variable r is the sum of variables s of all the nodes that can communicate with the node under analysis. A message is successfully received if $\mathsf{r} = 1$, i.e. if exactly one node has sent a message. Collisions therefore occur in every slot in which $\mathsf{r} > 1$. If a message is successfully received, the boolean variable synced is set to 1. A network topology is implemented in PRISM by creating a model that contains several of such modules. Connections between them are defined by setting the r variable of a node as a function of the correspondent s variables of the neighboring nodes.

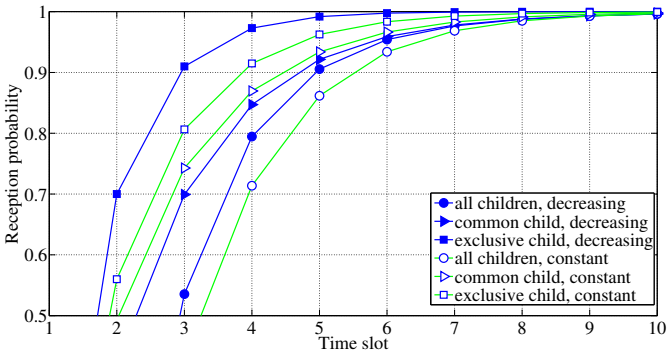


Fig. 4. Decreasing vs. constant transmission probability.

PRISM evaluates the probability under which some specified properties are satisfied by a stochastic process. The property that we want to check is whether all nodes in a network have received a message within a certain slot interval, i.e. if all the modules in the PRISM model have $\mathsf{synced} = 1$. An exhaustive search of the parameter space (P_{init}, P_{df}) allows to find the best parameter setting for the topology in Figure 2(c). In particular, with a constant transmission probability, $P_{init} = 0.56$ is the optimal value; with a decreasing probability, the optimal combination is ($P_{init} = 0.7$, $P_{df} = 0.8$). The resulting reception probabilities for both cases are shown in Figure 4. A decreasing probability achieves a higher reception probability for all the child nodes than a constant probability. The reason is that the two different kinds of child nodes in the network in Figure 2(c) require opposite transmission probabilities. The reception probabilities of exclusive children 4 and 5 are maximized when parents 1 and 2 transmit with probability $p = 1$, whereas the reception probability of the common child 3 is maximized with a transmission probability $p = 0.5$. A decreasing transmission probability yields a better trade-off by being fair to both common and exclusive children.

V. Secondis Protocol

From the discussion in the previous section, we derive that a time dissemination scheme should fulfill three main properties. (1) Simple schemes allow to provide probabilistic guarantees on the timing behavior. (2) Nodes need to have different importance levels for efficiently forwarding the information. (3) Several messages have to be sent with a decreasing transmission probability, in order to achieve both reliability and speed. Secondis is designed to fulfill all these properties.

Secondis follows the state graph depicted in Figure 3, except that Bernoulli trials are only performed every k-th time slot. This means that if a node receives the first message in slot 0, it performs Bernoulli trials only in slots $\{1, (1+k), (1+2k), \ldots\}$. This additional policy is introduced in order to reduce chances

TABLE I
MANDATORY PACKET FIELDS IN SECONDIS.

Field	*Explanation*	*Bytes*
T_{tx}	Timestamp, taken at the MAC layer	4
ID_S	ID of the sender	1
S_{nr}	Sequence number of the current round	1
HC	Sender's hop distance from the root	1
T_{sr}	Start time of the current round	4
ID_P	ID of the sender's parent	1

for collisions due to interferences from nodes out of the communication range (in general, the interference range is larger than the communication range). In particular, simulation results highlight that sending every $k = 3$ slots yields the best results.

The synchronization round ends for all nodes after T_s time units. Additionally, a node is also allowed to finish its round earlier, i.e. as soon as the message has been forwarded C_{max} times, so that it can switch off the radio and save energy. Due to the decreasing transmission probability, chances for transmissions would get in any case very small after several transmissions.

The root node has a special role. It sends its first message when its timer indicating the start of the synchronization round expires; it then keeps sending every k-th slot with probability one, until T_s time units have elapsed after the beginning of the synchronization round (i.e. it sets $P_{init} = 1$, $P_{df} = 1$, $C_{max} = \infty$). All nodes in a $(k-1)$-hop distance from the root are sending in other slots and thus they do not interfere.

As shown in the previous section, the nodes should get assigned different importance levels for the dissemination, which influence their values for the dissemination parameters P_{init}, P_{df}, and C_{max}. Due to the self-containment requirement, the importance levels should be updated online, allowing Secondis to adapt to changes in the topology. This operation has to be performed in a distributed fashion, without requiring that some of the nodes have complete knowledge of the topology. In order to determine the basic guidelines for the importance levels, we consider the elementary topologies in Figure 2 again. The first guideline is that a node should have a High importance level if it is the exclusive parent of a least one node, as depicted in Figure 2(b) and 2(c). The second guideline considers nodes that are not exclusive parents for any child. If two or more such (not exclusive) parent nodes have a common child, only one of them sets a High importance level; the other parents set it to Low. It is of no relevance which parent is the important one. However, since no nodes know the network topology, the guidelines can not be applied directly.

Secondis uses a heuristic that learns about the topology by overhearing the messages of the child nodes and adapts to changes in the network. Messages contain the parent ID (see Table I), which is the ID of the node the first message has been received from in the current round. If a node has several parents, it likely announces several different parent IDs over several rounds. If a node has instead only one (exclusive) parent, it always announces the same parent ID. Secondis determines the node's importance by keeping a short record of the overheard messages and evaluating them every level-update period consisting of R rounds. By default all nodes have Medium importance. A node sets its importance level to High/Low according to the following rules.

High importance. At least one child node announced that the node under analysis was its parent in more than f_H percent of the last R overheard rounds (e.g. $f_H = 0.7$).

Low importance. Every child announced that the node under analysis was its parent in less than f_L ($< f_H$) percent of the last R overheard rounds (e.g. $f_L = 0.3$).

In order to avoid false positives from nodes with bad link qualities, only children overheard in at least l of the last R rounds (e.g. 5 out of 16) are considered. Furthermore, transitions between the High and the Low importance levels (and vice versa) are not performed immediately, but through an intermediate step at the Medium level, minimizing the effects of possible estimation errors. Simulations show that values $R = 16$, $l = 5$, $f_H = 0.7$ and $f_L = 0.3$ provide high-quality estimation of the importance level and are used in Sections VII-VIII for the evaluation. The memory footprint required by the heuristic to store information about the heard children is $8 + 3 \log_2 R$ bits per child: 20 bits per child if $R = 16$.

This heuristic allows the nodes to automatically adapt their importance levels to changing conditions of the network topology. At the beginning, all the nodes start with a Medium importance level; after R rounds, they evaluate whether to update their importance levels, depending on the feedback received from the children. After a few periods of R rounds, they converge towards stable importance levels; in cases where two or more nodes have only common children, as in Figure 2(a), only one of them eventually gets a High importance level, whereas all the others converge to a Low level. The optimal values may then change again later on if for any reason the topology changes. In Section VII we show that our simple heuristic provides a near-optimal importance level assignment.

VI. SIMULATION

We have implemented the Secondis dissemination protocol in Castalia [17], an OMNeT++ based simulation platform. Castalia is designed for WSNs and allows for a realistic and detailed protocol evaluation, due to its realistic models of the wireless channel, the radio device, and the energy consumption. The wireless channel model is based on the log-normal shadowing model: a large fraction of the links in a network commonly exhibits only a poor packet reception rate and shows asymmetric behavior. Packets are further lost due to collisions, which are calculated using the signal to interference ratio model. Castalia features an accurate HW model for the radio, with transition times between the different radio states (e.g. switching from sleep to RX, or from RX to TX), which have to be accounted for in order to ensure correct timing of the implementation. We added a model for local clocks that bounds the drift and the variation of drift. This is a reasonable assumption, given that the clock drift is only gradually influenced by changing conditions such as temperature and battery voltage [6].

TABLE II
PARAMETER VALUES USED IN SECONDIS.

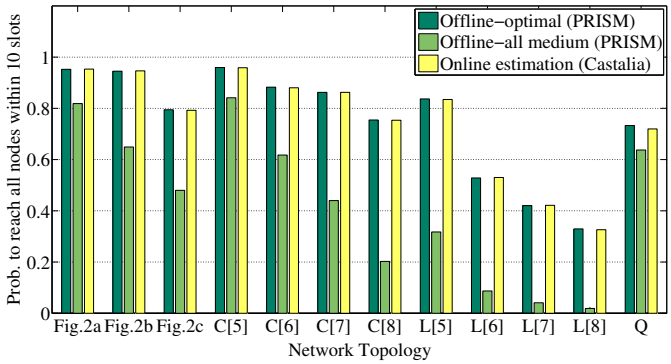
Param.	*Explanation*	*Node Importance Level*		
		High	*Medium*	*Low*
P_{init}	Initial transm. probability	0.7	0.4	0.1
P_{df}	Prob. decreasing factor	0.8	0.5	0.5
C_{max}	Max. transm. per round	7	5	2

The implementation of Secondis follows the state graph depicted in Figure 3, but Bernoulli trials are performed every k-th slot and a message is transmitted at most C_{max} times. In particular, a timer expires at the beginning of every synchronization round, which indicates to switch on the radio and to set the node to the "wait" state, waiting for a message with the updated time information. A node is synchronized after receiving the first synchronization message, and then it takes part in the dissemination of the updated timestamp. The root node omits the first "wait" state, since it provides the time reference; it sends its first message right after the synchronization round timer expires, and sends a message every k-th slot. The packet fields required by Secondis to disseminate the time information and to adapt to the network topology are shown in Table I. Other fields can be added when additional information has to be disseminated to organize activities in the network.

A Secondis slot has to be long enough to cover the time required by a node radio to transmit a packet and to provide enough guards against possible residual synchronization errors between the nodes. Due to the short length T_s of a synchronization round (a few tens of milliseconds, see Section VIII), the effects of the clock drift within a synchronization round is negligible. Since the transmission of a standard Secondis packet with the fields shown in Table I requires around 1 ms with common sensor node radios, a conservative choice for the slot length is $T_{slot} = 1.2$ ms. An extensive parameter search with Castalia showed that values provided in Table II are suitable for a wide range of network conditions. These values are used in Sections VII-VIII for the evaluation.

In the discussion above, it is assumed that a node already knows when the next synchronization round is going to take place, i.e. it has already scheduled a timer for the next synchronization round. When a node is joining the network (e.g. right after the deployment or after a node reset) it obviously does not know yet when the next synchronization round takes place. For the current implementation we chose the simplest solution, where a node listens continuously (at most for one frame T_f) in order to ensure that a synchronization round is overheard.

In the current implementation, the drift of the local clock is estimated using the same mechanism as in the FTSP [3] protocol: a linear regression is performed on the received timestamps. Secondis thus achieves exactly the same synchronization accuracy as FTSP, if the frame length T_f is equal to FTSP's synchronization period (e.g. 0.5 μs per hop for a synchronization period of 30 s, as claimed in the original paper). Secondis, however, is suitable for any other synchronization protocol that synchronizes the nodes by using a single reference clock and requires a one-way message exchange (see Section II). We chose FTSP since it is the most common protocol for synchronizing WSNs.



$C[i]$/$L[i]$: Cycle/line topology with i nodes. Q: Cube topology.
Fig. 5. Importance level estimation.

VII. NEAR-OPTIMALITY OF SECONDIS' HEURISTIC

Secondis proposes a distributed and adaptive heuristic for an online estimation of importance levels. In order to evaluate how it affects the overall predictability, we compare the simulated probability of reaching all nodes, when the importance levels are estimated online by the heuristic, to the ones resulting from offline assignments of the importance levels, which can be computed exactly by the PRISM model checker. In particular, two offline assignments are used for the evaluation. The first one is the offline–optimal assignment, which generates the lowest worst-case probabilistic bound for the dissemination delay and therefore provides the highest predictability. In the second assignment (offline–all medium) all nodes have Medium importance levels, and it corresponds to the initial setting of Secondis. We can thus evaluate in detail how well Secondis' importance estimation increases the predictability with respect to the initial case and how close the predictability is to the one of the offline–optimal assignment.

In order to get a detailed comparison, we study 12 topologies with up to 8 nodes in the network. We cannot model larger networks, since probabilistic model checkers suffer scalability problems due to state space explosion. In particular, we study the three topologies illustrated in Figure 2, linear topologies $L[i]$ of length i ranging from 5 to 8, cycle topologies $C[i]$ with 5 to 8 nodes and a cube topology Q. In order to compare the results from PRISM with the ones from simulations in Castalia we use the same channel model, i.e. perfect links are assumed. The probabilities of reaching all nodes within 10 time slots are shown in Figure 5. For each topology, the first (dark) bar shows the probability for an offline-optimal assignment of the importance levels, provided by PRISM. The second bar shows the probability when all importance levels are set to Medium, also computed by the model checker. The third (light) bar shows the probability resulting from Castalia simulations of Secondis' online estimation heuristic. In particular, 4000 rounds are simulated for each topology and the first $10R$ rounds are discarded, in order to remove the phase in which Secondis adapts to the topology.

The reception probabilities show a clear dependency from the topology and the maximum hop distance of the nodes from the root. For the linear topologies $L[i]$, the probability decreases with an increase of i. The same holds for the cycle topologies, but only for even numbers of i: the maximum hop distance increases by one only if two more nodes are added.

Secondis' online estimation increases the reception probabilities with respect to the initial Medium level and it approximates the optimal case in all the studied topologies. In particular, the maximum improvement from the fixed Medium level assignment is given by the $L[8]$ topology, where the probability to reach all the 8 nodes within 10 slots from the beginning of a round is increased from 1.9% to 32.6%, whereas the optimal assignment would lead to a probability of 32.9%. The cube topology represents the case in which there is largest difference between the probability of the offline-optimal assignment (73.3%) and the one of the Secondis' estimation (72.0%), whereas the Medium-level assignment achieves 63.7%. These results show that the heuristic used by Secondis provides an assignment of the importance levels that is close to the optimal one. The resulting predictability, therefore, is also always very close to the optimal one, whose values can be computed exactly with PRISM.

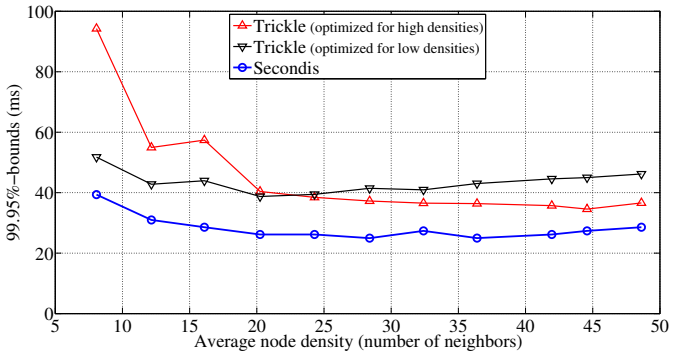
VIII. Evaluation

Secondis is further evaluated using Castalia's realistic model of the wireless channel, with random packet losses. A large number of different network topologies is used to compare it with Trickle. We place the nodes on randomized grid topologies (grid with Gaussian displacement), which are well suited for representing real deployments where the nodes are rather evenly spread over the observed area and not just randomly deployed. If not otherwise stated we run each simulation for 20,000 s, having a synchronization round every T_f =30 s. Since we are interested in the long run behavior of the estimation, we discard the synchronization delays of the first $10R$ rounds in order to remove the startup phase when Secondis adapts to the topology. This results in 506 evaluated synchronization rounds for every simulation run.

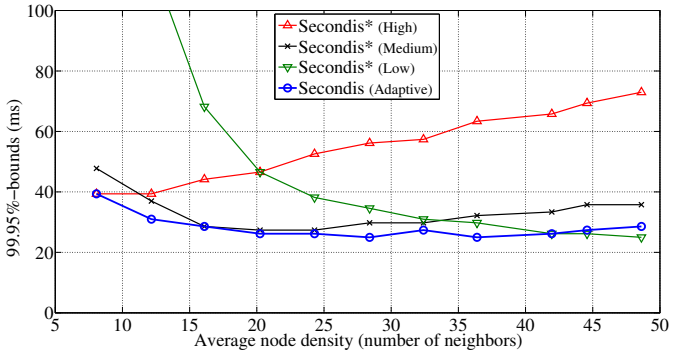
We analyze the 95% and 99.95% bounds, i.e. the time required to synchronize at least 95% and 99.95% of the nodes, or otherwise said the probability of missing a synchronization beacon every 20th and 2000th round, respectively. Furthermore, we discuss Secondis' ability to adapt to changes in the environment (network topology) and its energy efficiency. We do not evaluate the synchronization accuracy since we use exactly the same mechanism of FTSP to estimate the drift between the local clock of a node and the reference clock, achieving thus the same accuracy.

A. Node Density

We evaluate Secondis in detail, in order to analyze its dissemination speed for a wide range of network scenarios. In particular it is interesting to see how well Secondis adapts to different node densities, which depend on the networks deployment site. We also want to compare Secondis with other solutions that provide a quick dissemination of a timestamp. For this reason we implemented the popular Trickle protocol (see Section II) in Castalia.



(a) Trickle vs. Secondis.



(b) Secondis adaptive vs. non-adaptive.

Fig. 6. Delay bounds vs. node density.

We evaluate Secondis and Trickle using Castalia, exploiting its realistic models of the wireless channel, radios, and clocks. In order to evaluate different node densities we use a fixed area of 100 x 100 m, ranging the numbers of nodes from 30 to 140 in steps of 10. We generate 30 different topologies (randomized grids) for each number of nodes using different seeds and having the root always in the corner; this results in 360 different topologies with an average node density ranging from 8 to 48 neighbors per node, and an average network size of three hops. With low node densities the delay bounds are increased, since the average distance between the nodes is increased, and results in a large fraction of unreliable links.

Trickle adapts to the current traffic load only during a single synchronization round; it requires an initial parameter setting, which can be tuned towards a given network topology. Based on an exhaustive parameter search, we determined the optimized parameter set (τ_H, τ_L, k) for Trickle [12]. For low node densities values (50 ms, 10 ms, 5) showed to be optimal, whereas for high densities the optimal setting is (40 ms, 20 ms, 1). The root node uses (τ_H, τ_L) = (20 ms, 10 ms) in both cases, using the same k value as the other nodes. The effect of this parameter adaptation is depicted in Figure 6(a), showing the 99.95% probabilistic bound. It can clearly be seen that Trickle cannot provide a parameter setting that allows a fast dissemination for both dense and sparse networks, since it does not provide a mechanism to learn from

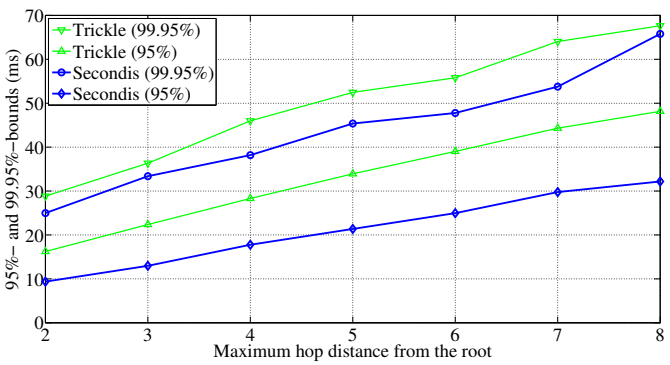


Fig. 7. Delay bounds vs. network size.

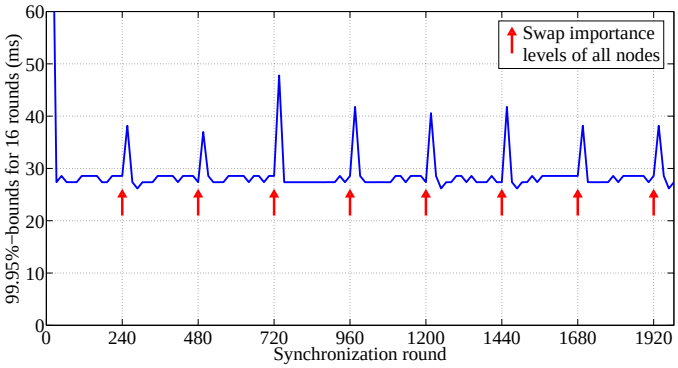


Fig. 8. Secondis' adaptivity to importance level changes.

the previous rounds. Although Trickle allows to be optimized for a given node density, this is usually not straightforward. Firstly, it is required to estimate the node density; secondly, the node density can differ a lot within the network and hence a global parametrization for all nodes is not suitable.

Secondis on the other hand adapts to the network topology at runtime, without needing a designer to manually tune any of the parameters of Table II for a specific scenario: each node adapts its importance level online, and thus chooses the most suitable parameters triple. As shown in Figure 6(a), Secondis features a rather constant dissemination speed for the whole range of the network densities. In particular, Secondis allows a much faster dissemination of the timestamp than Trickle. While the dissemination speed for Secondis ranges from 25.0 to 39.4 ms, it ranges from 34.6 to 51.8 ms for Trickle (using the best suited parameter setting for each data point, which has to be computed beforehand). In average Secondis allows for a 25% reduced dissemination time in comparison to Trickle.

In order to get a more detailed insight into the adaptation possibility we compare Secondis with a Secondis* version, which does not adapt the importance levels. Instead, all nodes have the same importance level (either High, Medium or Low). Similar to Trickle, Secondis* cannot be optimized for the whole range of network densities with the same parametrization as shown in Figure 6(b). However, it should be noted that if Secondis* is parametrized with a Medium importance level, it still outperforms Trickle for the whole node density range. This shows that the slot-based dissemination scheme of Secondis is better suited for a quick flooding of the network than Trickle's window-based scheme.

B. Network Size

The dissemination length is likely to increase with the network diameter and is analyzed in detail in the following. Since the dissemination speed depends also on the node density, we analyze the impact of the network size with a rather constant node density of 12 to 16. This is achieved by deploying 20 to 90 nodes on a rectangular area with a fixed height of 30 m and a variable width ranging from 80 to 480 m. For the comparison with Trickle, we optimize Trickle for the given node density ($\tau_H = 40$ ms, $\tau_L = 10$ ms, $k = 3$).

Additionally to the 99.95% bound discussed in the previous sections, we also analyze the 95% bound, as shown in Figure 7. The figure exhibits a nearly linear dependency of the dissemination speed on the network size for both Secondis and Trickle. Even though Trickle parametrization is optimized for the given networks, it cannot match the dissemination speed of Secondis for both bounds. Comparing the 95% and 99.95% bounds highlights the flexibility available for a designer to trade off reliability for a shorter length of the synchronization round T_s. For instance, a network with maximum distance of 5 hops from the root requires only 21.4 ms to flood the network with a 95% probability, but 45.3 ms to achieve a 99.95% probability. This clearly highlights the fact that Secondis is well suited to be optimized for a given application scenario.

C. Adaptivity to Changes

In the previous evaluations we considered the long-term behavior and discarded the initial phase where Secondis adapts to the network topology. In order to see how fast Secondis adapts to changes in the network, we set up simulations where the importance levels of the nodes are intentionally changed. Every 240 rounds the importance levels of all nodes are swapped, so that nodes that have High importance levels receive Low importance levels and vice versa. Figure 8 shows simulation results from 540 different randomized grid topologies of 50 nodes deployed in a 100 x 100 m area. For each run we simulated 2,000 synchronization rounds. The 99.95%-bounds are computed for every period of $R = 16$ rounds, i.e. for each update of the importance levels. At the startup the delay is high due to the fact that FTSP regression table needs to be filled to have a good synchronization quality. Secondis needs only one level-update period to adapt to the topology, and then the bound remains stable between 26.2 ms and 28.6 ms. After the swap of the importance levels, the bound increases to values that range between 37.0 ms and 47.8 ms. Secondis recovers from this misalignment of the levels in only one level-update period, highlighting its strength for adapting quickly to changing network conditions.

D. Energy Consumption

Energy is a very scarce resource for most WSN deployments. Having the radio device as the main energy consumer, it is important to estimate the energy requirement for

the network synchronization. Since most WSN deployments operate unattended for long periods (e.g. several years), we are interested in Secondis operation (i.e. maintaining the synchronization) and neglect the overhead for getting the network initially synchronized. The overall radio's duty cycle for keeping the network synchronized is $\gamma = \gamma_s T_s / T_f$, where γ_s is the duty cycle during a synchronization round with length T_s. Since the frame length T_f is usually given by the application, the energy consumption can be minimized by shortening the synchronization round T_s and by reducing the duty cycle during the synchronization round γ_s. Secondis switches on the radio at the beginning of the synchronization round and switches it off as soon as either the timestamp is forwarded C_{max} times or the synchronization round ends after T_s time units. Hence, the worst-case duty cycle during the synchronization round is $\gamma_s = 1$. If we have an application that needs synchronization every $T_f = 30$ s and we choose a very conservative round length of 80 ms we result in a worst-case duty cycle of only $\gamma_{wc} = T_s/T_f = 0.08/30 = 0.27\%$. This worst-case duty cycle is independent from the importance level of a node, i.e. from the number of transmissions during the synchronization rounds.

Simulations show an average duty cycle of 0.18%, indicating that the radio is usually switched off before the end of the synchronization period, thanks to the use of the maximum number of transmissions C_{max}. Trickle always results in a higher duty cycle, since it requires a higher T_s to disseminate the time information with a comparable predictability. A comparison with standard synchronization protocols, such as RBS, FTSP and GTSP, is not straightforward since their energy cost depends greatly on the underlying communication structure. In particular, the number of messages that need to be exchanged depends on the network protocol, whereas the energy consumption for transceiving a single message depends on the MAC. Furthermore, there might be concurrent activities that could result in collisions and may require retransmissions.

IX. Conclusion

In this paper we proposed Secondis, a dissemination protocol for real-time WSNs. Secondis increases the predictability of available time synchronization protocols, and can easily be extended to propagate small amounts of data for coordinating various network activities. It achieves its predictability by confining all the communication required by a synchronization protocol to very short windows that are repeated periodically, where no other activities are allowed to be concurrently executed. It only requires a few tens of milliseconds to synchronize the entire network, leaving enough time in between to execute other operations. Unlike common dissemination protocols, Secondis provides probabilistic bounds on the fraction of the nodes that receive the desired information within a given period by means of probabilistic model checking analysis.

Simulation results under realistic settings show that Secondis adapts to a wide range of network scenarios and to changes in the environment. Hence, there is no need to manually tune its parameters, due to its ability to dynamically determine the importance level of a node for the dissemination. Due to this speed it also achieves a high energy efficiency: the duty cycle is bounded for all nodes in the network, regardless of the number of messages sent during the dissemination. Secondis outperforms Trickle, the reference dissemination protocol for WSNs, in all the simulated scenarios. Within 50 ms it synchronizes with a probability higher than 99.95% networks with a distance of up to 6-hops from the reference node and different node densities.

Acknowledgment

The work presented in this paper was supported by 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. The authors thank Matthias Woehrle and Marco Zimmerling for revising a draft of this paper, and Kai Lampka for his valuable help with the PRISM model checker.

References

[1] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[2] S. Ganeriwal *et al.*, "Timing-sync protocol for sensor networks," in *SenSys'03*, New York, USA, 2003.

[3] M. Maróti *et al.*, "The flooding time synchronization protocol," in *SenSys'04*, New York, USA, 2004.

[4] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *IPSN'09*, Washington, DC, USA, 2009.

[5] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *OSDI '06*, Berkeley, CA, USA, 2006.

[6] K. Römer, P. Blum, and L. Meier, *Sensor Networks*. John Wiley & Sons, New York, Jul. 2005, ch. Time Synchronization and Calibration in Wireless Sensor Networks.

[7] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A time-synchronized link protocol for energy- constrained multi-hop wireless networks," in *SECON'06*, Sep. 2006.

[8] V. Rajendran, K. Obraczka, and J. J. G.-L. Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *SenSys'03*, New York, USA, 2003.

[9] V. Rajendran, J. Garcia-Luna-Aveces, and K. Obraczka, "Energy-efficient, application-aware medium access for sensor networks," in *MASS'05*, Nov. 2005.

[10] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom'99*. New York, NY, USA: ACM, 1999.

[11] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *MobiCom'99*, New York, USA, 1999.

[12] P. Levis *et al.*, "Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *NSDI'04*, Berkeley, USA, 2004.

[13] J. Lu and K. Whitehouse, "Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks," in *INFOCOM'09*, 2009.

[14] C. Sengul, M. J. Miller, and I. Gupta, "Adaptive probability-based broadcast forwarding in energy-saving sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, no. 2, 2008.

[15] R. Kawano and T. Miyazaki, "Distributed coloring algorithm for wireless sensor networks and its applications," in *CIT'07*, Oct. 2007.

[16] A. Hinton *et al.*, "PRISM: A tool for automatic verification of probabilistic systems," in *TACAS'06*, 2006.

[17] A. Boulis, "Demo Abstract: Castalia: Revealing Pitfalls in Designing Distributed Algorithms in WSN," in *SenSys'07*, New York, USA, 2007.