CUPID: A Communication Pattern Informed Duty Cycling for Large-scale, Low-delay Sensor Applications

Daniela Krüger, Stefan Fischer, and Dennis Pfisterer
Institute of Telematics, University of Lübeck, Germany
{*krueger, fischer, pfisterer*}*@itm.uni-luebeck.de*

***Abstract*—In an Internet of Things (IoT), a plethora of tiny devices will extend the Internet to the physical world and allow for a completely new class of applications. Two possible IoT scenarios are public safety (e.g., surveillance of areas and borders) and smart cities that offer smart services to improve the lives of a city's inhabitants. Both share many underlying challenges in terms of realization. They comprise large-scale deployments of sensor nodes and require long-term, battery-driven operation, low-delay reporting of events, as well as secure and attack-resilient design. However, experiences from real-world trials have shown that decent trade-offs between these two conflicting goals are hard to find. In this paper, we show how staggered wake-ups achieve this. We call this low-delay and low-power duty cycle management scheme *CUPID* because its parameterization is based on the expected communication patterns in the network, duty-cycle and latency requirements. We show by simulations and real-world experiments with more than 150 nodes that our scheme significantly reduces the packet delay for low duty cycle settings, especially in large networks.**

***Keywords*-Wireless Sensor Network, Duty Cycling, Low-delay event reporting, Attack-resilience, Real-world measurements**

I. Introduction

It is envisioned that in the future, all kinds of devices ranging from resource-constraint wireless sensor nodes to powerful server-class computers will interact to form an Internet of Things (IoT). In such a setting, tiny devices will extend the Internet to the physical world and allow for a completely new class of applications. Since the introduction of the IoT vision over a decade ago, it has evolved tremendously since the underlying technologies are maturing. Envisioned application scenarios include environmental monitoring, personal health monitoring, monitoring and control of industrial processes, public safety, smart spaces and increasingly smart cities.

In the public safety domain, the surveillance of areas (e.g., for trespasser detection) is a typical application domain of wireless sensor networks. Examples are [2], [3] and the FleGSens project [4], which realize trespasser detection and localization systems for borders and critical areas. Smart cities have recently gained momentum and examples of smart cities are projects such as CitySense [5], Oulu Smart City [6], T-City Friedrichshafen [7] and the recently started EU FP7-funded SmartSantander [8] project. I such city-scale deployments dozens of thousands devices are being or going to be deployed and will offer smart services to improve the lives of a city's inhabitants.

Despite their differences, both application scenarios share many underlying challenges in terms of realization. These networks are comprised of a large number of wireless sensor nodes that are deployed in a large-scale environment. For the majority of nodes, mains power supply is not an option due to cost or unavailability and hence the nodes must operate on batteries. Another shared property is that in case of an observed event (e.g., a trespasser or a bus approaching a station), the event must be reported as fast as required to a base station that provides access to a backbone network such as the Internet. Compared to the number of nodes, the number of base stations will be small and nodes must forward data in a multi-hop fashion towards the nearest base station. In addition, both networks operate in a potentially hostile environment where they are subject to attacks.

In such networks, important factors are network lifetime, the time to report an event, and resilience. As the replacement of batteries is disproportionately expensive because of inaccessibility and the large number of nodes, saving energy is imperative – typically by using an very low duty cycle ($< 1\%$) – to achieve a long network life time. Simply decreasing the duty cycle results in high multi-hop latencies as nodes must wait for their neighbors to wake up and be ready-to-receive before they can forward a message but especially for critical messages (e.g., alarms) low latencies are crucial. To save energy, approaches such as Low Power Listening (LPL) are frequently used where nodes periodically sample the radio interface for preambles that indicate that a packet is to be received and switch to active mode for reception. Low Power Listening is vulnerable to so-called *sleep deprivation* where attackers drain the nodes' batteries by wasting energy, e.g., by sending messages or preambles that cause nodes to stay awake.

This goal conflict requires a careful trade-off and is nowadays often custom-tailored for each individual application. In this paper, we present a novel delay and communication pattern optimized duty cycle management scheme for sensor networks called CUPID (CommUnication Pattern Informed Duty Cycling in Sensor Networks). Its focus is to minimize the latency in multi-hop networks, which use extremely

low duty cycles. We picked up the idea of *staggered wakeups* [9], [10], [11], [12], which we have extended and generalized. In this approach, nodes wake up subsequently in the direction where packets are forwarded. In contrast to existing work, CUPID copes with unreliable communication links and hardware constraints, no node redundancy, and with different communication patters (instead of only allowing network-to-sink communication). In addition to the aforementioned properties, CUPID is resilient against sleep deprivation attacks and can therefore be used in security related applications. The work presented in this paper is an extended version of [1].

The remainder of this paper is organized as follows. In the next section, we review related approaches. Following, Section III introduces CUPID in detail. Then, Section IV provides a simulative and Section V an experimental evaluation. Section VI concludes the paper with a summary and future work.

II. Related Work

In general, there are different types of power management techniques. One of the earliest proposals is *Low Power Listening* (e.g., used in Wisemac [13]), in which nodes are unsynchronized and periodically activate their radio for a fixed duration. A sender transmits short preambles before sending the actual message causing all addressed receivers activating their radio at the respective point in time. However, this implies vulnerability to sleep deprivation attacks.

In *MAC layer schemes* like S-MAC and T-MAC media access during synchronized active slots is coordinated using Ready-To-Send/Clear-To-Send mechanisms, which implies that only unicast communication is possible as well as vulnerability to sleep deprivation attacks.

K-coverage approaches assume that sensing is only possible when a node is active. We do not consider these approaches here, since we assume that sensors have the capability of waking up the node. Also, we don't compare to *on-demand wakeup* radios [14] where ultra-low-power radio interfaces wake up the main radio interface to receive a message. The special hardware required for the first radio significantly increases cost.

Scheduled wakeup schemes do not integrate the duty cycling into the MAC layer but they organize their sleep cycles using predetermined or random schemes. These can be further divided into mechanisms with and without time synchronization. Asynchronous wakeup methods (e.g., [15]) do not require synchronized clocks, but they increase the end-to-end delay especially over long distances. It is therefore common practice to assume (loosely) synchronized clocks. Examples for this approach are *staggered wakeups* where nodes wake up subsequently in the packet forward direction. Stankovic et al. [10] call it *streamlined wakeup*, but assume reliable communication links and high node redundancy. Moreover, they explain their end-to-end optimization only in a short overview and focus on minimizing the detection delay of one sensor event instead.

In [9], Li et al. call the approach *fast path algorithm*. Applying additional wake phases they establish a single path between source and sink, but this is independent from a global duty cycle and from other paths. As a node can handle only a few paths their approach is only appropriate if few nodes send data to few other nodes. In contrast to them we concentrate on a solution for the whole network to allow for network-to-sink communication and vice versa.

Bisnik et al. [11] convert the network into a graph, vertices representing nodes and edges representing communication links. The goal is now to assign (awake) slots to all nodes so that the maximum end-to-end communication delay is minimal. Using the graph-theoretical abstraction the authors prove that this optimization problem is NP-hard given all data is available at a central station. They derive an optimal solution for a rectangular topology where nodes are adjusted in a grid and have exactly four neighbors. However, such a topology can hardly be produced in reality and also assumes reliable communication links. Moreover, the smallest considered duty cycle is 5% if no messages are sent. Nodes sending a message must have knowledge about the wake phases of all destination nodes, which are not necessarily at the same time, and must be additionally awake during these times. By contrast, we investigated constant duty cycles of up to $0{,}1\,\%$ while enabling broadcast communication at the same time.

The authors of [16] describe a wave based communication mechanism to deliver sensor information to and from a designated node without the need of centralized controls. However, they do not consider the resulting duty cycle whereas we meet the requirement of a constant duty cycle. In addition, they base there results exclusively on simulations.

Keshavarzian et al. propose two methods they call *crossed-ladders* and *multi-parent scheme* [12], where nodes are divided into groups depending on their hop based distance to a reference node. Each group is awake during one slot. But their theoretical considerations also assume reliable communication links and disregard the hidden-terminal-problem. Additionally, the authors consider only small networks of at most 4 hops. Their second approach does not support broadcast communication as the network is divided into several partitions, which work independently from each other. Simulation results refer exclusively to the generation of the network partitions. Experimental results are not presented.

To summarize, MAC layer schemes are inefficient, vulnerable to sleep-deprivation attacks or do not support broadcast communication, k-coverage protocols make different assumptions, and on-demand wakeup radios require expensive additional hardware. The techniques closest to our approach are scheduled wakeup schemes. Here, we point out that

other approaches do not scale or do not support periodic or broadcast communication. Many authors base their work on unrealistic assumptions and it remains open in which way varying communication ranges influence their approaches. In contrast to these, we investigate the influence of medium access and the hidden terminal problem, which is unavoidable when nodes residing in the same level cannot hear each other. Moreover, CUPID guarantees that all neighbors of a sending node are ready-to-receive so that arbitrary communication patterns are supported and nodes are not obliged to keep track of forwarding directions or sleep cycles of parent nodes.

III. CUPID - Communication Pattern informed Duty Cycling

In this section, we present CUPID, our communication pattern informed duty cycle management scheme. The major goals of CUPID are

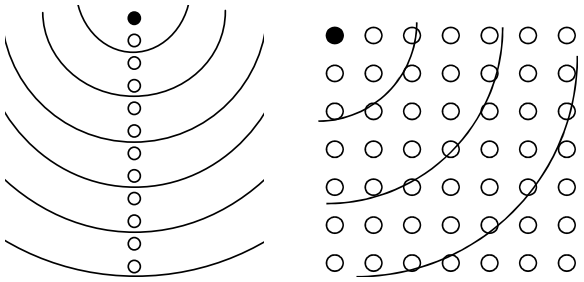
1) conservation of energy,
2) resilience against sleep deprivation attacks,
3) minimization of end-to-end message delay,
4) support for broadcast communication, and
5) reduction of collision probability.

Depending on the requirements of the application, CUPID can be parameterized in order to meet them.

The basic idea of our scheme is that groups of adjacent nodes wake up and enter sleep mode subsequently. The idea is not completely new, but for practical use, several problems must be taken into account and the user must be aware of advantages and disadvantages. All nodes in one group follow the same sleep/wake-cycle. One after the other group wake up (depending on their hop-based distance to a reference node, e.g., a sink), stay awake for the same period of time and go asleep one after the other again. On a global level, these wake phases form a *wave* from the sink to the farthest nodes and vice versa.

We consider a line-shaped network topology as shown in Figure 1(a) where nodes are arranged in a 1-dimensional way and the generalized case shown in Figure 1(b). The lines limit the hop-based range to the reference node (black dot). The particular property (discussed in Section IV-B) of the rectangle topology with regard to CUPID is that not all nodes of the same group are within each other's communication range.

Figure 2 shows three groups from a line-shaped network: Group $n-1$, n and $n+1$ where n means n hops away from the reference node. The boxes indicate different states, e.g., Group $n+1$ wakes up at T_0 and T_2 and goes to sleep at T_1 and T_3. During a wake period, a node is in receiving mode (indicated by Rx) and may only transmit during a certain interval (indicated by Tx). Like this, the nodes from Group n can transmit to the nodes of both neighboring groups at the same time.



(a) Line-shape (b) Rectangle-shape

Figure 1. Considered network topologies and the hop-based distances to the reference node (black dot)

The major assumption of our scheme is that some kind of time synchronization makes sure that the difference between two clocks stays below an upper bound (cf. Section III-A1). In addition, we assume the presence of at least one reference node (typically a sink), from which all other nodes calculate their hop-based distance. For estimating the network diameter another reference node is needed, which is furthest to the sink or the user must estimate the diameter. The scheme is based on fixed node positions. A looser assumption is that data flows from or to a reference node. CUPID is optimized for this kind of data flow. However, multi-hop communication orthogonal to the wave takes disproportionately more time.

A. Design

This section introduces our considerations when CUPID is applied on real network. We assume that application designers have a fixed application scenario where some parameters are known a priori (e.g., the network's diameter) and some are desired (e.g., maximum end-to-end delay or duty cycle). Table I lists the parameters relevant to CUPID.

Abbreviation	Unit / Range	Name
ND	$[Hops]$	Network Diameter
EED	$[sec]$	max. End-to-End Delay
DC	$\in [0,1]$	Duty Cycle
$d_{slotlen}$	$[sec]$	Slot Length
d_{sil}	$[sec]$	Silence Length
d_{SFL}	$[sec]$	Super Frame Length
d_{tol}	$[ms]$	Tolerance Length

Table I
Parameters relevant to CUPID

The *Network Diameter* $ND[hops]$ is the maximum number of hops from the reference node. The *End-to-End Delay* $EED[sec]$ describes the maximum amount of time messages need to travel from source to sink. The *Duty Cycle* DC is the percentage of time, where nodes are awake, i.e., the radio interface is active. The remaining parameters are shown in Figure 3. Each group shares a so-called *Sending Slot* of

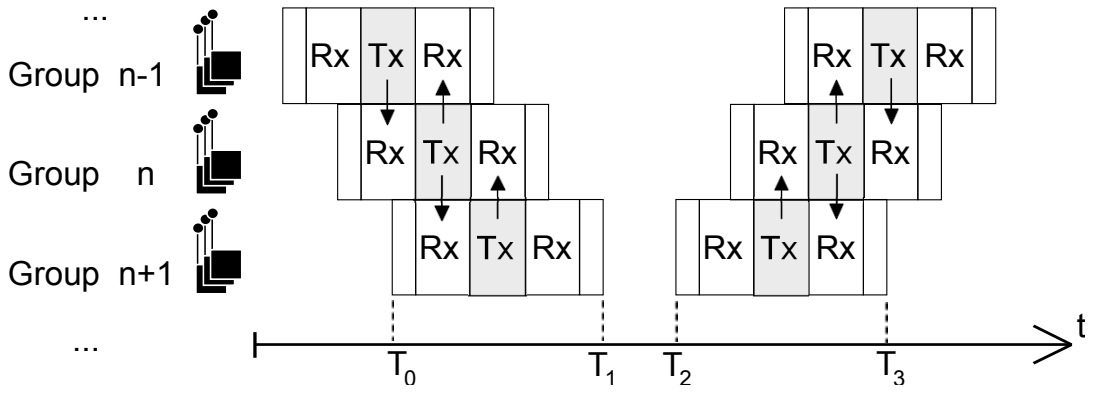


Figure 2. Overview of CUPID's sleep/wake-cycle scheme

length $d_{slotlen}[sec]$, i.e., during this time they access the medium via CSMA/CA. The overall number of sending slots equals the Network Diameter. After the last sending slot, there is a *Silence Period* of length $d_{sil}[sec]$ where no node is awake. In addition, we define the *Super Frame Length* $d_{SFL} = ND \cdot d_{slotlen} + d_{sil}$.

The designer can choose the proportion of waves towards and away from the reference node to support the expected data flow. For instance, a data collection application forwards much data towards a sink, so that the wave towards the sink should be employed much more often than the other one.

1) Coping with clock drift: Coordinated sleep/wake-cycles require synchronized clocks, which can – to a certain extent – be achieved using an appropriate protocol. However, between two clock synchronizations, the hardware clocks of the nodes drift unpredictably, but within known upper bounds. In order to prevent message loss because of sleeping neighbors, there are two different approaches: The first one is to let the nodes wake up d_{tol} ms earlier and go asleep d_{tol} ms later so that the receivers compensate for possible clock drift. This *Tolerance Length* d_{tol} is shown in Figure 3. The second option is to shorten the duration of each sending slot so that the senders compensate for clock drift.

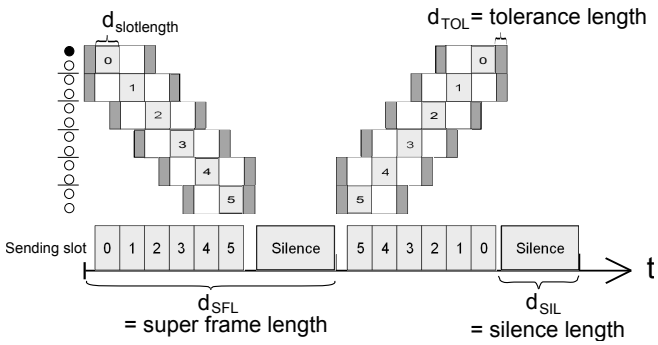


Figure 3. CUPID in detail

We decided to use the first option because the overhead of option two is $3 \cdot 2 \cdot d_{tol}$ ms in the worst case, where the tolerance period remains unused. This is three times larger than the other overhead of $2 \cdot d_{tol}$ ms. However, we must keep in mind that this decision results in slots directly following each other and unsynchronized clocks result in overlapping slots, which cause additional contention for medium access. The scheme is adaptable to unsynchronized clocks by adjusting the duration of the tolerance phase.

2) Mutual Parameter Dependencies: Not all combinations of Network Diameter, Duty Cycle, Slot Length and Super Frame Length (i.e., end-to-end delay) are possible. We show, which parameters have mutual dependencies and which value combinations are feasible.

As each node stays awake three sending slots – its own sending slot as well as the sending slots of preceding and succeeding group - the duty cycle DC results to

$$DC = \frac{3 \cdot d_{slotlen} + 2 \cdot d_{tol}}{d_{SFL}} \cdot 100. \quad (1)$$

Imagine a required EED of 300 ms and a Network Diameter of 30 hops (and therefore 30 sending slots). Then the maximum value for Slot Length is 10 ms with no Silence Period ($d_{sil} = 0$). Hence, $\frac{d_{SFL}}{ND}$ is the upper bound for Slot Length. Reducing the Slot Length for a given Network Diameter decreases the End-to-End Delay. The lower bound for Slot Length is the time required to transmit at least one single message. However, the Slot Length should allow for sending all desired messages. The more messages nodes have to send or forward, the longer the slots must be.

In addition, depending on the communication pattern it may be beneficial not to use equally long Slot Lengths but to adapt their value depending on the location of a node. For instance, if every node wants to send data to the sink without data aggregation, the message density increases exponentially with the proximity to the reference node. Hence, it can make sense to prolong sending slots (when the wave goes back to the sink) the closer the nodes are

to the sink node, so that they have more time for message forwarding.

The Network Diameter also limits the maximum possible value for the Duty Cycle for a given EED. To obtain this, it is again $d_{sil} = 0$. In the case of perfectly synchronized clocks ($d_{tol} = 0$) using Equation 1 we get:

$$DC = \frac{3 \cdot d_{slotlen} + 2 \cdot d_{tol}}{ND \cdot d_{slotlen} + d_{sil}} \cdot 100 \quad (2)$$

$$= \frac{3}{ND} \cdot 100. \quad (3)$$

For instance, with $ND = 5$, the maximum value of Duty Cycle is 60%, while for $ND = 100$ this upper bound goes down to 3%. For some applications it might be necessary to further reduce the Duty Cycle. This can be achieved by decreasing the Slot Length, which automatically increases the Silence Period since the Super Frame Length is fixed by the selected End-to-End Delay.

B. Arbitrary communication patterns

Until now, an implicit assumption has been that data flows from or to a reference node and therefore along the directions of the wave. Our scheme is optimized for this kind of data flow. In the directions of the wave, the delay for a message scheduled at random point of time is nearly equal for all nodes in the network and only depends on the path length to the reference node.

However, if two arbitrary nodes want to exchange messages, CUPID causes additional delay compared to other schemes (such as synchronous cycling). Consider two neighboring nodes N_1 of group 1 and N_2 of group 2. N_2 sends a message MSG_1 to N_1, which is supposed to respond with message MSG_2. In the inconvenient case, the wave goes from group 1 to n and on reception of MSG_1 N_1's sending slot is already over. Hence, MSG_2 has to wait for the duration of d_{SFL} until it will reach N_2. CUPID is especially suited for communication in two (preferred) directions, towards or from a reference node. Multi-hop communication orthogonal to the wave takes disproportionately more time.

C. Complex network topologies

More complex network topologies, as shown in Figure 4, lead to waves moving towards each other, which increases the probability of message loss due to hidden terminals. Consider the example: Here, two wave fronts moving away from the sink collide at node A and two wave fronts moving towards the sink collide at node B.

In complex topologies with vertices and holes CUPID can be applied, but some settings systematically lead to collisions. Representing the topology by a graph, we observe that cycles result in these settings and vertices if the wave moves towards the sink.

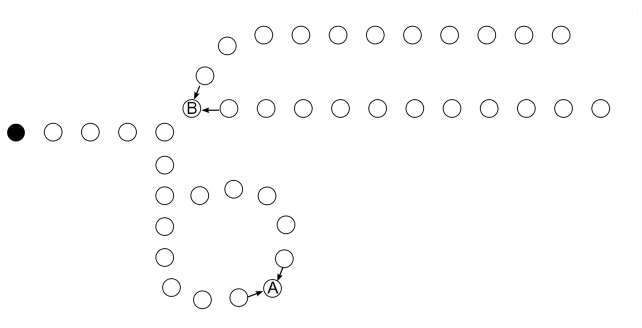


Figure 4. Example of a more complex topology

D. Installing CUPID at run-time

CUPID is divided into the start-up phase that is executed after network deployment and the subsequent operating phase describe above. We assume that the values for Duty Cycle and Tolerance Period are known at compile-time. However, for a node to know the wake-up, transmit and sleep times, it must determine (during the start-up phase) the number of groups and its group number. We briefly present one possibility to obtain the values.

- **Group numbers:** A way to assign the group number to each node is to use shortest paths to the reference node. Therefore, the reference node initiates a tree construction. If all messages contain their hop based distance, each node knows its distance to the reference node. Figure 5 shows an example how S_1 initiates a tree construction, how a possible constructed tree looks like, and how the nodes assign their respective group numbers.

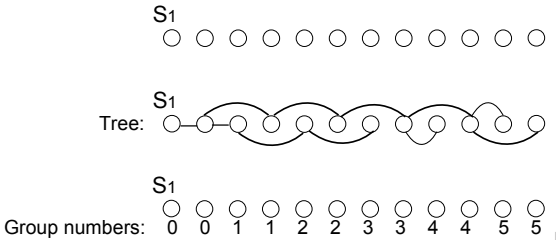


Figure 5. Assignment of group numbers

- **Overall number of groups:** If the network diameter cannot be estimated, each node sends a message to its parent node in the tree containing the maximum of its own group number and all previously received group numbers. The parent node forwards the message if the included value is greater than the actual local maximum. Like this, the maximum is forwarded to the reference node. In order to save messages, the further a node is located from the root of the tree the earlier it should send. The reference node then increments the maximum by one to obtain the overall number of

groups and floods a message containing this number to inform all nodes. If the diameter is estimated by the user, the estimation should be greater than the real diameter by any means to ensure that every node can use its slot number. An overestimation simply increases the silence period, but does not invalidate slot numbers.

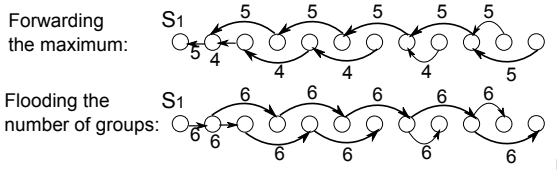


Figure 6. Detection of the number of groups

Algorithm 1 shows the computation of the slot and silence lengths during the start-up phase. First, the greatest possible slot length is determined as well as the resulting duty cycle. If this duty cycle is greater than the desired duty cycle, the slot length is recomputed using the desired duty cycle. Finally, the length of the silence period is computed.

Algorithm 1 Computation of CUPID's parameters

```
slot_len := (EED – 2*d_tol)/ND;
DC := (3*slot_len + 2*d_tol)*100/EED;
if DC > DC_DESIRED then
  slot_len := EED*DC_DESIRED/100;
  if slot_len > 2*d_tol then
    slot_len := slot_len – 2*d_tol;
    slot_len := slot_len / 3;
  else
    //Error: Invalid parameter combination!
    slot_len := 1;
  end if
end if
d_sil := EED – slot_len*ND – 2*d_tol;
```

In order to switch to the operation phase, the reference node floods another message containing the switching time T_C. In case of lower densities, but of higher risk of packet loss, additional broadcasts should increase the probability of a reliable message distribution. If new nodes join the network, they can either listen to the messages of their neighbors to learn their group number or the start-up phase could be repeated.

IV. Simulative Evaluation

To evaluate our approach, we conducted a set of simulations in the simulator Shawn [17] and compared CUPID to two other schemes representing the most commonly employed classes of duty cycling protocols. The first benchmark scheme called *SC-CSMA* (Synchronous Cycling with CSMA) is a simple synchronized duty cycling of all nodes, which access the medium by CSMA/CA during wake phases. Directly after waking up and before going to sleep, there is the same tolerance period like in CUPID to safeguard against clock drift. We call the second benchmark *SC-TDMA* (Synchronous Cycling with TDMA). Here, nodes are awake synchronously with the same tolerances like in both other schemes, but access the medium exclusively using a two-hop wide unique slot. To establish these slots we used the DRAND-algorithm [18].

Our goal was to model the real world as good as possible. The sensor nodes used for the real-world experiments are iSense nodes [19]. Software written for these nodes can be compiled for execution in Shawn, which provides also an IEEE 802.15.4 compliant CSMA/CA transmission model.

To model communication links, we employed the *stochastic communication model* where the packet reception probability remains constant at p_{max} up to the distance r_1 from the sender and decreases linearly from p_{max} to 0 for $r_1 < d < r_2$. We applied the values for $p_{max} = 98\%$, $r_1 = 28$ m, and $r_2 = 37.5$ m obtained from a set of real-world outdoor experiments.

The accuracy of the oscillators of iSense nodes is 20 ppm and by repeating time synchronization every 5 minutes, an offset of less than 12 milliseconds is guaranteed, so we set $d_{tol} = 12$ ms. To obtain statistically sound results, the results presented here are averaged over 100 simulations using the same parameter set but different random seeds.

A. Line-shaped topology

For our first simulations, we used a linear simulation area where nodes were placed in equal distances. We chose the following parameters of which we varied always one within each simulation set: $ND = 50$ hops, $DC = 1\%$ and the density was set to 10 neighbors on average to obtain a realistic working setup. As the influence caused by the chosen super frame length is expected to vary for different schemes, we altered the super frame length and then picked the best for each scheme.

At opposite ends of the simulation area, we placed two dedicated sink nodes S_1 and S_2 whereas S_1 triggered CUPID's installation as well as the operating phase start. We measured the EED of a single (flooded) packet from S_1 to S_2, which was scheduled at the beginning of one awake phase of S_1.

1) Influence of super frame length on latency: Figure 7 shows the average EED over the super frame length for the different duty cycling schemes. The lines indicate how the latency develops with an increasing super frame length. The vertical bars indicate minimum and maximum occurring value. Note that the larger the super frame length is, the larger is the awake phase, the slot length, and thereby the time for sending or forwarding a packet.

First, we have a look at the results of CUPID. For super frames shorter than 8 s, it depends on the random seed how

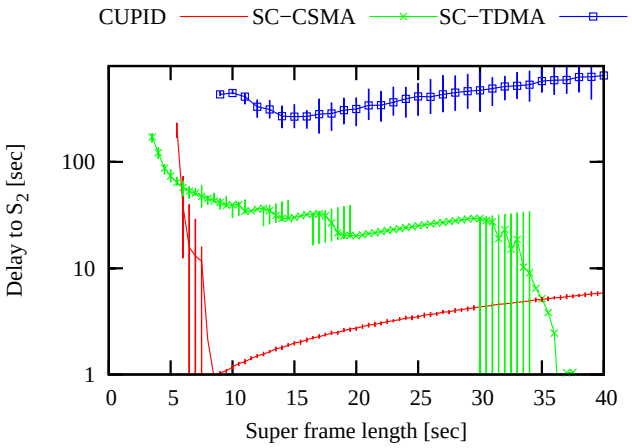


Figure 7. Average latency in the best case (diameter 50)

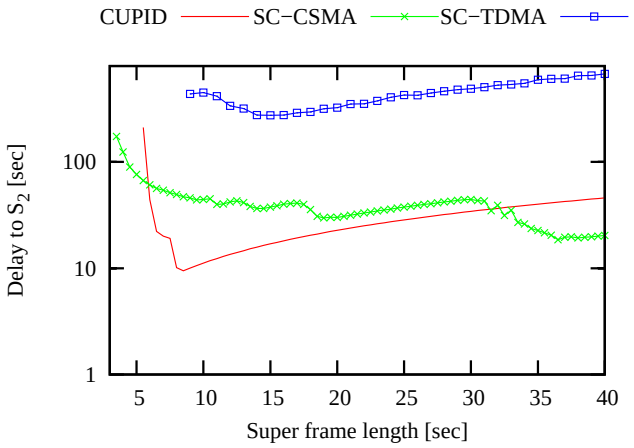


Figure 8. Average latency for an alarm message (diameter 50)

many cycles the flooded packet needs to arrive at the end of the network. At most it takes about 50 s, which corresponds to 7 cycles. The longer the wake phase is, the more likely it becomes that the packet is passed through the whole network within one cycle. 8 s as super frame length is obviously enough for the network diameter of 50 hops. The packet forwarding is done in only 942 ms. Increasing the super frame length leads to larger slot lengths and in particular in later slots (except the first one). This is why the delay of the packet slowly rises with an increasing super frame length.

The curve of the SC-CSMA scheme proceeds in stages because the message needs several awake phase to pass through the whole network and the longer the awake phase is the more hops are bypassed within one cycle. For larger super frame lengths than 40 s, the EED remains low because one awake phase is long enough to forward the packet over 50 hops. While CUPID features a delay between 900 ms and 5 s for super frame lengths between 10 s and 40 s, for SC-CSMA it becomes that low not before a super frame length of 38 s. This is especially harmful if the packet to be sent is an alarm message, which can be scheduled at any point in time. Hence, just in the case of an alarm where a low delay is important such a great super frame length affects the SC-CSMA in a negative way.

Figure 8 shows the average latency of an alarm message. Note that the time needed for forwarding the packet can be relatively short in comparison to the additional latency caused by waiting until neighbors have woken up. For CUPID it is 942ms vs. 10 s, which are caused by considering the random schedule time of an alarm.

SC-TDMA needs significantly more time (at least 260 s) to deliver the packet to S_2 because sending slots are not ordered in the forwarding direction. Hence, in the worst case the packet is forwarded only once during one wake phase. If it is clear, that the principle occurring communication pattern consists of messages in one direction, it is possible to adapt CUPID, so that the respective cycle is processed more frequently than the other cycle and the average EED is reduced further.

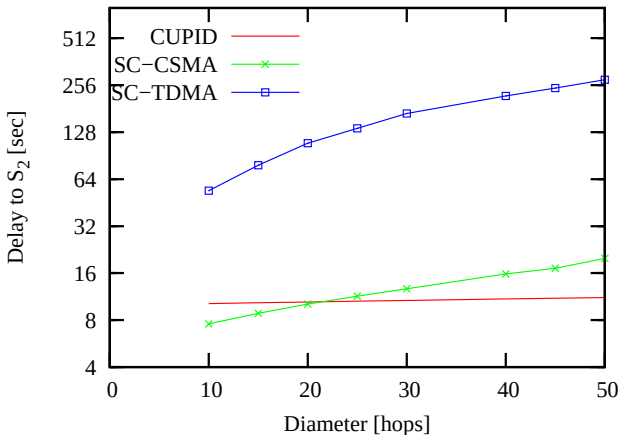


Figure 9. Average latency over diameter

2) Influence of network diameter on latency: Figure 9 illustrates the latency in the average (alarm) case over different diameters. We chose the optimal super frame length here for each scheme and each diameter. For CUPID, we set $d_{SFL} = 8$ s whereas for the other schemes $d_{SFL} = diameter + 20$ s holds.

We notice that in smaller networks with a diameter of less than 20 hops CUPID and SC-CSMA perform similarly, while SC-TDMA in contrast does not achieve the same efficiency. For larger diameters than 20, CUPID outperforms the other schemes.

3) Influence of density on latency: In contrast to different network diameters, other densities than 15 neighbors on average do not influence the latency of the packet very much. We set the density to 6, 10 and 20, but the results are similar

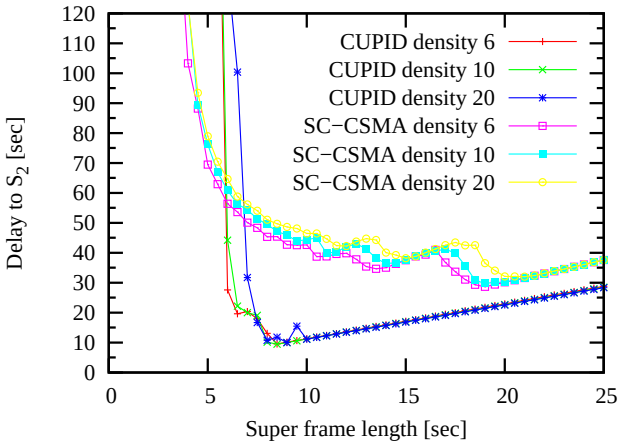


Figure 10. Influence of density

for all schemes. This is barely surprising because a higher density hinders the medium access, but hardly the latency until a packet is forwarded as always at least one node in every hop forwards the message.

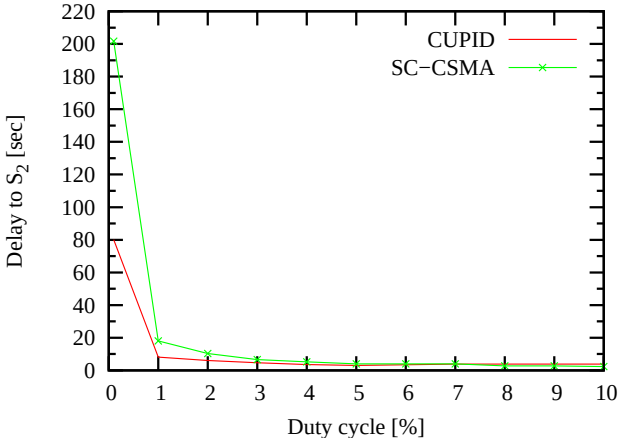


Figure 11. Average latency over duty cycle

4) Influence of duty cycle on latency: Finally, we investigated the influence of the duty cycle. Figure 11 depicts the latency of the packet for different duty cycles. It becomes clear that CUPID is beneficial for duty cycles lower than 2%. If energy is not scarce and a higher duty cycle may be used, it does not make sense to use CUPID, but especially for even lower duty cycles than 1% CUPID is extremely advantageous.

The following section presents the performance analysis of CUPID in the more general case, a two-dimensional topology.

B. Rectangle-shaped topology

The line-shaped topology may occur at borders or dikes, but to evaluate the performance of CUPID in other networks we conducted simulations with the scenario shown in Figure 12 comprising different structures. We placed 900 nodes in equal distances of 15 m in the simulation area so that we obtained about 10 hops in vertical and 30 hops in horizontal direction and about 20 neighbors on average.

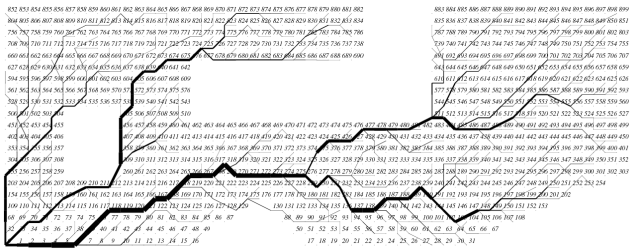


Figure 12. Rectangle scenario

Based on the assumption that sending measured data to a sink is a typical communication pattern in sensor networks, we let nodes send messages to the sink node in the lower left corner (using a tree routing). The resulting message density on each link is indicated through the thickness of the lines in Figure 12. We let different subsets of nodes (from 1 % to 20 %) of all 900 nodes send packets to the sink during the same wake phase. Each node of the subset sends one packet during this special wake phase resulting in 9 to 180 packets. We counted how many of these arrived at the sink and logged the delay. The super frame length was again set to 8 s. Along the tree, single hop acknowledgements are used, where each message is sent at most three times and is discarded after three tries. The lack of multi-hop acknowledgements leads to unreliable multi-hop communication.

The following two sections show the evaluation results concerning delivery ratio and the latency. We do not consider the SC-TDMA scheme in this scenario as it yielded in unacceptable high latencies.

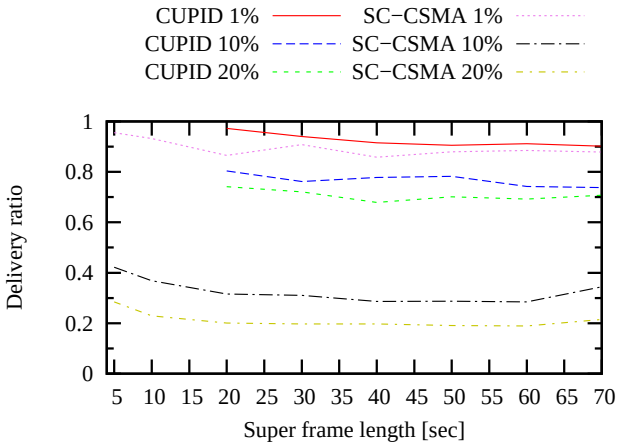


Figure 13. Delivery ratio

1) Delivery ratio: Figure 13 shows the receiving rate at S_1 for the different numbers of sent packets. SC-CSMA provides for all the percentages worse results. When 9 packets are sent, between 90 % and 95 % arrive at their destination. When 90 or 180 nodes send a packet only 30 % to 40 %, respectively 20 % to 30 %, arrive at S_1. The reason for this is that nodes that cannot hear each other send potentially at the same time to the same parent node leading to systematic collisions.

By contrast, CUPID transports 90 % to 98 % of the messages to S_1, if 1 % of the nodes send a message and between 70 % and 80 % if 90 or 180 nodes send a message. The regulated sending increases the delivery ratio, especially in case of high traffic volume.

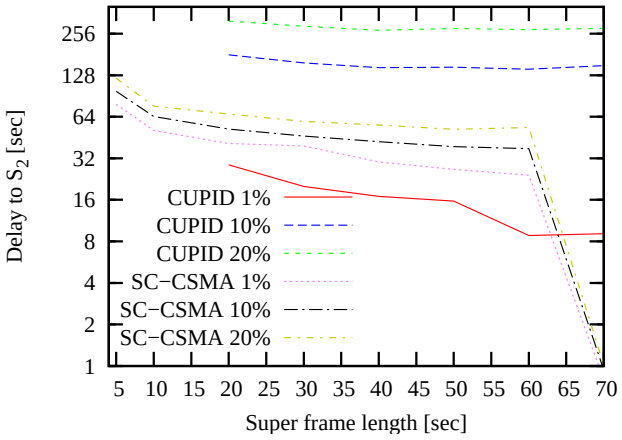


Figure 14. Average latency

2) Latency: The previous section showed that CUPID provides a good delivery ratio. Figure 14 shows the corresponding averaged latencies of the successfully delivered messages. Both schemes provided similar delivery ratios if only 9 messages are sent, but CUPID provides the shorter delays. Additionally, in case of a message scheduled at a random point in time, we must add the half of the super frame length on average. Moreover, shorter super frames allow for an earlier sending of another message. Using SC-CSMA a slot length of 700 ms is sufficient to transport the messages to the sink within one wake phase. As soon as the slot length is smaller the delay increases from 24 to 78 s. CUPID need only between 9 and 28 s.

The smaller values of the SC-CSMA scheme for 90 and 180 sent messages show that after a short time of traffic with many collisions no packets are forwarded or delivered anymore. CUPID delivers messages within up to 180 s, respectively 315 s. Of course, the more messages are sent the longer the delivery takes. However, CUPID reduces the probability of collisions since fewer nodes sent at the same time.

V. Real-world Experiments

In order to demonstrate the correctness of CUPID we conducted a series of real-world experiments using 156 iSense sensor nodes. As a comparison scheme we picked the better one of the benchmark schemes (SC-CSMA).

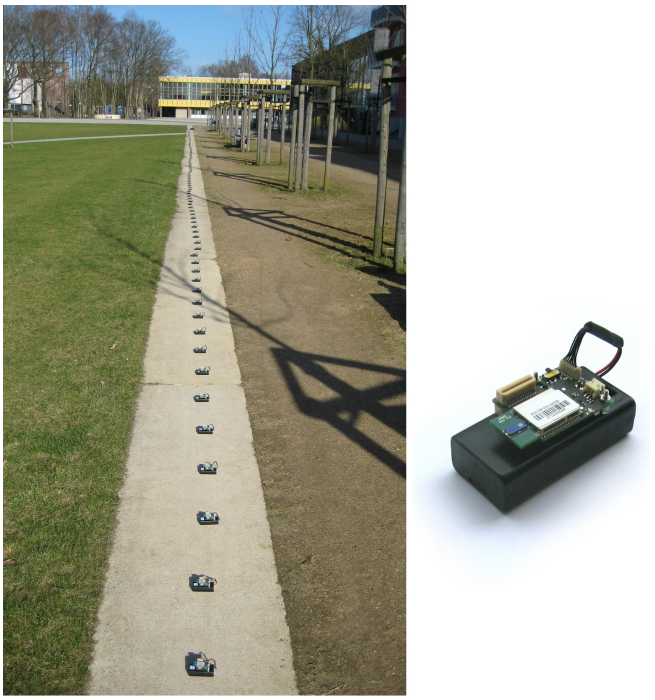


Figure 15. Experimental scenario (Second place)

In order to get a large network diameter, we reduced the sending power of the nodes by -30 db since otherwise, the length of the network would have been more than $1,5$ km. Using this sending power, the nodes were deployed in a row with an equal distance of 40 cm. We deployed the devices at two different places. First place was near a building where we got only 12 hops (due to many reflections and varying ranges), the second deployment was in a free area where we obtained 23 hops. After time synchronization and initialization we let one gateway node flood one packet at the beginning of a wake phase (leading to the best case) and logged its sending and arrival time at the farthest node. Figure 16 shows the results of this EED averaged over 120 packets.

It can be seen that in the first deployment near the building (diameter 12) the SC-CSMA scheme performs better as it can benefit from simultaneous wake phases. Only here are the temporarily larger communication ranges resulting in a smaller network diameter.

In the larger scenario without reflections CUPID performs better: It needs only 500 ms for the delivery of the packet, SC-CSMA needs 900 ms. Taken into account that an average offset of 10 seconds has to be added to these values for the

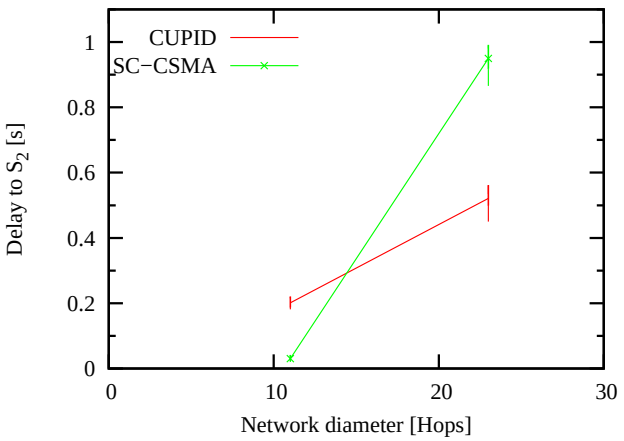


Figure 16. Latency for diameter 12 and 23

average case of an alarm packet, the results match exactly the simulation results shown in Figure 9.

VI. Conclusion

In this paper, we motivated, introduced, and evaluated a novel protocol called CUPID for low-delay, low-power, and attack-resilient operation of wireless sensor networks. Application domains include public safety or smart cities where large-scale, long-term, battery-driven operation, low-delay reporting of events and operation in potentially hostile environments are important. Our scheme is beneficial for any application that requires low message delivery latency and low duty cycles of 1% or even less at the same time.

CUPID is adaptable to different in-network communication patterns by the application designer and can be custom-tailored to application demands. In particular it supports communication from sink-to-network and vice versa, but is adaptable to the predominant communication pattern, duty-cycle settings and latency requirements. The scheme ensures that all neighbors of sending nodes are ready-to-receive while only assuming synchronized clocks to a certain extent. We show by simulations and real-world experiments with more than 150 nodes that our scheme significantly reduces the packet delay for low-duty cycle settings, especially in large networks.

While CUPID deals successfully with changing network structures caused by new or removed nodes as well as with unreliable links, it cannot cope with mobile nodes or very dynamic communication ranges. In future work, we will extend CUPID to semi-mobile scenarios where some nodes are mobile but the majority is static. In addition, we plan to evaluate CUPID in a larger-scale testbed such as SmartSantander, which will eventually support more than 10.000 nodes.

References

[1] D. Krüger, D. Pfisterer, and S. Fischer, “CUPID - Communication pattern informed Duty Cycling in Sensor Networks,” in *The Fifth International Conference on Systems and Networks Communications (ICSNC 2010)*. IEEE Computer Society Conference Publishing Services, 8 2010.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, “A line in the sand: a wireless sensor network for target detection, classification, and tracking,” *Computer Networks*, vol. 46, no. 5, pp. 605 – 634, 2004, Military Communications Systems and Technologies.

[3] C. Gui and P. Mohapatra, “Virtual patrol: a new power conservation design for surveillance using sensor networks,” in *Proceedings of the 4th international symposium on Information processing in sensor networks*, ser. IPSN ’05. Piscataway, NJ, USA: IEEE Press, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1147685.1147728

[4] P. Rothenpieler, D. Krüger, D. Pfisterer, S. Fischer, D. Dudek, C. Haas, A. Kuntz, and M. Zitterbart, “Flegsens - secure area monitoring using wireless sensor networks,” *Proceedings of World Academy of Science, Engineering and Technology*, 2009.

[5] Cambridge (MA) Smart City, “CitySense - an open, urban-scale sensor network testbed,” 2010, http://www.citysense.net [Last accessed: May 17, 2011].

[6] “Oulu Smart City,” 2010, http://www.ubiprogram.fi [Last accessed: May 17, 2011].

[7] German Telekom and City of Friedrichshafen, “Friedrichshafen Smart City,” 2010, http://www.telekom.com/dtag/cms/content/dt/de/217308 [Last accessed: May 17, 2011].

[8] The SmartSantander consortium, “SmartSantander,” 2010, http://www.smartsantander.eu/ [Last accessed: May 17, 2011].

[9] Y. Li, W. Ye, and J. Heidemann, “Energy and Latency Control in Low Duty Cycle MAC Protocols,” USC/Information Sciences Institute, Technical Report ISI-TR-595, August 2004.

[10] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, “Towards optimal sleep scheduling in sensor networks for rare-event detection,” in *IPSN ’05: Proceedings of the 4th international symposium on Information processing in sensor networks*. Piscataway, NJ, USA: IEEE Press, 2005, p. 4.

[11] N. Bisnik and A. A. Abouzeid, “Delay and capacity in energy efficient sensor networks,” in *PE-WASUN ’07: Proceedings of the 4th ACM workshop on Performance evaluation of wireless ad hoc, sensor,and ubiquitous networks*. New York, NY, USA: ACM, 2007, pp. 17–24.

[12] A. Keshavarzian, H. Lee, and L. Venkatraman, “Wakeup scheduling in wireless sensor networks,” in *MobiHoc ’06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 322–333.

[13] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. Le Roux, "Poster abstract: wiseMAC, an ultra low power MAC protocol for the wiseNET wireless sensor network," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003, pp. 302–303.

[14] L. Gu and J. Stankovic, "Radio-triggered wake-up capability for sensor networks," in *RTAS '04: Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*. Washington, DC, USA: IEEE Computer Society, 2004, p. 27.

[15] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[16] Y. Taniguchi, N. Wakamiya, and M. Murat, "A traveling wave based communication mechanism for wireless sensor networks," *Journal of Networks*, vol. 2, pp. 24–32, 2007.

[17] S. P. Fekete, A. Kröller, S. Fischer, and D. Pfisterer, "Shawn: The fast, highly customizable sensor network simulator," in *Proceedings of the Fourth International Conference on Networked Sensing Systems (INSS' 07)*, Jun. 2007.

[18] I. Rhee, A. Warrier, J. Min, and L. Xu, "Drand: distributed randomized tdma scheduling for wireless ad-hoc networks," in *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 190–201.

[19] coalesenses GmbH, "iSense," http://www.coalesenses.com [Last accessed: May 17, 2011].