BitMAC: A Deterministic, Collision-Free, and Robust MAC Protocol for Sensor Networks

Matthias Ringwald, Kay Römer
Dept. of Computer Science
ETH Zurich, Switzerland
{mringwal,roemer}@inf.ethz.ch

***Abstract*— Collisions are a source of inefficiency in contention-based MAC protocols that should be reduced to a minimum. We show that concurrent multiple access to a communication channel will, however, not necessarily lead to a collision with undesirable effects. Rather, we demonstrate that it is possible for a receiver to hear the bitwise "or" of the transmissions of multiple synchronized senders within communication range. This unconventional communication model allows the efficient implementation of a number of basic operations that serve as a foundation for BitMAC: a deterministic, collision-free, and robust MAC protocol that is tailored to dense sensor networks, where nodes report sensory data over multiple hops to a sink.**

I. Introduction

Many widely used Medium Access Control (MAC) protocols for sensor networks are based on contention, where concurrent access to the communication channel by multiple sensor nodes leads to so-called collisions. Collisions are a source of many undesirable properties of these MAC protocols. For example, collisions lead to a reduced effective channel bandwidth and to increased energy consumption. Often, probabilistic methods are used to resolve collisions, which typically leads to increased, unpredictable delays. In particular, collisions introduce an indeterminism, which makes this type of MAC protocol ill-suited for systems with real-time constraints. Hence, collisions are commonly considered a "bad thing", and MAC protocols strive to avoid them wherever possible.

In sensor networks, communication activity is typically triggered by an event in the physical world. In dense networks, such an event triggers communication activity at many collocated nodes almost concurrently, such that a high probability of collisions must be expected. This has been verified by quantitative studies in [6], [26].

In this paper, we adopt another, more positive view on collisions. In particular, we show that a set of synchronized nodes can concurrently transmit data, such that a receiver within communication range of these nodes receives the bitwise "or" of these transmissions. Using existing sensor node hardware, we show that such a communication model can indeed be implemented with commonly used low-power radios.

Based on this communication model, we can provide efficient parallel implementations of a number of basic operations that form the foundation for BitMAC, which has the following key features:

- *No collisions*: Although nodes access the channel concurrently, this does not lead to the bad effects of collisions mentioned above.
- *Determinism*: There are deterministic bounds on the execution time of all protocol elements.
- *Robustness*: A large class of temporary and permanent node failures can be tolerated in dense networks.
- *Efficiency*: The protocol overhead of BitMAC is low and the channel bandwidth can be utilized efficiently.
- *Self-containedness*: BitMAC does not assume any out-of-band mechanisms. Time synchronization is an integral part of the protocol.
- *Dense, many-to-one networks*: BitMAC is particularly tailored to dense sensor networks, where sensor nodes report sensor readings to a sink node over multiple hops.

The remainder of this paper is structured as follows. We discuss assumptions about physical layer communication and about applications in Section II. An overview of BitMAC will be given in Section III, before presenting the protocol in detail in Sections IV to VI. We evaluate BitMAC in Section VII, discuss remaining issues in Section VIII and related work in Section IX, and conclude the paper in Section X.

II. Basic Assumptions

In this section, we present our communication model and characterize the class of applications, for which BitMAC has been designed.

A. Communication Model

Our work is based on the assumption, that a node receives the "or" of the transmissions of all senders within communication range. In particular, if bit transmissions are synchronized (e.g., slotted access to the medium) among a set of senders, a receiver will see the bitwise "or" of these transmissions.

This behavior can actually be found in practice for radios that use On-Off-Keying (OOK), where "1"/"0" bits are transmitted by turning the radio transmitter on/off. Note that transmitting a zero is then equal to sending nothing.

The two currently most used radios for sensor networks, the RFM TR 1000 series (e.g., MICA and RENE motes [27], Scatterweb ESB [32], TecO Particles [33], and the EYES project prototype [30]), and the Chipcon CC1000 series (e.g., MICA2 motes, MANTIS Nymph [31], BTnode3 [28]) support this mode of operation. Although the Chipcon uses Frequency-Shift-Keying (FSK) by default, a Chipcon application note [29] describes how to emulate OOK. The newer Chipcon CC1020 and CC1021 directly support OOK modulation. We have verified the above communication model using the Chipcon CC1000 on BTnode3 as discussed in Sections VII-A and VII-B.

BitMAC will use this communication model (and hence OOK) only for a limited number of control operations. For the remainder (e.g., payload data transmission), other, perhaps more efficient modulation schemes can be used if supported by the radio. In this case it might be necessary to adjust the transmit power such that the communication ranges are similar for both modulation schemes.

Furthermore, our work is based on the assumption that the radio supports a sufficient number of communication channels, such that nodes within communication range can switch to different channels to avoid interference. The Chipcon CC1000, for example, can support up to 130 channels in the 915 MHz ISM band, or 35 channels in the 868 MHz ISM band (assuming a channel width of 200 kHz). The number of available channels determines the maximum network density that can be supported by BitMAC (see Section VII-C).

We will also exploit the fact that typical radios cannot send *and* receive at the same time. Rather, they support a send mode and a receive mode. Switching between these modes requires a certain amount of time. That is, if two nodes within communication range either both send or both receive, they do not interfere with each other.

B. Application Characteristics

BitMAC is designed for data-collection sensor networks, where many densely deployed sensor nodes report sensory data to a sink across multiple hops. In order to avoid the bottleneck of the sink, data from multiple sensor nodes may be aggregated by nodes in the network. Data communication is mostly uplink from the sensor nodes to the sink, although the sink may issue control messages to the sensor nodes. One prominent example of this application class are directed diffusion [9] and TinyDB [16]. Many concrete applications (e.g., [2], [11], [19], [20]) show this behavior as well.

Furthermore, it is assumed that the network topology is mostly static. That is, after initial deployment, node mobility and addition are rare events. However, BitMAC is designed to support a large class of permanent or temporary node failures efficiently and without introducing contention or indeterminism. Hence, BitMAC can support applications with real-time and robustness requirements. Applications such as [2], [11], [20] fall into this class.

III. Protocol Overview

BitMAC is based on a spanning tree of the sensor network with the sink at the root. In this tree, every internal node and its direct children form a star network. That is, the tree consists of a number of interconnected stars. Within each star, time-division multiplexing is used to avoid interference between the children sending to the parent. Time slots are allocated on demand to nodes that actually need to send. Using a distributed graph-coloring algorithm, neighboring stars are assigned different channels as to avoid interference between them. Both the setup phase and actual data transmission are deterministic and free of collisions.

In the following sections we will describe the complete protocol with increasing level of complexity. We will begin with basic techniques for communication among a set of child nodes and a parent node in a star network. We will then discuss the part of the MAC protocol used to control a single star. Finally, we will describe how these stars can be assembled to yield the complete multi-hop MAC protocol.

IV. Basic Techniques

In this section, we describe basic techniques for communication of a set of children with a parent in a star network as depicted in Figure 1 (a). While the following discussion is tailored to MAC protocols, the techniques might also be applicable in other contexts.

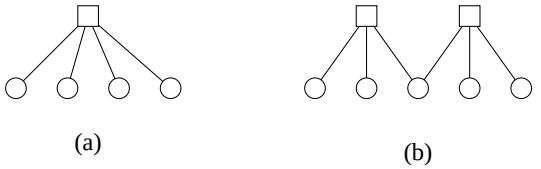

Fig. 1. (a) Star network with a single parent. (b) Multiple stars with shared children.

A. Single-Bit Transmissions and Preamble Elimination

Every transmission of payload data typically has to be preceded by a preamble (often a "101010..." bit sequence) and a start-of-packet (SOP) delimiter with a total size of about 100 bit. Preamble and SOP are needed to synchronize the receiver to the sender and to adjust the bit-decision threshold.

Control traffic of a MAC protocol often consists of data packets with few or even single bits. For example, children have to signal a send request and the parent has to send acknowledgments to children. Preceding every such transmission with a preamble represents an significant overhead of MAC protocols.

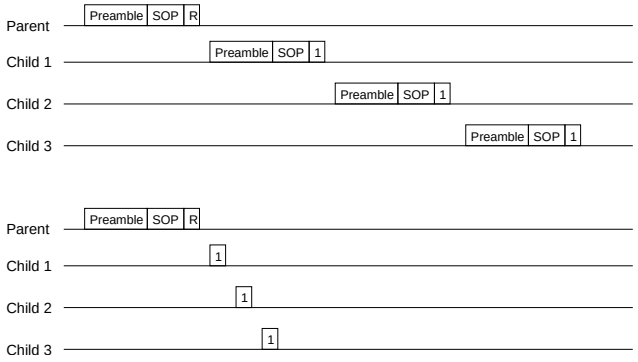

Fig. 2. Single-bit transmissions to a parent node: with (top) and without (bottom) preambles. Time increases to the right.

Fortunately, many preamble transmissions can be eliminated. For uplink communication, the parent triggers (by means of a request message "R" in Figure 2 (top)) the children to send single bits of data (e.g., send requests) within a short time frame. Typically, these single bit transmissions are preceded by preambles as in Figure 2 (top).

However, since the parent's preamble does already synchronize all children, the latter can maintain synchronization for the duration of several hundred bits using their hardware clocks (see Section VII-B). Hence, single bits can be transmitted without preambles as depicted in Figure 2.

For downlink communication, where the parent has to send individual bits to many children (e.g., acknowledgments), many individual bit transmissions with individual preambles can often be concatenated to a single packet with a single preamble at the cost of an increased delay. We will see an example in Section V.

B. Integer Operations

Let us assume that all or a subset of children need to transmit k-bit unsigned integer values to the parent, where the latter is interested in various aggregation operations (OR, AND, MIN, MAX) on the set of values of the children. Below we discuss efficient ways to implement these operations, assuming synchronized nodes and the communication model described in Section II-A.

Obviously, a bitwise "or" can be implemented by having the children synchronously transmit their values bit by bit. Our communication model ensures that the parent will receive the bitwise "or" in time $O(k)$. Since $x \text{ AND } y = \overline{\bar{x} \text{ OR } \bar{y}}$ (where $\bar{x}$ is the bitwise inversion of x), the bitwise "and" can be obtained if the children invert their values before transmission and if the parent inverts the received value.

By interpreting an integer value as a set (where i is contained in the set if and only if the i-th bit of the value is 1), the operations OR and AND implement the UNION and INTERSECTION of integer sets, respectively. Integer values with k bits can then support sets of up to k elements. Often, long sequences of zero bits have to be transmitted where values represent integer sets with only a few elements. However, as mentioned in Section II-A, transmitting sequences of zeros equals doing nothing, allowing for an energy-efficient implementation by switching off the radio transmitter.

In order to compute MAX, k communication rounds are performed. In the i-th round, all children send the i-th bit of their value (where $i = 0$ refers to the most significant bit), such that the parent receives the bitwise "or". The parent maintains a variable $maxval$ which is initialized to zero. When the parent receives a one, it sets the i-th bit of $maxval$ to one. The parent then sends back the received bit to the children. Children stop participation in the algorithm if the received bit does not equal the i-th bit of their value as this implies that a higher value of another child exists. After k rounds or time $O(k)$, $maxval$ will hold the maximum among the values of the children. Note that children who sent the maximum will implicitly know, since they did not stop participation. Likewise, children who did not send the maximum can also detect this. Additionally, all children can find out the maximum by listening to *all* messages sent by the parent. The MIN operation can be implemented if the children invert their values before the procedure and if the parent inverts $maxval$ after the procedure.

C. Vectorial and Parallel Integer Operations

In the previous section we discussed, how a single instance of an integer operation can be performed. In this section we discuss the efficient execution of multiple instances of the same operation. We distinguish two different problems: *vectorial* and *parallel* integer operations.

For a vectorial operation, each child has a vector of n integer values, such that the parent would have to perform an operation n times, where the i-th execution considers the values at position i in the vectors of the children (cf. single instruction multiple data). However, these n sequential operations can be combined into a single operation as described in the previous section, where the messages contain the respective information for all elements of the vector. For k-bit integers, this requires time $O(nk)$. A vectorial operation is more efficient that the independent execution of many operations, since many preambles and radio switches (transmit to receive and vice versa) can be omitted.

For a parallel operation, the parents of multiple stars that share one or more children have to perform the same integer operation as depicted in Figure 1 (b). If all involved nodes are synchronized, all of the above integer operations can be performed (synchronously) in parallel. For the OR and AND operations, our communication model will ensure that all parents will obtain the correct result for their respective children. For MIN and MAX, the parent will in general not obtain the correct result. However, each child n will find out whether it presented the MIN/MAX value among the children who share a parent with n. As explained in Section IV-B, a node presented the MIN/MAX if it did not stop participation in the algorithm. We will use this operation in Section VI-A. Note that any number of such parallel operations on k-bit values can be performed in time $O(k)$. Vectorial and parallel operations can also be combined, requiring time $O(nk)$.

D. Bit Errors

The techniques described in the previous sections are sensitive to errors, where one or more bits are not correctly delivered to the receiver. For the discussion of error handling we distinguish two cases. In the first case, one or more nodes send the same bit value concurrently. In this case, traditional error handling mechanisms such as checksums or coding techniques can be used.

In the second case, two or more transmitters send different bits concurrently. According to our communication model, the receiver will see the "or" of these bits. In this case, checksums and coding techniques cannot be applied, since the bitwise "or" of checksums or encoded values is generally not equal to the checksum or encoded value of the bitwise "or" of the original bits. For example, if $b_1 \neq b_2$ are the bits to be transmitted by two nodes and $e(b_i)$ are the encoded bits, then $e(b_1 \,\mathrm{OR}\, b_2) \neq e(b_1) \,\mathrm{OR}\, e(b_2)$ must be expected.

To detect errors in this case, the bits can be sent unencoded two or more times. If different values are received, then an error is assumed. If three or more transmissions are performed, then errors can be corrected if the majority of the transmissions are identical. Bluetooth uses such a forward error recovery to protect its protocol headers.

Sometimes it is possible to handle transmission errors more efficiently at the application level. Errors can be detected, for example, if constraints (e.g., minimum/maximum/exact number of "1" bits in a received bit vector) are known on the transmitted values. For example, if a node signals a send request, then it should be assigned a time slot for data transmission (see Section V). Due to bit errors, a node may find that it hasn't been assigned a slot and can retry transmission in a later round. Likewise, if a node didn't signal a send request, then bit errors may lead to the false assignment of a time slot. Although this can result in reduced efficiency, it is not strictly necessary to correct this error.

In the remainder of the paper we will assume error-free transmissions. As part of future work, we will incorporate the above techniques into our protocol.

V. Star Network

Using the basic techniques presented in the previous section, we present a MAC protocol for star networks, which will be used as a building block for the multi-hop protocol presented in the subsequent section.

The protocol supports uplink communication from the children to a parent, and downlink communication from the parent to one or more children. Time-division multiplexing is used to avoid collisions and to ensure a deterministic behavior of the protocol. In order to optimize bandwidth utilization, time slots are allocated on demand to nodes that actually need to send data.

Let us assume for the discussion that children have been assigned small, unique integer IDs in the range $1...N$. We will show in Section VI how these can be assigned.

The protocol proceeds in rounds with the parent acting as a coordinator. A round starts with the parent broadcasting a beacon message to the children. The beacon contains an indicator whether this round is downlink or uplink communication. If the round is downlink, the

beacon message will be followed by the payload data, which will usually contain the address (e.g., ID) of the target node(s). If this is an uplink round, children will transmit send requests to the parent. After receiving these requests, the parent constructs a schedule and broadcasts it to the children. From this schedule, children can deduce their time slot for transmission. During their time slots, children send their payload data to the parent. The parent will then acknowledge successful receipt. If transmission failed, the affected children will try a retransmission in the next round.

The beacon serves multiple purposes: it indicates the begin of a new round, carries control information, and synchronizes the children (by means of a preamble). For the send requests and the acknowledgments, an integer set of node IDs (i.e., a bit vector of length N) is used to reduce preamble transmissions as explained in Section IV-A. A node with ID i is part of such a set if and only if the bit i is "1" in the bit vector.

For the send requests, each child with pending data sends such an integer set containing only its ID without a preamble. The parent will then receive the UNION of these sets as depicted in the lower part of Figure 2. The parent then sends back the received integer set to the children. Having received it, the children also know the IDs of the nodes with pending send requests. Assuming that nodes with smaller IDs send first, both parent and children can assign time slots for the transmissions. After receiving all the data transmissions, the parent sends an integer set to the children that contains the IDs of the children with successful transmissions.

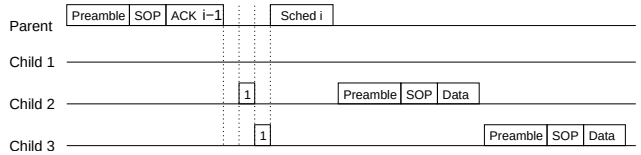

Fig. 3. Round i of the optimized MAC protocol for star networks.

As a further optimization, the acknowledgment set can be concatenated with the beacon of the following message as depicted in Figure 3. Also, the schedule can be sent without an additional preamble, since the preamble that is part of the beacon is sufficient to synchronize the children for the duration of several hundreds of bits (see Section VII-B). Hence, preambles are only needed for the beacon and for the payload data packets.

VI. Multi-Hop Network

We now show how to extend the protocol for star networks to a multi-hop network. This will be based on a spanning tree of the network with the sink at the root, where each internal node and its direct children form a star that uses the protocol presented in the previous section. As to avoid interference, stars are assigned different communication channels.

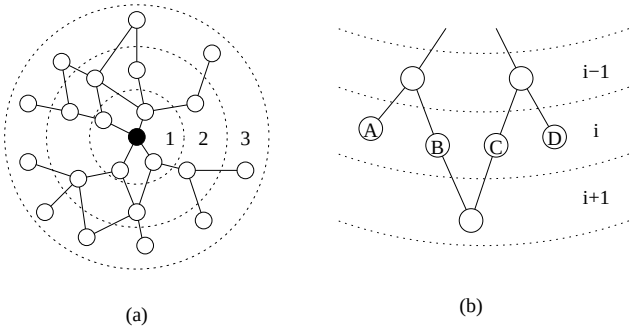

Fig. 4. Distance from the sink (•) imposes a ring structure on the network. Network links between nodes in the same ring are not shown.

For our discussion let us assume that nodes possess unique MAC addresses (e.g., 16 bit identifiers) and that each node knows its hop-distance from the sink. We show in Section VI-B how this can be achieved. Nodes in the network can now be partitioned based on their distance from the sink, such that nodes with equal distance form a ring around the sink. We will denote the ring of nodes with distance i as the ring i. A node in ring i now has one or more parents in ring $i-1$, and zero or more children in ring $i+1$. Note that nodes in ring $i+1$ by definition are *not* within communication range of nodes in ring $i-1$. Nodes in the same ring will not interfere with each other, since they will either all send or all receive at the same time.

A. Assigning Channels and IDs

In order to turn this hierarchical ring structure into a tree, each node must be assigned to a single parent. A parent in ring i then must share a channel with its children in ring $i+1$, such that no other parent in ring i of the children uses the same channel. More formally, this requires the assignment of small integers (i.e., channel identifiers) to nodes in ring i, such that nodes who share a child in ring $i+1$ are assigned different numbers.

We assumed in Section V that children of a single parent are assigned small unique integer numbers. With respect to the ring structure, this task can be formulated as assigning small integers to nodes in ring i, such that nodes who share a parent in ring $i-1$ are assigned different numbers.

The above two problems can be combined into a two-hop graph coloring problem: assign small integers (i.e., *colors*) to nodes in ring i, such that nodes with the same number do not share a common neighbor in ring $i-1$

or $i+1$. Note that common neighbors in ring i are not considered. In Figure 4 (b), a valid color assignment would be $A = D = 1, B = 2, C = 3$.

In order to solve this problem, let us assume for now that all nodes in rings $i-1, i, i+1$ are synchronized, such that the vectorial and parallel integer operations described in Sections IV-B and IV-C can be applied.

Let us further assume that a small number C is known, such that the numbers $1...C$ (the *color space*) are sufficient to solve the coloring problem. We will discuss C in Section VII-C. In practice, C will be equal to the number of available radio channels.

Under these assumptions, we can use a deterministic variant of the algorithm presented in [10] to solve the above described two-hop graph coloring problem for ring i. For this algorithm, each node in ring i maintains a set P (the *palette*) of available colors, which initially contains $1...C$. Initially, all nodes use the same communication channel.

The algorithm proceeds in rounds. In each round, every node in ring i selects an arbitrary (e.g., smallest, random) color c from its palette P. Some nodes may have selected conflicting colors. For each possible color, the algorithm will allow the nodes with the largest respective MAC address to keep its color. All other nodes reject the color. For this, each node sets up a vector $v[1...C]$ of MAC addresses with one entry for each possible color. $v[c]$ is set to the MAC address of the node, all other entries are zero.

Now, all nodes in rings $i-1$ and $i+1$ synchronously perform a vectorial parallel MAX operation (see Section IV-C) over the vectors v of the nodes in ring i. As a side effect of this operation, all nodes in ring i will then know whether or not they are the node with the maximum MAC address for the selected color within two hops. If a node is the maximum, then it keeps color c and is finished. The node will not participate in consecutive rounds. If a node detects that it is not the maximum, it removes c from P and proceeds with the next round. By listening to the values received from the parents as part of the MAX operation, the node can also find out which colors have been selected by other nodes. These are also removed from P.

Since the above algorithm assigns at least one color per round, it requires at most C rounds to finish.

A node in ring i with color c will use the channel associated with c for communication with its children in ring $i+1$. c will also be used by nodes in ring i as the small integer ID for communication with its parent in ring $i-1$ (see Section V).

In a final step, all nodes in ring i synchronously broadcast an integer set (i.e., bit vector) that contains the selected color c. Nodes in ring $i+1$ will receive the UNION (see Section IV-B) of the colors of all their parents. By picking one of the colors and switching to the according communication channel, every node in ring $i+1$ can select a single parent. Note that this selection can be changed at any time, for example, if the parent fails.

Note that the above procedure can be performed for multiple rings in parallel, because rings $i-1, i, i+1$ do not interfere with rings $i+3, i+4, i+5$. Due to this, rings $i, i+4, i+8, i+12, ...$ can be colored in parallel. Hence, four parallel coloring steps are sufficient to color an arbitrary network with an arbitrary number of rings. Since each step requires at most C rounds of the coloring algorithm, an arbitrary network can be colored in $4C$ parallel rounds. Moreover, as each node knows its ring number, it can autonomously decide when to start the coloring algorithm, since it knows how many rounds are required to color the other rings (e.g., after some point in time t_0, a node in ring i waits $C(i \mod 4)$ rounds before starting with coloring).

B. Time Synchronization and Ring Discovery

In the previous section, we assumed that rings are synchronized and each node knows its ring number r (i.e., distance from the sink). In this section, we show how this can be accomplished.

In the protocol for stars described in Section V, the parent sent a beacon message to synchronize its children. This approach can be adopted to the ring structure (and hence also to trees) as follows. For ring discovery, we also include a *level* field in the beacon message. The sink emits such a beacon with level 1. Nodes that receive this beacon set r to the level contained in the beacon and do themselves emit a beacon message with level $r+1$. In addition, the node must ignore the beacon sent by its children (which will contain level $r+2$).

During the setup phase, such a beacon broadcast is performed before each round of the coloring algorithm to keep the nodes synchronized.

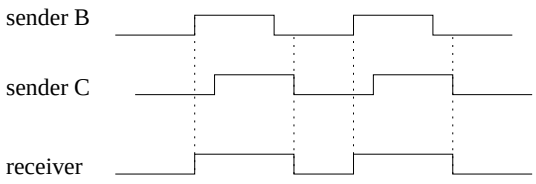

Fig. 5. Identical transmissions by two senders with small synchronization errors. The receiver will see slightly stretched "1" bits and slightly compressed "0" bits.

A beacon broadcast can be used to synchronize the receivers with respect to each other with high precision,

since the broadcast is received almost concurrently by all receivers [4]. Since the MAC protocol knows the size of the beacon message and bit lengths and has total control over access to the medium, the receivers can also accurately synchronize to the sender of the beacon [5]. As in [4], multiple beacon transmissions can be used to compensate the clock drift of the hardware clocks of the nodes, which allows the nodes to stay synchronized even longer after the transmission of a beacon.

In the initial ring structure used during tree setup, each node may receive the beacon from two or more (already synchronized) parents concurrently. In Figure 4 (b), for example, the node in ring $i+1$ will receive the beacon concurrently from nodes B and C in ring i. According to our communication model, these identical transmissions are merged, such the receiver will see the "or" of all transmissions. If the parents have a small mutual synchronization error, then the receiver will see slightly stretched "1" bits and compressed "0" bits as illustrated in Figure 5. However, if the duration of a bit is long compared to synchronization errors, this can be tolerated by the receiver. Hence, the bit length has to be selected appropriately. See Section VII-B for a discussion of this issue.

If the receiver synchronizes to the merged bit sequence of all its parents, it will effectively synchronize to the average of its parents. As a result, multiple parents improve the robustness of synchronization with the parents. Additionally, the synchronization precision errors among neighboring receivers that share multiple parents will benefit from this averaging process. We will analyze synchronization performance in Section VII-B.

C. Maintenance

So far we have described the setup phase of BitMAC that organizes the nodes of a deployed network into an interference-free tree with the sink at the root. In this section we discuss what happens if nodes are added or removed after this initial setup. Note that node mobility can be interpreted as removing a node and adding it again at a different place. Please keep in mind that we assumed in Section II-B that node mobility and node additions are rare events.

Let us first consider what happens if a node is removed temporarily or permanently, or if communication is temporarily disturbed. In dense networks, each node will typically possess communication links to multiple parents in the initial ring structure. As the last step of the setup procedure in Section VI-A, each node was informed of the channels of all its parents, such that the node could choose one parent by selecting the respective channel. We also noted that this selection can be changed at any time without the need for repeating tree setup. Hence, if the selected parent fails, a node can switch over to another parent simply by selecting the appropriate communication channel. If a node "reappears" after a temporary communication failure, no further actions are required. Note that parent re-selection can also be used to balance the load of the parents or to select a parent with the best communication link etc.

If a node in ring i runs out of operational parents, the tree is either partitioned or there are connections to the remainder of the tree via neighbors in ring i or $i+1$. If the node scans all communication channels and does not receive any beacon messages, the network is partitioned. If the node does receive a beacon, then the node will effectively move to another ring $> i$, since the sender of the beacon in ring $\geq i$ will become the new parent of the node. In this case, the setup phase must be repeated. For this, a special bit is used during signaling of send requests. If a parent receives a "1" during this bit slot, it will propagate the request to its parent and so on, until the sink is informed, which eventually initiates a reconstruction of the tree.

If a node is added to the network, it will first scan the communication channels for beacon messages from other nodes. Upon receiving a beacon, the node will signal a rebuild request to its parent to enforce a reconstruction.

D. Operation Phase

Up to now we have discussed setup and maintenance of BitMAC. In this section we discuss the operation phase, where data are transmitted up and down the tree.

For synchronization, the procedure described in the previous sections is used as well. However, at this point every node has selected a single parent, and the stars operate largely independent of each other (see below for a detailed discussion of this issue). Each node will therefore receive the beacon from its single parent only, which is sufficient to synchronize a parent and its children.

As discussed in Section V, the MAC protocol for the stars proceeds in rounds, where each round starts with a beacon broadcast. Using the procedure described in Section VI-B, the rounds are synchronized among the stars. However, in contrast to the setup phase, each node will receive the beacon only from a single parent and bit-level synchronization is only required among a parent and its children. Synchronization requirements across different stars are rather relaxed, since the stars operate independent of each other due to the interference-free channel assignment.

The rounds in all stars are of equal length. Hence, the number of time slots for data transmission in a round

is also limited. If in a star more children signal a send request than available time slots, the parent will schedule the maximum possible number of transmission for the round. Children which have not been assigned a time slot will retry in the next round. However, the parent remembers the children with rejected send requests to give them preference in the next round(s).

Note that a node in ring i has to act both as a parent for communication with children in ring $i+1$ and as a child for communication with its parent in ring $i-1$. Therefore, a node will act as a parent in even rounds and as a child in odd rounds.

So far, BitMAC enables nodes to send data to its parent or to its children in the tree. Various approaches can be used to route data over multiple hops. For the applications described in Section II-B, data is either sent from nodes to the sink, or from the sink to some or all nodes, which can be easily implemented. However, tree routing protocols can be used to exchange data between any two nodes in the network. Directed diffusion [9] or TinyDB [16] could also be easily adopted to BitMAC.

VII. Evaluation

To evaluate BitMAC, we apply a mix of experiments involving actual hardware, analysis, and simulation. Experiments are used to verify the basic operation of the protocol elements on a few nodes. The obtained results are used analytically or in simulations to evaluate BitMAC in larger networks.

In particular, we will analyze the impact of the number of available communication channels, the precision of time synchronization, as well as delays and protocol overhead of BitMAC. For this evaluation, we assume a perfect channel (i.e., no bit errors).

A. OOK Bit Transmission

In a first experiment we validate our communication model by showing that a node can indeed receive the bitwise "or" of the transmissions of two other nodes.

For the hardware experiments we used BTnode3 [28] sensor nodes, which provide a Chipcon CC1000 low-power radio. Although this radio does not directly support OOK transmission, an application note [29] describes how to emulate it. For this, the transmitter sets the frequency separation to 0 Hz and switches the power amplifier in the transmitter section on and off to transmit single bits. On the receiver side, the received signal strength indicator (RSSI) is used to decide whether a "1" is transmitted or not. A "1" is assumed if the RSSI value is above a fixed threshold. The built-in analog-to-digital converter (ADC) of the ATmega128 was used to measure the analog RSSI output. The BTnode3 provides a 7.328 MHz system clock, which allows a sampling rate of 250 kilo samples per second (i.e., one sample every 4μs).

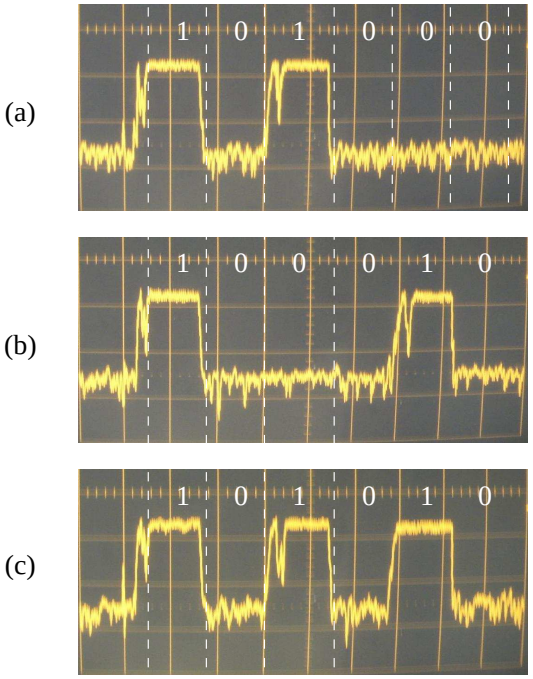

Fig. 6. RSSI measurements for reception of OOK data (time 0.2 ms/div, voltage 0.5/div), low voltage represents a "0" bit. (a) First node sending "101000" (b) Second node sending "100010" (c) Both nodes sending.

Note that radios with direct support for OOK and with digital signal output (such as the RFM TR 1000 or newer Chipcon radios) provide much better results. Hence, our experiments with the Chipcon CC1000 can be considered as worst-case results.

To verify OOK transmissions, we used three nodes in the setup shown in Figure 7 (b). Two sender nodes are triggered concurrently via a wire to start sending their bits via radio. A receiver hears both transmissions. As illustrated by the oscilloscope images in Figure 6, the senders transmit the bit sequences (a) "101000" and (b) "100010". The third node receives (c) "101010". The images were obtained by connecting the analog RSSI output of the receiver to the oscilloscope. A low voltage represents a "0" bit.

B. Time Synchronization

In a second experiment, we evaluate the precision of the time synchronization approach described in Section VI-B using the setups shown in Figure 7 (a-d). All nodes are connected by a wire to trigger the start of the experiment at t_0. In (a), the top node starts transmission

of a single "1" bit at t_0, which will be received by the bottom node. In (b), the two top nodes start transmitting a "1" bit concurrently at t_0. The receiver (i.e., bottom node) measures the point in time t_c corresponding to the center of the received bit by averaging the points in time corresponding to the rising (t_l) and the falling edge (t_r) of the "1" bit. By observing the variation of t_c over multiple runs, we can estimate the precision of time synchronization. In (c) and (d), the top node starts transmitting a "1" bit at t_0. After receiving the bit and measuring t_c, the middle node(s) start(s) transmitting a "1" bit at $t_c + t_x$ with a small fixed delay t_x. The bottom node measures t_c using the (or-ed) bit received from the middle node(s).

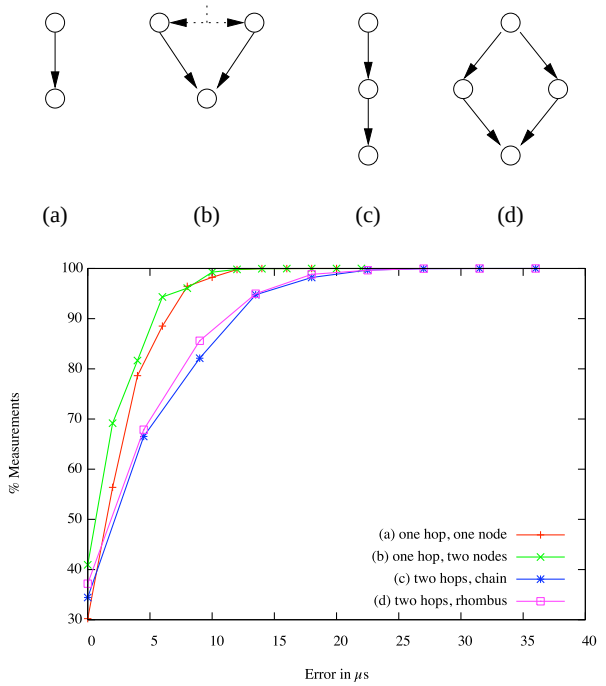

Fig. 7. Experiment setups and results for time synchronization. (a) one hop, one sender (b) one hop, two senders (c) two hops, chain (d) two hops, rhombus.

During the measurement of t_c we observed rare (less than 1%) larger variations for t_l, while t_r is rather stable. However, since the duration of the bit t_{bit} is known, we can detect these errors and correct them by setting $t_l := t_r - t_{bit}$ in such cases. With this fix, we performed the above measurements 100000 times to determine the maximum variation t_{err} of t_c. The results are shown in the diagram in Figure 7. A point on the curve indicates the percentage (y axis) of the measurements for which the variation of t_c is bounded by an interval of a certain length (x axis).

The diagram shows than the maximum variation in the single-hop experiments (a) and (b) is about 20μs. For the two-hop experiments we would expect about twice the maximum variation of the single-hop measurements. In the experiments we observed a maximum variation of about 36μs, which is slightly smaller then the expected value. As mentioned in Section VI-B, the results are indeed slightly better if multiple senders are involved (cases (b) and (d)).

As mentioned in the previous section, we can expect that the above results can be significantly improved if the radio directly supports OOK modulation and provides a digital signal output. Indeed, the authors of [12] report a maximum error of 2μs for the setup (a) when using the RFM TR 1000, which is a 10-fold improvement over our measurement results.

Let us now consider the worst case synchronization error in larger networks. If t_{err} is the worst case error for a one-hop setup, then the worst case error after r hops is rt_{err}. Note, however, that we can expect better results if the network is dense, because then each node has many parents in the ring structure.

In Section VI-B we mentioned that synchronization errors lead to stretched or compressed bits. Hence, the duration of a bit t_{bit} has to be long enough to tolerate these effects. Let us assume that bits can still be correctly received if their length is reduced to $t_{bit}/2$ due to synchronization errors. If the radius of the ring is r, then t_{bit} must be at least $2rt_{err}$. For example, if $r = 7$ and $t_{err} = 20\mu$s, then $t_{bit} \geq 280\mu$s, which equals a bit rate of about 3.5 kilobit per second. With $t_{err} = 2\mu$s, we would obtain $t_{bit} \geq 28\mu$s or 35 kilobit per second. Note, however, that these "long" bits only have to be used during the setup phase for control traffic. Data transmissions can use any bit rate and modulation scheme that is supported by the radio.

Let us finally consider for how long two nodes can maintain synchronization using their hardware clocks, assuming they are perfectly synchronized initially. With a bounded clock drift ρ_{max}, at least $1/(4\rho_{max})$ bits can be transmitted until the synchronization error can result in a bit duration smaller than $t_{bit}/2$. For example, the oscillator used on the BTnode3 provides $\rho_{max} = 100$ ppm according to the data sheet, which allows to send up to 2500 bits without resynchronization. Note that multiple synchronization rounds could be used to compensate the clock drift to reduce the frequency of synchronization.

C. Network Density

In Section VI-A we assumed the knowledge of a constant C, such that the coloring problem can be solved with C colors. We also mentioned that in practice C is

set to the number of available radio channels. Hence, we have to examine which network density can be supported by BitMAC given a certain C.

It is a well-known fact that a graph with maximum node degree Δ can be colored with $\Delta + 1$ colors. Since the maximum two-hop degree (i.e., number of nodes within distance ≤ 2 hops) of a graph with maximum degree Δ is at most Δ^2, we would need $\Delta^2 + 1$ colors for a standard two-hop coloring, where any two nodes with distance ≤ 2 must be assigned different colors. However, recall from Section VI-A, that in our coloring algorithm the effective two-hop neighborhood of a node n in ring i is the set of nodes in ring i that share a common neighbor with n in rings $i-1$ or $i+1$. We will call the respective two-hop degree the *two-hop ring degree*. We can expect that the two-hop ring degree of nodes in a plane is significantly smaller than Δ^2.

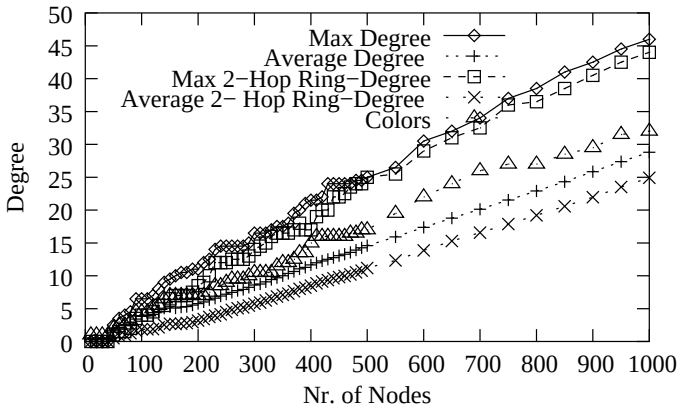

Fig. 8. Relationship between node density and node degrees. Nodes with communication range 1 are randomly placed in a 10-by-10 area.

To verify this expectation we performed a set of simulations, where a given number of nodes with communication range 1 is randomly placed in a 10-by-10 rectangular area. The sink is placed at the center of the area. For different numbers of nodes we determined the average and maximum degree, as well as the average and maximum two-hop ring degree. We performed 150 simulation runs and computed averages. The results are depicted in Figure 8. As can be seen, the average and maximum two-hop ring degrees are even smaller then the average and maximum degree, respectively. The figure also shows the actual minimum number of colors ("Colors" in the legend) required by the coloring algorithm described in Section VI-A. This value is obtained by letting the algorithm always select the "smallest" available color (i.e., the color represented by the smallest integer ID) from the palette.

Recall from Section II-A that the Chipcon CC1000 supports up to 35 channels in the 868 MHz ISM band. With this setup we can expect to be able to color random graphs with an average degree of up to about 20.

If the number of channels is not sufficient to color the graph, then some nodes will end up without a color assignment after the coloring algorithm. Such nodes cannot participate in the sensor network. However, the remaining nodes will form an operational network. Due to the high degree, this network is most likely connected.

D. Setup Phase

In this section we examine the setup phase of our protocol, in particular we derive the amount of time t_{setup} that is needed to setup a network with given parameters. t_{setup} depends on the following parameters: the duration t_{bit} of a single bit as examined in Section VII-B, the number C of channels, the length L_{mac} of a MAC address in bits, the length L_{beacon} of the beacon in bits, the time t_{rxtx} needed to switch the radio between transmit/receive modes, and the radius r of the ring.

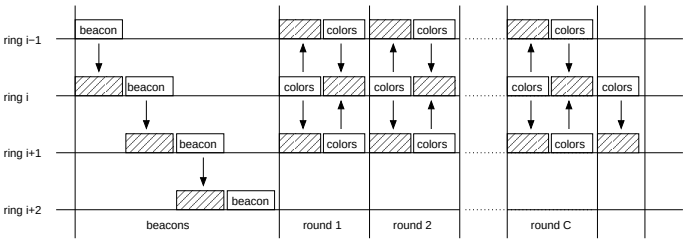

Fig. 9. One parallel step of the coloring algorithm on ring i. Shaded packets indicate data receptions.

As explained in Section VI-A, rings with numbers $i, i+4, i+8, i+12, ...$ can be colored in parallel. Hence, 4 parallel coloring steps are required. One coloring step is illustrated in Figure 9. Each step consists of a beacon broadcast for synchronization, followed by the coloring algorithm. Recall from Section VI-A that the coloring algorithm consists of C rounds. In each round, a parallel vectorial MAX operation is performed. In a final step, colored nodes announce assigned colors to the children. Let us consider the duration t_{step} of such a parallel coloring step, which can be expressed as follows:

$$\begin{aligned} t_{step} &\leq t_{beacon} + C\, t_{round} + t_{announce} \\ t_{beacon} &\leq 4\,(L_{beacon}\, t_{bit} + t_{rxtx}) \\ t_{round} &\leq 2\, L_{mac}\,(C\, t_{bit} + t_{rxtx}) \\ t_{announce} &\leq C\, t_{bit} + t_{rxtx} \end{aligned}$$

Due to the delay caused by the beacon broadcast, coloring of ring r is started at time $r t_{beacon}/4$ after the sink has initiated the setup. Hence, we obtain for the setup time:

$$t_{setup} \leq 4\, t_{step} + r\, t_{beacon}$$

Let us consider the sample network we used for the simulations in Section VII-C with 800 nodes. With $r = 7$, $C = 35$, $L_{mac} = 16$, $L_{beacon} = 110$, $t_{rxtx} = 250\mu$s, we obtain $t_{setup} \leq 48$s for $t_{bit} = 280\mu$s, or $t_{setup} \leq 6$s for $t_{bit} = 28\mu$s.

E. Operation Phase

In this section we evaluate the operation phase, where payload data is transmitted from nodes towards the sink. Let us assume that each round consists of S slots for data packets. Recall from Section VI-D that each node acts as a child in even rounds and as a parent in odd rounds. Hence, when a node wants to transmit, it may have to wait up to two rounds until it can transmit a send request to its parent. Then the packet is forwarded one hop towards the sink in every round. Since the overlay tree is a shortest-path tree from the sink to all nodes, packets are always forwarded on the shortest path to the sink. Hence, it takes at most $i + 2$ rounds to deliver a packet from a node in ring i to the sink, provided that a free slot can be allocated in each round. If a free slot cannot be assigned immediately, then a node may have to wait for $2\lfloor C/S \rfloor$ rounds until a slot is assigned, since a parent cannot have more than C children.

As with most other MAC protocols, there is a trade-off between latency and energy consumption in BitMAC. If the duration of a round (i.e., S) is increased, the latency will increase and energy consumption will decrease (due to fewer beacon transmissions). The energy consumption of an idle network is dominated by the duration of a round and the length of the beacon. In every idle round, each non-leaf node has to receive and transmit a beacon (without acknowledgments and schedules), and has to listen to C send-request bits from its children. A child node only receives and forwards the beacon. Hence, we obtain for the average radio-on time:

$$t_{on} \leq 2\ (L_{beacon}\ t_{bit} + t_{rxtx}) + C\ t_{bit}/2$$

Let us consider an example, where $C = 35$, $t_{bit} = 52\mu$s (19200 baud), $t_{rxtx} = 250\mu$s, and $L_{beacon} = 110$. With these parameters we obtain $t_{on} \leq 13$ms. If the duration of a round is 200ms, then $S = 9$ data packets of 32 bytes each can be delivered in each round. The duty cycle of the nodes in an idle network is then about 6% on average.

If the network is not idle, then parents have to send schedules and acknowledgments, resulting in an additional $2C$ bits. Overall, the time per round that cannot be used for data transmissions is:

$$t_{overhead} \leq 2\ L_{beacon}\ t_{bit} + 4\ t_{rxtx} + 3\ C\ t_{bit}$$

Note that $t_{overhead}$ can be interpreted as the overhead of our protocol compared to an ideal protocol, where all nodes know a priori when to transmit data packets without collisions. For the above example we obtain $t_{overhead} \leq 18$ms, such that about 9% of the bandwidth cannot be used for payload data transmissions if a round lasts for 200ms.

VIII. Discussion

In this paper we assumed that links between nodes are either error-free or nonexistent. A more realistic model to characterize communication links is the distinction of communication range, in which the communication is more or less error-free, and interference range, in which data transmitted from a node A to node B does harm node C's reception of data sent by node D. By design of BitMAC, there are no collisions between nodes in communication range, so we would like to discuss issues related to nodes in interference range here.

Nodes on the same ring either transmit or receive at the same time, hence neither collisions nor interference do occur on one ring. Also, by the way the rings are constructed, we conclude that transmissions from nodes on ring $i + 1$ to nodes on ring i do not interfere with parallel transmissions from ring $i - 1$ to $i - 2$ (3 hops). Even if they would, we could insert a "buffer ring" to avoid such interferences.

The vulnerable part of BitMAC is the channel and ID assignment. If there are two nodes A and B which do not have a common child or parent, they might choose the same channel independently. Not having a common child refers to not having a node in the communication range. Yet still node A might have a child node C which might be in interference range to node B and disturb transmissions from B's child nodes.

One way to alleviate this problem is to ensure that such a node C is seen as a common child node of A and B during coloring. This can be done by temporarily increasing the radio power in such a way that the communication range during this phase equals the interference range of the later communication. It is important that the radio power is increased only during coloring, otherwise the situation would not improve. By doing this, additional parent-child links are created which are not available at normal transmission power. Therefore, all links should be validated by an additional link probing phase. Real-world deployment will allow us to evaluate this approach.

IX. Related Work

Besides contention-based algorithms such as [3], [15], [22], [25], a number of MAC protocols have

been proposed that avoid contention using time or frequency-division multiplexing. However, most of these approaches suffer from serious limitations or drawbacks as discussed below (e.g., long-range communication, static schedules, low bandwidth utilization, high latency, out-of-band time synchronization, some protocol elements introduce collisions, regular network topologies required).

The approach in [1] partitions the network into large clusters. In each cluster, a centralized scheduler assigns time slots to nodes via long-range communication. After the schedule has been assigned, nodes communicate via multi-hop, short-range communication.

SS-TDMA [13] uses a fixed schedule throughout the lifetime of the network and operates only on regular topologies such as square and hexagonal grids. Due to the fixed schedule, bandwidth utilization is low under varying traffic conditions.

LEACH [7] partitions the network into clusters, where the cluster head assigns a TDMA schedule to the nodes in the cluster, such that nodes can send to the cluster head via single-hop transmissions. Cluster heads are rotated to distribute energy consumption equally among all nodes. However, since cluster heads communicate via long-range radio with the sink, all nodes must be capable of long-range communication. BMA [14] is similar to LEACH but provides dynamic slot allocations within each cluster by means of single-bit transmissions similar to our protocol for star networks.

With TRAMA [18], TDMA scheduling is replicated over the nodes of the network. The schedule is adapted to long-term traffic flows through the network. The protocol assumes an out-of-band mechanism for time synchronization and exhibits a high latency.

In EMAC [24], some nodes are assigned a slot in a TDMA schedule and can transmit during this slot. All nodes must listen during the slots of all its neighbors to be able to receive data, which can represent a significant energy overhead. Nodes that do not own a slot use contention to compete for the right to transmit in a slot owned by a neighbor. An out-of-band mechanism for time synchronization is assumed. Due to the fixed slot assignment, the bandwidth utilization is rather low.

LMAC [23] is a variant of EMAC, where nodes use contention to compete for slots. All nodes must listen to detect collisions. Messages are not acknowledged and an out-of-band mechanism for time synchronization is assumed. Due to the fixed slot assignment, the bandwidth utilization is rather low.

The algorithm presented in [21] establishes a collision-free TDMA schedule that adopts to varying network topologies, but uses an underlying contention-based MAC for establishing the schedule. An out-of-band mechanism for time synchronization is assumed.

Two recent publications consider concurrent transmissions of multiple senders. With SDJS [12], synchronized nodes can decide to transmit a jam signal, such that receivers can detect two cases: (1) no transmission, or (2) one or more jamming nodes, which can be considered as a form of “or” channel. This mechanism is used to estimate the number of neighbors in an efficient way. In [17], a very similar approach is mentioned. To transmit a “1” bit, a sender emits a carrier signal. If a receiver detects this carrier signal or a collision, then a “1” bit is received. This feature is used in [17] to implement data aggregation. No details on a possible implementation on real hardware are given.

In [8], a time synchronization approach is described which is somewhat similar to our approach to time synchronization. There, a designated node emits a sequence of pulses. Nodes that hear this pulse sequence predict when the next pulse will be sent and transmit a synchronous pulse themselves at the predicted point in time. Eventually, all nodes in the network will hear and transmit synchronous pulse sequences.

X. Conclusion

We have presented and analyzed BitMAC, a deterministic, collision-free, and robust protocol for dense wireless sensor networks. BitMAC is based on an “or” channel, where synchronized senders can transmit concurrently, such that a receiver hears the bitwise “or” of the transmissions. Using the BTnode3 platform, we have shown the practical feasibility of this communication model and analyzed the performance of time synchronization. We gave deterministic bounds on the execution time of all protocol elements and showed that the protocol overhead is small compared to an ideal protocol. We are currently implementing the complete protocol on the BTnode3 platform and intend to evaluate it in a setup with about 100 nodes.

BitMAC includes a number of parameters (duration of a round, bit length, C) which need to be properly selected in order to efficiently support a specific network setup. Although we illustrated by examples throughout the paper that a reasonable default setting for the parameters can support a wide range of networks, it is desirable to automatically adapt these parameters at runtime to the actual network setup. For example, although C is limited by the number of available radio channels and has to be big enough to ensure a correct coloring of the network, a smaller C results in better performance and less protocol overhead. The duration of a round could be

adapted to the actual traffic in order to find the best trade-off between latency and energy consumption. Using a variable bit length for control traffic that depends on the distance from the sink (i.e., the ring number) could eliminate the need for a predefined bit length.

XI. Acknowledgments

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

We would like to thank Ankur Agiwal who tested the OOK emulation on the BTnode3 and implemented our first Chipcon CC1000 driver.

References

[1] K. Arisha, M. Youssef, and M. Younis. Energy-Aware TDMA-Based MAC for Sensor Networks. In *IMPACCT*, New York, USA, May 2002.

[2] R. Beckwith, D. Teibel, and P. Bowen. Pervasive Computing and Proactive Agriculture. In *Adjunct Proc. PERVASIVE 2004*, Vienna, Austria, April 2004.

[3] A. El-Hoiydi, J. D. Decotignie, C. Enz, and E. Le Roux. WiseMAC, an Ultra Low Power MAC Protocol for the WiseNET Wireless Sensor Network. In *SenSys 2003*, Los Angeles, USA, November 2003.

[4] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. In *OSDI 2002*, Boston, USA, December 2002.

[5] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys 2003)*, November 2003.

[6] D. Ganesan. Network Dynamics in Rene Motes. Powerpoint Presentation, January 2002.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Sensor Networks. In *HICCS*, Hawaii, USA, January 2000.

[8] A. Hu and S. Servetto. Asymptotically Optimal Time Synchronization in Dense Sensor Networks. In *WSNA*, San Diego, USA, September 2003.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *6th Intl. Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, USA, August 2000.

[10] Öjvind Johannson. Simple Distributed $\Delta + 1$ Coloring of Graphs. *Information Processing Letters*, 70:229–232, 1999.

[11] C. Kappler and G. Riegel. A Real-World, Simple Wireless Sensor Network for Monitoring Electrical Energy Consumption. In *EWSN 2004*, Berlin, Germany, January 2004.

[12] A. Krohn, M. Beigl, and S. Wendhack. SDJS: Efficient Statistics in Wireless Networks. In *ICNP*, Berlin, Germany, October 2004.

[13] S. Kulkarni and M. Arumugam. TDMA Service for Sensor Networks. In *ADSN*, Tokyo, Japan, March 2004.

[14] J. Li and G. Lazarou. A Bit-Map-Assisted Energy-Efficient MAC Scheme for Wireless Sensor Networks. In *IPSN*, Berkeley, USA, April 2004.

[15] G. Lu, B. Krishnamachari, and C. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Sensor Networks. In *WMAN*, Santa Fe, USA, April 2004.

[16] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *OSDI 2002*, Boston, USA, December 2002.

[17] S. Olariu, A. Wadaa, L. Wilson, and M. Eltoweissy. Wireless Sensor Networks - Leveraging the Virtual Infrastructure. *IEEE Network*, pages 51–56, July 2004.

[18] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves. Energy-Efficient, Collison-Free Medium Access Control for Wireless Sensor Networks. In *SenSys 2003*, Los Angeles, USA, November 2003.

[19] R. Riem-Vis. Cold Chain Management using an Ultra Low Power Wireless Sensor Network. In *WAMES 2004*, Boston, USA, June 2004.

[20] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a Sensor Network Expedition. In *EWSN 2004*, Berlin, Germany, January 2004.

[21] T. Herman and S.Tixeuil. A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks. In *ALGOSENSORS*, Turku, Finland, June 2004.

[22] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *SenSys 2003*, Los Angeles, USA, November 2003.

[23] L. van Hoesel and P. Havinga. A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches. In *INSS 2004*, Tokyo, Japan, June 2004.

[24] L. van Hoesel, T. Nieberg, H. Kip, and P. Havinga. Advantages of a TDMA Based, Energy-Efficient, Self-Organizing MAC Protocol for WSNs. In *IEEE VTC Spring*, Milan, Italy, May 2004.

[25] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *IEEE Infocom 2002*, New York, USA, June 2002.

[26] J. Zhao and R. Govindan. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *SenSys 2003*, Los Angeles, USA, November 2003.

[27] Berkeley Motes. www.xbow.com/Products/_Wireless_Sensor_Networks.htm.

[28] BTnodes. www.btnode.ethz.ch.

[29] Chipcon Application Note AN016 – CC1000 / CC1050 used with On-Off Keying. www.chipcon.com.

[30] EYES: Energy Efficient Sensor Networks. eyes.eu.org.

[31] Multimodal Networks of In-situ Sensors (MANTIS). mantis.cs.colorado.edu.

[32] Scatterweb Embedded Sensor Board. www.scatterweb.net.

[33] Smart-Its Particle Computer. particle.teco.edu.