A Slot Sharing TDMA Scheme for Reliable and Efficient Collision Resolution in WSNs

Philip Parsch and Alejandro Masrur
Department of Computer Science
TU Chemnitz, Germany

ABSTRACT

With the advent of Internet of Things (IoT), an increasing number of devices may spontaneously communicate to exchange information. This puts emphasis on wireless sensor networks (WSNs) and, in particular, on intelligent medium access control (MAC) protocols, as there is a need to guarantee a certain quality of service (QoS) on timely data/packet delivery. Most existing approaches, however, are either of random nature, making it impossible to guarantee any bounded delay, or do not scale well for a higher number of nodes. As a result, we propose slot sharing TDMA (short s^2TDMA), a deterministic contention resolution scheme in form of generating TDMA cycles with shared slots at the event of collisions. Every TDMA slot is assigned to a range of IDs, in which the corresponding nodes can transmit. By further dividing these slots in case of collisions, we implement an *interval tree search* enabling for fast collision resolution in $\log_{\hat{s}}$-complexity, where $\hat{s}$ is the number of slots in each cycle. Since our scheme is activated upon collisions, it incurs in zero overhead during normal operation and is able to quickly react to changing traffic load such as bursty traffic. We perform a large set of detailed simulations on OMNeT++ showing that our technique offers a fast collision resolution and is able to handle a large number of nodes in the network.

CCS Concepts

•**Networks** → **Network protocol design;** *Network reliability; Sensor networks;*

Keywords

Medium Access Control; delay bounded communication; reliability; receiver-initiated

1. INTRODUCTION

In the era of IoT, different devices will spontaneously communicate with each other to exchange information allowing for new interesting applications. For example, wearables can communicate with existing smart home structures to improve comfort and user experience or even call an ambulance in emergency situations.

This, however, poses a number of challenges that need to be addressed. In particular, there can be a potentially large number of nodes transmitting on a communication channel and, hence, leading to increased collisions and data loss.

While we expect IoT applications to tolerate some data loss, there is a need to guarantee a certain QoS on which they can rely. That is, devices should be able to reliably deliver data in a timely manner, which needs to be enabled from the MAC layer upwards.

In this context, CSMA-based approaches seem to be suitable, as these do not require devices to synchronize and, hence, offer good energy efficiency. In addition, they are known to be very effective in handling retransmissions in case of collisions at the communication channel. However, their effectiveness drastically reduces at high load, e.g., as the number of transmitting devices increases. That is, the retransmission delay potentially becomes unbounded and, therefore, no QoS guarantees can be given.

On the other hand, TDMA-based approaches allow for a reliable communication at high loads, but they lack the necessary flexibility, particularly, for dynamic reconfiguration — devices may join and leave the network at arbitrary points in time — apart from relying on devices to synchronize.

Hybrid MACs [2] [9] have been proposed to overcome this problem. These are based on combining CSMA and TDMA to allow for deterministic behavior, especially, at high loads and, at the same time, for reconfiguration flexibility.

Although these hybrid approaches allow guaranteeing QoS at high loads, they do not easily scale to an increasing number of devices or nodes in the network. In other words, these are effective when a few nodes produce high communication load, but rapidly degrade when multiple nodes are responsible for that communication load, as discussed later in more detail. In addition, they require control messages to be exchanged adding further delay and making them react slowly, which can be problematic for bursty traffic.

In order to allow for faster response times, other approaches have been presented [5] [7] [10]. These rely on collision resolution schemes that schedule retransmissions and therefore avoid further interference between nodes. Since they just activate on collisions, they do not result in any overhead during normal operation and allow for fast data transfers. However, they are either of random nature again making any QoS guarantees impossible, or only work for a small number of nodes.

MSWiM '16, November 13-17, 2016, Malta, Malta
© 2016 ACM. ISBN 978-1-4503-4502-6/16/11…$15.00
DOI: http://dx.doi.org/10.1145/2988287.2989164

Similar to the previously mentioned approaches, this paper proposes a hybrid MAC. However, in contrast to them, our approach better scales to an increasing number of nodes. This makes it more suitable for upcoming IoT and, in general, for WSN applications where large numbers of nodes are to be expected.

1.1 Contributions

In this paper, we propose s^{2}TDMA, a hybrid MAC that allows deterministic collision resolution in receiver-initiated WSNs, i.e., those where nodes only send when awakened by the receiver. Our technique consists in generating TDMA cycles — called arbitration cycles — upon collisions. The particularity of the proposed scheme is that every TDMA slot is assigned to a range of IDs, in which the corresponding nodes can transmit, i.e., slots are shared by multiple nodes.

Whenever a collision occurs, the current slot is further divided into multiple sub-slots with smaller ID ranges. This enables fast collision resolving in log_x-complexity being x the number of slots per each such cycles as illustrated later in detail.

In addition, we analyze the probability of collision and the worst-case communication delay according to the proposed scheme. Finally, extensive simulation results are presented based on OMNeT++. These compare the performance and show benefits by s^{2}TDMA with respect to other approaches from the literature.

1.2 Structure of the Paper

The rest of this paper is structured as follows. Related work is discussed in Section 2. Next, Section 3 explains our system model and assumptions. Section 4 introduces the proposed MAC protocol and Section 5 analyses collision probabilities and communication delays mathematically. Section 6 presents our experimental evaluation based on simulation and Section 7 concludes the paper.

2. RELATED WORK

There are many different approaches from the literature that are concerned with making WSNs more reliable and energy-efficient. In general, these can be classified into contention-based, reservation-based and hybrid schemes.

Contention-based methods are usually asynchronous protocols that can access the channel at arbitrary points in time. Whenever a collision occurs or the channel is busy during carrier sensing, they perform techniques to resolve the contention, such as random back-off schemes. Due to their low overhead, high flexibility and simplicity, they enjoy great popularity. However, since channel access is uncoordinated, contention-based approaches generally offer bad performance for high traffic loads. A widely used contention-based protocol is CSMA.

In contrast, reservation-based protocols avoid collisions by assigning nodes individual time slots to transmit. This ensures that even high numbers of nodes can transmit their data reliably and within a bounded delay. Nevertheless, reservation-based protocols generally require synchronization, which incurs in additional overhead and worsens the energy efficiency, especially during low network load. Examples of synchronous networks are TDMA and slotted Aloha [1].

Hybrid approaches try to combine the advantages of both contention and reservation-based protocols. This can, for example, be done by switching from CSMA to TDMA in high traffic [9] or by leaving space in TDMA frames, where nodes can transmit packets with CSMA [2]. However, both methods require additional control messages for switching modes or assigning TDMA slots. Since these messages can also be lost, no real-time guarantees can be made. Further, they impose an additional delay making these schemes react only slowly to changes in traffic load.

Another contention-based approach, called Strawman, is presented in [7]. Here, nodes actively contend after collisions by sending a contention packet of random length. The receiver then selects the node with the longest contention packet and transmits an RTS-like decision message. After the actual data is transmitted, the arbitration starts anew until all collisions are resolved. In summary, Strawman allows fast adaption to changing traffic, as well as a fast collision resolution. However, its random nature makes it not suitable for real-time applications.

Similar to Strawman [7], STAIRS [5] uses active contention messages after every collision. However, instead of just picking the longest packet out of a number of contention packets, the RSSI channel is used to determine the number of contenders and, hence, create a schedule. This greatly reduces the overhead, as contention packets have to be transmitted only once in the best case. On the other hand, the use of the RSSI channel is highly error-prone and works only for a low number of contenders, as it quickly starts to saturate.

In order to manage higher number of contenders, for example, in dense sensor networks, Carlson et al. propose Flip-MAC [3]. Here, the receiver first reduces the number of contenders by a series of probe-acknowledgment cycles. In every cycle, each node randomly picks one of two possible IDs and all nodes matching the ID of the probe message send an acknowledgment while all others drop out. The receiver repeats this cycle until no more ACKs are received indicating that contenders have been reduced to a manageable level. The remaining contenders then use CSMA to transmit data.

The previously mentioned contention-based approaches were of random nature enabling fast collision resolution, but making it impossible to guarantee any bounded delay. Since some applications are delay sensitive and require deterministic behavior, different MAC protocols are needed. To this end, BIN-MAC [10] proposes a hybrid protocol that allows a delay-bounded contention resolution. More precisely, every node is assigned a unique ID and every time a collision occurs, the receiver replies with a negative acknowledgment (NAK) containing a range of IDs. Only nodes with an ID within that range can retransmit, all others have to wait. Similar to a binary tree search, this range is halved upon every collision until data is successfully transmitted. This results in a log_2-complexity allowing fast collision resolutions even for high number of contenders.

In this paper, we propose s^{2}TDMA, a deterministic contention resolution scheme similar to BIN-MAC [10]. However, instead of using a binary tree search, we implement an interval tree search in form of generating TDMA cycles upon collisions. This does not only speed up the collision resolution, but also allows multiple nodes to send consecutively within a cycle. We later show that our approach greatly improves the performance, such as latency, throughput, etc. of the network, which makes it, together with its high scalability, be better suited for larger networks.

3. MODELS AND ASSUMPTIONS

We consider a WSN consisting of multiple sensor nodes and one or more sink nodes that are spatially distributed. Upon activation, nodes do not transmit spontaneously, but wait for the next query/probe message from their corresponding sinks. This receiver-initiated topology, which is also used by similar approaches from the literature [3] [5] [7] [10], offers two main advantages: First, it allows the usually very resource constrained nodes to put most of the burden on the receiving node. Second, it limits the number of contending nodes as they can only participate in the communication cycle after a receiver's probe. This implicit sender synchronization prevents nodes from joining during later MAC phases, where they might interrupt the ongoing arbitration process.

In order to avoid conflicts between multiple sink nodes, we reduce contention by using multiple radio channels for operation. Similar to A-MAC [4], the initial probe messages are sent on a pre-determined channel to be receivable for all nodes. Later data transfers are then performed on other radio channels as defined in the probe message.

Transmitting a data packet takes a given amount of time, which depends on the number of bits to be transmitted and the bandwidth of the communication channel. We refer to this time to as *packet length* and denote the length of any packet of node i by l_i.

Since probe messages can wake up multiple nodes at once, data transmissions may interfere with each other leading to packet loss. As a result, to achieve reliability, the receiver node acknowledges (ACK) a packet after successful reception. We denote by l_{ack} the length of an acknowledgment packet. If the data packet is corrupt — we assume that even the slightest overlapping of two different data packets leads to data loss — the sink node replies with a negative acknowledgment (NAK). For simplicity, we set the length of NAK packets to be equal to $l_{nak} = l_{ack}$.

Every node in the system has a unique ID, which allows the sink node to distinguish between different data transmissions and to assign TDMA slots as described later. These IDs can be either be hard-programmed into the nodes memory, or be assigned dynamically by registering and de-registering in the network. However, in this paper we focus on collision resolution mechanisms and the assignment of IDs is out of scope.

4. PROPOSED SCHEME

In this section, we introduce s^{2}TDMA, a reliable and energy-efficient collision resolution scheme for receiver-initiated WSN. As already mentioned, this consists in generating TDMA cycles upon collisions. Every TDMA slot is assigned to a range of IDs, in which the corresponding nodes can transmit. Whenever a collision occurs within a slot, it is further divided into sub-slots to resolve that collision. This allows fast collision resolution as well as multiple consecutive data transmissions in a single arbitration cycle.

4.1 Working Principle

The communication cycle starts when a sink node broadcasts a probe message. All sensor nodes — addressed in the probe — may start transmitting their data after the probe message. However, since this can potentially trigger multiple sensor nodes at once, there can be collisions of packets. In this case, the proposed collision scheme is activated as displayed in Fig.1.

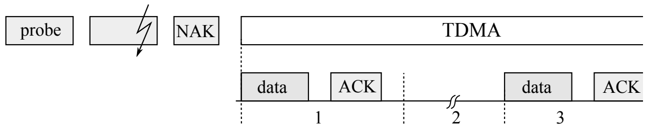

Figure 1: Different steps of the proposed contention resolution scheme: A probe message triggers 2 nodes, whose packets collide at first. After receiving a negative acknowledgment (NAK), these two nodes participate in a TDMA arbitration cycle and successfully transmit their data.

If corrupt data is received, the sink node replies with a negative acknowledgment (NAK) to inform the sensor nodes that they have to retransmit. However, the receiver does not know the number of contending nodes a priori. If these retransmit directly and without any resolution scheme, multiple collisions can occur again. To overcome this problem, a TDMA arbitration cycle is started.

Each TDMA slot is assigned to a range of IDs in which only nodes with an ID in that range are allowed to transmit. Assuming we have a system with a set of N nodes each of which has its unique ID, the initial TDMA arbitration cycle equally splits those IDs upon $\hat{s}$ slots. For example, if we have a system of 100 nodes and therefore IDs 1 to 100, each slot would contain $|N|/\hat{s} = 100/4 = 25$ IDs, with $\hat{s} = 4$ and $|N|$ being the number of nodes in N. The first slot contains the lower IDs, in this case 1 to 25, the second contains IDs 26 to 50, the third 51 to 75 and the fourth 76 to 100. Note that whenever $|N|/\hat{s}$ produces a remainder, this is added to the last slot.

This ID splitting reduces the number of contending nodes, however, it does not yet guarantee successful transmission. To this end, a TDMA slot is further divided into equal sub-slots, whenever a collision occurred within that slot. In our previous example, a collision within slot 1 would split this slot into another $\hat{s} = 4$ sub-slots with the ID ranges of 1-6, 7-12, 13-18 and 19-25. As a consequence, the number of contending nodes is reduced from 100 to 6 after two collisions. This greatly reduces the chance of further collisions, however, it still cannot fully prevent them. In the worst case, this splitting continues until each slot contains just one ID and therefore guarantees successful communication. If, in that case, the range of IDs to be divided is smaller than $\hat{s}$, the resulting TDMA cycle will have as many slots as IDs. For example, if a slot contains 3 IDs and $\hat{s} = 4$, this slot is divided into $k = 3$ sub-slots with one ID each.

Fig 2 shows the working principle of the sub-slot creation in more detail. Here, two nodes collided first and, hence, a TDMA arbitration cycle is created. In this example, after the first empty slot has finished, they both transmit in slot 2, in which their data packets collide again. The sink then replies with a NAK indicating that a sub-cycle begins with $\hat{s} = 4$ sub-slots. Now, due to the finer ID resolution, node i can successfully transmit in slot 2 and node j in slot 3. After the last slot of the sub-cycle has finished, another new cycle is started containing the remaining IDs that have not been resolved so far. Although there are no further nodes wanting to transmit in this example, empty slots cannot be skipped and the sink will go to sleep after slot 4 of cycle 2 has finished.

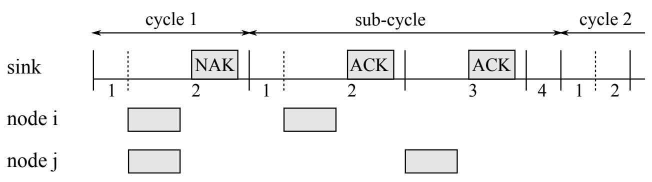

Figure 2: Illustration of the sub-cycle splitting principle: A collision within a TDMA cycle, here in slot 2, causes the immediate creation of a sub-cycle with $\hat{s} = 4$. Once these slots have been executed, the remaining IDs are processed in another full cycle (cycle 2). Note that all slots, independent of being empty or used, have the same length and have only been shortened for better illustration.

Note that whenever a sub-cycle is completed, i.e., no more collisions occurred in it, a new cycle is created to resolve the remaining IDs. This is a simple and efficient solution, since other methods such as going back the interval tree, i.e., stepwise increase the slot sizes again, would require many ACK/NAK notifications. Recall that at this point of time, the sink does not know the number of nodes participating in the arbitration cycle, i.e., how many nodes have been awakened by the initial probe message, but just the already resolved IDs from previous (sub-)cycles. Therefore up to $|N| - x$ nodes can still be pending, where $|N|$ is the total number of IDs and x the number of already resolved IDs. In summary, this process can be repeated until no IDs are left to resolve, in which case the sink goes back to sleep.

All parameters needed for generating the arbitration cycle, such as the ID ranges, number of slots per cycle $\hat{s}$, etc., are specified and calculated by the sink. These are then broadcast in every ACK/NAK message to inform the contending nodes. This can be done in two possible ways: First, ACK/NAK messages only contain the current ID range of the nodes that are allowed to transmit in the next slot. This reduces complexity at the nodes side, since they only need to check whether their IDs match the given range. However, it also requires more control bytes to be send within the ACK/NAK messages as discussed later. Second, either no or only very little information is broadcast and nodes use hard coded parameters. The missing ID ranges are then calculated depending on whether ACKs or NAKs are received. This reduces the data overhead of control messages, but on the other hand, also decreases the flexible of the system regarding dynamic changes.

By assigning IDs to TDMA slots, nodes are prioritized according to these IDs. This means that lower IDs are resolved faster, whereas higher IDs can take a longer time to be resolved. This is useful in applications, where different types of nodes have different requirements regarding their maximum tolerable delay. For example, in a home automation system, critical devices such as controllers or alarm systems need to relay their data faster than lower priority devices such as wireless light switches.

4.2 Timings and Data Structures

Every action performed by the transceiver requires time and, hence, influences the total delay of the proposed scheme. Fig. 3 depicts the timings of a successful packets transmission.

After a data packet is transmitted, both the transceivers of the sink and the sensor node have to switch their operation mode: The sensor node switches to receive mode in order to be able to receive an ACK/NAK, whereas the sink node switches to transmit mode to transmit the ACK/NAK. However, the sink node has to safely detect the end of node i's data packet first. Although data packets include a length code, as discussed later, this may corrupt in case of a collision and, hence, the receiver has to sense the channel continuously to detect the end of a packet. We denote this delay by sensing and switching operation modes by t_d.

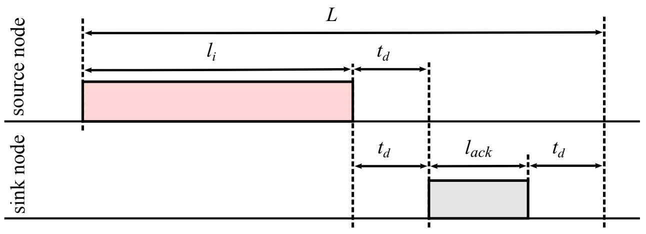

Figure 3: Timing of a successful transmission: After every data packet of length l_i, t_d is needed to process the data and to switch the operation modes of transceivers. Similarly, ACK messages (of length l_{ack}) require a processing time t_d.

After reception/transmission of an ACK/NAK, again both nodes have to switch their transceiver modes. Although ACK/NAK messages are of constant length and, hence, sensing the end of the packet is not necessary, we still assume the same t_d for this delay. This does not only facilitate the system analysis, but also adds the possibility to dynamically change the payload of ACK/NAK messages, which might be required for future extensions.

Similar to other approaches from the literature , e.g., BinMAC [10], s^{2}TDMA requires additional structure fields to be sent within each packet. Besides the usual overhead, such as preamble, start frame delimiter (SFD), etc., each data packet contains a length code to define the number of payload bytes and a cyclic redundancy check (CRC) field to detect corrupt data. In case of ACK/NAK messages, we do not need the length code, since we previously assumed that both have the same length $t_{nak} = t_{ack}$, which is already known by sensor nodes. However, each NAK/ACK contains a type field defining if it is either a NAK or an ACK and an ID field for the ACK address. In addition, as discussed previously, further bytes are required for informing sensor nodes about the arbitration cycle. These can be two address fields defining the upper and lower bound on IDs or either no data or just the number of slots per cycle $\hat{s}$. In summary, this means an overhead of 2 *bytes* for data packets and 3 to 7 *bytes* for ACK/NAK messages, when assuming a size of 1 *byte* for control fields and 2 *bytes* for address fields.

5. ANALYSIS OF COMMUNICATION

The proposed s^{2}TDMA can be easily configured to meet desired requirements. In particular, at the event of a collision on the communication channel, we can adjust (i) the number of slots per arbitration cycle and (ii) the number of (sender) nodes in each slot. Clearly, this affects communication delay, i.e., the time taken from a node's activation to its packet being successfully delivered.

There are different policies one can follow to select (i) and (ii). For example, one slot can be dedicated to one node — guaranteeing that this node sends alone on that slot — and assign more nodes to other slots or one can also uniformly assign nodes to slots. The more nodes are sending on the same slot, the higher the probability of collision on that slot. On the other hand, having less nodes per slot increases the number of slots, since all nodes need to be accommodated in one arbitration cycle.

The used policy for selecting (i) and (ii) has a direct influence on communication delay and, hence, the analysis in this section depends on the used policy. As explained before, s^2TDMA uses a uniform distribution of nodes to slots. Note that other policies are also possible and can be easily derived in a similar manner, for example, where nodes may be unevenly distributed to slots.

5.1 Probability of Collision

As stated above, we consider a receiver-initiated WSN, i.e., where the receiver wakes up a given number of (sender) nodes and allows them to send. Let us denote by N the set of these nodes. Further, p_i denotes the probability that node i in N sends after waking up. We assume that this follows a random process and, hence, p_i can obtained statistically by observing the behavior of a large set of similar nodes over a large period of time.

Now, if node i is stochastically independent of all other nodes in N, the probability that it suffers no collision is given by the following expression:

$$\bar{p}_i = p_i \prod_{j \in N, j \neq i} (1 - p_j), \tag{1}$$

where $1 - p_j$ is the probability that a node j in N does not send, i.e., (1) gives the probability that node i sends alone. As a result, node i undergoes a collision with a probability of $1 - \bar{p}_i$, i.e., at least one of the other nodes in N sends simultaneously.

In case of a collision, the receiver starts an arbitration cycle with a given number of slots $\hat{s}$. We consider the case of a uniform distribution of nodes to slots, where nodes in N are sorted in order of decreasing priority, i.e., node i has a higher priority than node j if $i < j$ holds. Node 1 to node $\left\lfloor \frac{|N|}{\hat{s}} \right\rfloor$ are allocated to the first slot, where $|N|$ is the number of nodes in N. Next, node $\left\lfloor \frac{|N|}{\hat{s}} \right\rfloor + 1$ to node $2 \cdot \left\lfloor \frac{|N|}{\hat{s}} \right\rfloor$ are allocated to the second slot and so on. Finally, node $(\hat{s} - 1) \cdot \left\lfloor \frac{|N|}{\hat{s}} \right\rfloor + 1$ to node $|N|$ are assigned to slot $\hat{s}$.

If a collision occurs in one of these slots, the receiver starts a second arbitration cycle, where the subset of nodes in the corresponding slot are now split into $\hat{s}$ slots — we consider that the same number of slots is used at every such cycle; however, s^{2}TDMA can also be configured for a varying number of slots. A node i may undergo a certain number of arbitration cycles until it sends without collisions. Note that, in the worst case, this may not happen until node i has exclusive use of a slot.

Now, in a given arbitration cycle ℓ, a node i's probability of sending without collisions is given by the following expression:

$$\bar{p}_{i\ell} = p_i \prod_{j \in N_{\ell s}, j \neq i} (1 - p_j), \tag{2}$$

where $N_{\ell s}$ is the subset of nodes assigned together with node i to the same slot s being $1 \leq s \leq \hat{s}$ and $1 \leq \ell \leq \hat{\ell}$. Here, $\hat{\ell}$ is the maximum number of arbitration cycles for given N and $\hat{s}$. Similar to before, $1 - \bar{p}_{i\ell}$ is the probability that node i suffers a collision in the arbitration cycle ℓ.

The maximum number of arbitration cycles $\hat{\ell}$ can be obtained considering that nodes are divided into $\hat{s}$ slots every time there is a collision. In the first arbitration cycle, this is $\left\lfloor \frac{|N|}{\hat{s}} \right\rfloor \leq \frac{|N|}{\hat{s}}$. In the second arbitration cycle, this is $\left\lfloor \frac{\left\lfloor \frac{|N|}{\hat{s}} \right\rfloor}{\hat{s}} \right\rfloor \leq \frac{|N|}{\hat{s}^2}$, and so on. Since no collision can happen and, hence, no further arbitration cycle will be started when there is only one node per slot, we have that $\frac{|N|}{\hat{s}^\ell} = 1$ must hold. We can apply logarithm to solve for ℓ and round up to obtain an upper bound on the number of arbitration cycles as shown below:

$$\hat{\ell} = \left\lceil \frac{\ln |N|}{\ln \hat{s}} \right\rceil. \tag{3}$$

5.2 Communication Delay

A node i's shortest communication delay is given by $l_i + l_{ack} + 2t_d$, i.e., the time needed by node i to send its packet plus the time needed by the receiver to send an acknowledgment plus the time needed by the transceiver IC to process both of them — see again Fig. 3. This happens when node i needs to send and no other node in N has anything to send, which again has a probability as per (1).

On the other hand, a node i's longest communication delay — denoted by $\hat{c}_i$ — happens when it needs to send data together with all nodes in N that (i) either have higher priority or (ii) are allocated to the same slot. This is because higher-priority nodes occupy the first slots in any arbitration cycle and lower-priority nodes send their data on node i's slot producing additional collisions. Considering again that nodes in N are sorted in order of decreasing priority, the upper bound of $\hat{c}_i$ can be computed in the following manner for $1 \leq i \leq |N|$:

$$\hat{c}_i \leq \sum_{j=1}^{i} L_{ack,j} + \sum_{j=1}^{\lceil \frac{i}{2} \rceil} \left\lceil \frac{\ln(|N| - 2j)}{\ln \hat{s}} \right\rceil \times L_{nak}, \tag{4}$$

where $L_{ack,j} = l_j + l_{ack} + 2t_d$ and $L_{nak} = l_{max} + l_{nak} + 2t_d$.

The first term in (4) is the sum of the transmission times of packets of higher-priority nodes with their respective acknowledgments. The second term requires more explanation. As we already know, a collision is resolved when no more collisions occur in the corresponding (sub-)cycle. Since that sub-cycle can contain up to $\hat{s}$ slots, this means that up to $\hat{s}$ nodes can finish their transmission. However, the number of slots may vary in each arbitration cycle, for example, if the remaining IDs are less then $\hat{s}$, which typically happens after several ID splits. For the sake of simplification, let us pessimistically assume that every such (sub-)cycle has 2 slots. This means that only two IDs are resolved each time. This way, considering node i in N, up to $\lceil \frac{i}{2} \rceil$ sub-cycles will be generated. As previously discussed, each of these require at most $\hat{\ell}$ NAK messages — see again (3) with $\hat{s} = 2$. However, after every successful sub-cycle j, the number of remaining nodes reduces to $|N| - 2j$. Since NAK messages follow collided packets, their total duration is $(l_{max} + l_{nak} + 2t_d)$, where l_{max} is the longest among all collided packets.

Table 1: CC2420 radio and simulation parameters

Parameter	Value
Bit rate	250 $kbps$
CCA Sampling time	128 μs
RX/TX Switching time	192 μs
Reception response delay t_d	420 μs
CSMA slot length	320 μs

In other words, whereas the first term in (4) results from the transmission times by higher-priority nodes, the second term accounts for protocol/arbitration overhead in the worst case.

6. SIMULATION RESULTS

In this section, we present the results of a simulation based on the OMNeT++ network simulation framework [12] and an extension for mobile and wireless networks named MiXiM [6]. This allows us to effectively simulate our network with different physical parameters and to record statistical values for very large numbers of transmissions.

The simulated network consists of one receiver and a selectable number of n transmitters that can either be within range of each other and, hence, interfere with each other, or simulate hidden terminals. The receiver node is a simple data sink, whereas transmitter nodes are data sources that transmit packets with a certain pattern according to the compared MACs as explained below. Note that our proposed MAC also supports multihop communication. However, for simplicity, a single hop and single sink setting is used, which also matches the requirements of Strawman [7] and Bin-MAC [10].

Our simulation is based on a receiver-initiated protocol, similar to RI-MAC [11], in which the sink periodically broadcasts probe messages to wake up transmitting nodes and trigger data communication. The number of nodes to wake up can be specified, whereas the nodes themselves are picked randomly to ensure that different possible combinations of IDs are considered. All simulation data is recorded and processed by the framework at runtime. In particular, the time stamps of the different packets sent are compared to determine whether packets overlap and, hence, get lost. Each simulation was performed with different parameters, for which at least 1,000 probe-cycles have been simulated each time.

We consider the following four MAC protocols and compare them in the simulation:

- The s^2TDMA scheme is our transmission scheme as presented in Section 4.
- The *BTCR* scheme uses the deterministic binary tree contention resolution from BIN-MAC [10].
- The *Strawman* scheme is a contention-based MAC of random nature as presented in [7].
- The *CSMA* scheme is based on the non-persistent Carrier Sense Multiple Access (CSMA) method as defined in IEEE 802.15.4.

The transmission rate was fixed to 250 $kbps$ and the radio parameters were taken from the widely used transceiver CC2420, as displayed in Table 1. Similar to the IEEE 802.15.4 standard, each data frame consists of 4 $bytes$ preamble, 1 $byte$ SFD (start frame delimiter), 1 $byte$ length code, 1 $byte$ CRC (cyclic redundancy check) and a number bytes for data payload, resulting in 7 $bytes$ overhead. The control messages, such as ACK, NAK, Strawman notification, etc., can omit the length field, since they are of constant size and, hence, have just 6 $bytes$ overhead.

The *Strawman* and *CSMA* control messages both contain a 1 $byte$ type field, defining the type of control message, and a 2 $bytes$ ACK address field. In addition, *BTCR* and s^2TDMA require further 4 $bytes$ for address fields to specify the upper and lower ID ranges. For simplicity, we again assume that all control messages have the same length for a specific MAC, for example, *Strawman* notification messages have the same length as *Strawman* ACKs and NAKs.

Unless otherwise noted, the remaining parameters are set as follows: The simulated system consists of 50 nodes, of which $2 \leq N \leq 50$ nodes contend each cycle. Data packets have a fixed length of 8 $bytes$, hence, the payload is 1 $byte$. In case of s^2TDMA, we set the number of TDMA slots per arbitration cycle to $\hat{s} = 4$. For *Strawman*, we use the parameters as specified in [7] and [8], resulting in a maximum contention length of 3.7 ms and a maximum retransmission number of 2. For more details about *BTCR* and *Strawman*, see Section 2.

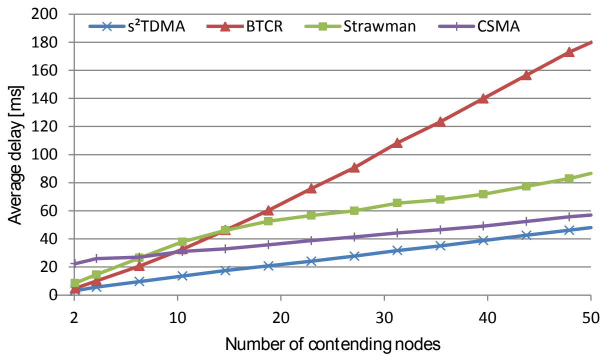

Figure 4: Average transmission delay for successful data transmission

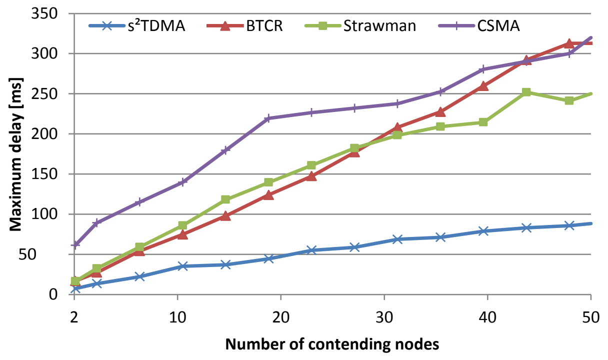

Figure 5: Maximum recorded transmission delay for successful data transmission

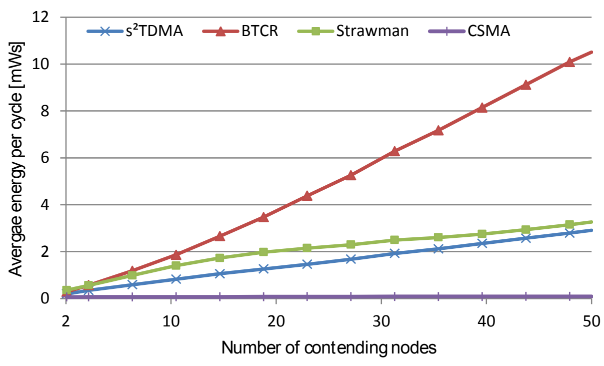

Figure 6: Average energy consumption for successful data transmission

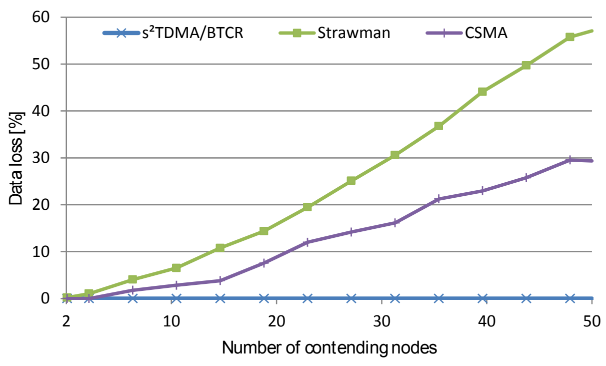

Figure 7: Amount of lost (failed) data transmissions

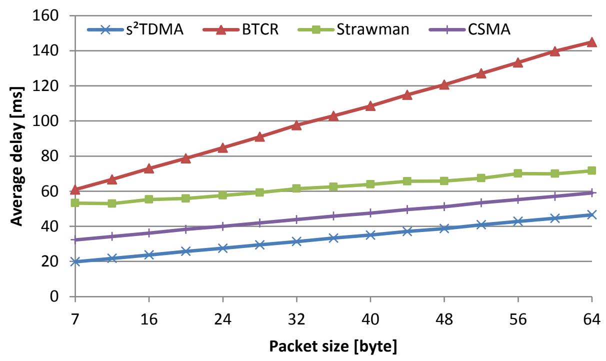

Figure 8: Average delay for a varying packet size

6.1 Delay

Let us first analyze the average transmission delay, i.e., the time from waking up the node until successful reception of its data, as depicted in Fig. 4.

Clearly, the delay rises for a greater N for all nodes, since more contenting nodes imply more collisions and, hence, a longer resolution time. In case of $s^2 TDMA$ and $BTCR$, the delays rise linearly for higher N, whereby the delay of the $BTCR$ scheme is generally higher, as its binary tree scheme needs more retransmissions for collision resolution. Similarly, $CSMA$ also rises linearly with rising N, however, it rises at a slower rate. This is because $CSMA$ starts to loose data, as we discuss later in more detail. Nodes, therefore, start to drop out of the contention resulting in (little) less load for the remaining contenders. However, for low $N < 8$, $CSMA$ has higher delays than the other schemes, as its back-off mechanism produces relatively long delays for low N.

In contrast, $Strawman$ has a relatively high delay compared to $s^2 TDMA$ and $CSMA$. This can be explained by the fact that is uses pulses instead of full packets for contention resolution, which is a much slower operation mode for most transceiver ICs (including the CC2420). For example, one CCA request with $128\,\mu s$ takes as much time as transmitting $4\,bytes$ of data. Consequently, every $Strawman$ contention pulse, which we previously set to a maximum length of $3.7\,ms$, can have a length equal to up to $115\,bytes$. This effect, however, mitigates when considering larger packet sizes, as discussed later. Similar to $CSMA$, the delay also rises at a lower rate for higher N, as $Strawman$ looses packets.

Fig. 5 shows the maximum delay that was recorded during simulation of Fig. 4. As we can see, both probabilistic approaches have a relatively high maximum delay compared to $BTCR$ and $s^2 TDMA$. This is because these are designed for simplicity and good average performance, but not for guaranteeing any QoS. During high network load these will therefore produce high loss rates and high delays. In contrast, $BTCR$ now has a relatively low maximum delay, however, $s^2 TDMA$ results again in the lowest delay of all four schemes.

6.2 Energy Efficiency

Lets us now analyze the average energy consumption of the different MAC protocols, as displayed in Fig. 6. Analogous to the delay from Fig. 4, the energy consumption curves of $BTCR$, $Strawman$ and $s^2 TDMA$ share a similar shape. This is because delay and energy consumption are directly connected to each other, since these schemes are active during the whole contention resolution and do not implement sleeping times like $CSMA$. This means that higher contention levels require the nodes to be active longer, resulting in a higher delay and energy need. On the other hand, $CSMA$ requires very little energy, since retransmission numbers are much smaller and nodes sleep during back-off times.

6.3 Reliability

Fig. 7 shows the reliability, i.e., the percentage of lost packets, for different levels of contention. As expected, both deterministic schemes result in 0 packet loss, as these are specifically designed for reliable data transfer. On the other hand, both random protocols incur in data loss, which strongly increases for higher N. Since probabilistic approaches are generally uncoordinated in channel access, a higher number of contending nodes makes collisions more likely. As a consequence, the maximum retransmission number as well as the limited back-off retries of $CSMA$ or contention packet sizes of $Strawman$, start to not be sufficient anymore and data is lost. This makes both $CSMA$ and $Strawman$ not applicable for networks, where possibly high numbers of nodes can contend simultaneously.

The reliability of $CSMA$ and $Strawman$ can be improved by increasing contention packet sizes, back-off retries or retransmission numbers. However, in return, both delay and energy consumption will increase.

6.4 Packet Size vs Performance

Next, we discuss the effects of varying the data size with respect to delay, energy consumption and data loss. To this end, we simulated a network of $N = 20$ contending nodes per probe-cycle and vary the data size from $7\,bytes$ (empty packet) to $64\,bytes$. The results regarding the average delay are depicted in Fig. 8. Since the energy consumption again shows similar behavior, we forgo to examine it separately.

Clearly, increasing the data size increases the delay and energy consumption of all four MAC protocols. For *BTCR*, this effect is the most dominant, since its contention resolution is triggered upon collisions of data packets. Increasing the packet size therefore linearly increases the delay. Similar, our *s^2 TDMA* scheme is also based on the same principle. However, its generally faster convergence results in less collisions, hence, bigger data sizes still increase the delay, but not a strong as for *BTCR*. The delay of *Strawman* and *CSMA* also rises linearly for increasing data sizes, but *Strawman* has a higher starting (offset) value due to its arbitration process.

In contrast, changing the packet sizes does not affect the loss rate of the *BTCR* and *s^2 TDMA* and it remains at $0\,\%$. Also for *Strawman*, no change can be observed, since its arbitration process is independent of the packet size. For *CSMA*, however, increasing the packet size slightly increases the loss rate. This is because a longer transmission time will increase the probability that nodes, which randomly want to access the channel, sense the channel as busy and perform a back-off.

7. CONCLUSION

In this paper, we proposed a MAC providing a deterministic contention resolution mechanism in form of generating spontaneous TDMA cycles upon collisions. This results in an interval tree search featuring a fast conflict resolution with $log_{\hat{s}}$-complexity, being $\hat{s}$ the number of slots in each such cycle.

The proposed scheme, called s^{2}TDMA, is well suited for receiver-initiated networks with high numbers of contenders, which are expected in application in the area of in Internet of Things, Cyber-Physical Systems, etc. Since s^2TDMA is only triggered on collisions and does not rely on periodic synchronization, it offers both low overhead and good energy efficiency under low contention as well as fast collision resolution under high contention.

In addition to analyzing the worst-case behavior, we performed extensive simulations using OMNeT++. The proposed technique never leads to packet losses (within the network) and shows good scalability and quick adaption to fast changing traffic load. Further, our experiments suggest that the proposed s^{2}TDMA is energy-efficient and significantly outperforms other approaches from the literature.

8. REFERENCES

[1] N. Abramson. The Aloha System: Another Alternative for Computer Communications. In *Proceedings of the Fall Joint Computer Conference*, 1970.

[2] J. Afonso, L. Rocha, H. Silva, and J. Correia. MAC Protocol for Low-Power Real-Time Wireless Sensing and Actuation. In *Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2006.

[3] D. Carlson and A. Terzis. Flip-MAC: A Density-Adaptive Contention-Reduction Protocol for Efficient Any-to-One Communication. In *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, 2011.

[4] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2010.

[5] X. Ji, Y. He, J. Wang, W. Dong, X. Wu, and Y. Liu. Walking down the STAIRS: Efficient Collision Resolution for Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2014.

[6] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. K. Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating Wireless and Mobile Networks in OMNeT++: The MiXiM Vision. In *Proceedings of the International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools)*, 2008.

[7] F. Österlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels. Strawman: Resolving Collisions in Bursty Low-Power Wireless Networks. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks (IPSN)*, 2012.

[8] F. Österlind, N. Wirström, N. Tsiftes, N. Finne, T. Voigt, and A. Dunkels. StrawMAN: Making Sudden Traffic Surges Graceful in Low-power Wireless Networks. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensors (HotEmNets)*, 2010.

[9] I. Rhee, A. Warrier, M. Aia, J. Min, and M. Sichitiu. Z-MAC: A Hybrid MAC for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 16:511–524, 2008.

[10] V. Salmani and P. H. Chou. Bin-MAC: A Hybrid MAC for Ultra-Compact Wireless Sensor Nodes. In *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2012.

[11] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the ACM conference on Embedded Network Sensor Systems (SenSys)*, 2008.

[12] A. Varga. The OMNeT++ Discrete Event Simulation System. In *Proceedings of the European Simulation Multiconference (ESM)*, 2001.