Electronic Communications of the EASST
Volume 17 (2009)

Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009 (WowKiVS 2009)

Multi-hop Clock Synchronization in Wireless Ad-Hoc Networks

Dennis Christmann, Reinhard Gotzhein, Thomas Kuhn

12 pages

Guest Editors: M. Wagner, D. Hogrefe, K. Geihs, K. David
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Multi-hop Clock Synchronization in Wireless Ad-Hoc Networks

Dennis Christmann[1], **Reinhard Gotzhein**[2], **Thomas Kuhn**[3]

[1] christma@cs.uni-kl.de
[2] gotzhein@cs.uni-kl.de
[3] kuhn@cs.uni-kl.de
Networked Systems Group
University of Kaiserslautern, Germany

Abstract: In this paper, we introduce *Black Burst Clock Synchronization* (*BBCS*), a novel protocol for multi-hop time synchronization in wireless ad-hoc networks, located at MAC level. *BBCS* is based on the exchange of synchronized *tick and time frames*, which are protected against collisions by a special encoding using *black bursts*. It provides a deterministic upper bound for clock offset that only depends on maximum network diameter, and on the used transceiver hardware. *BBCS* has low complexity in terms of communication, computation, storage, structure, and energy consumption. It provides low and deterministic convergence delay, and is robust against node movements and node failures. In this work, we introduce BBCS, provide a formal analysis of its properties, and evaluate the required overhead for clock-synchronizing a multi-hop wireless ad-hoc network.

Keywords: clock synchronization, ad-hoc networks, black bursts, BBS, MacZ

1 Introduction

The objective of time synchronization in communication networks is to keep all local clocks in synchrony. This is important for user-level applications (e.g. data fusion in wireless sensor networks, networked control systems [ASSC02]) as well as for system-level applications (e.g. duty cycling, network-wide medium slotting [YHE02]). General requirements on time synchronization protocols are:

- provision of a small and/or bounded *clock offset*, i.e. an accurate time basis
- fast and/or bounded *convergence*, i.e. a short and ideally predictable delay until (re-)synchronization is achieved
- low and/or bounded *complexity* concerning, e.g., computation, communication, storage, energy, and structure
- high robustness against topology changes such as node movements and node failures

More specific requirements on time synchronization protocols depend on concrete application requirements and network topology. For instance, data fusion applications require a small average clock offset for time stamping of sensor values. On the other hand, a small *and* bounded clock offset is needed for duty cycling or network-wide medium slotting. Similar considerations apply to convergence delay, complexity, and robustness.

In our previous work, we have introduced Black Burst Synchronization (*BBS*), a protocol for network-wide tick synchronization [GK08]. We have argued that tick synchronization is

sufficient for multi-hop medium slotting, and we have devised a MAC layer protocol called *MacZ* that provides both exclusive medium access based on slot reservations, and contention-based access with priorities [BGK07]. *BBS* is based on the exchange of synchronized tick frames, which are protected against collisions by a special encoding with black bursts [KI07]. It provides small and bounded tick offset and convergence delay, has low complexity, and is robust against topology changes.

In this work, we extend *BBS* by an algorithm for time synchronization called *Black Burst Clock Synchronization* (*BBCS*). The idea is to use global ticks as reference points in time, and to propagate the time values of these global ticks in special *time frames*. For this purpose, we introduce two different frame encodings with black bursts called *cooperative* and *arbitrating encoding*. Both encodings are resistant against collisions, and guarantee upper bounds for convergence delay. Being based on *BBS*, *BBCS* preserves the properties regarding offset, complexity, and robustness. Neither BBS nor BBCS rely on static network topology. We have formally specified *BBCS* with the Specification and Description Language (SDL [ITU07]), and have implemented a subset of *BBCS* on MICAz motes [Cro].

The rest of this paper is structured as follows: In Section 2, we introduce basic concepts of tick and time synchronization, and outline *BBS*, our tick synchronization protocol upon which *BBCS* is based. We then introduce two types of encoding with black bursts called cooperative and arbitrating encoding in Section 3. In Section 4, we present *BBCS*, a novel protocol for time synchronization, and a quantitative analysis of the protocol. Section 5 surveys related work, Section 6 draws conclusions and lays out future work.

2 Multi-hop tick synchronization with BBS

Based on and extending [SBK05], we use the following terminology: By *real time*, we refer to the (global) time t as measured by a (perfect) clock. A *real tick* is a (global) reference point in time. At real time t, the *local time* of some node A is given by the value $c_A(t)$ reported by its (physical) clock c_A. The *clock offset* is the difference between the local time reported by clock c_A and real time, i.e. $c_A(t) - t$, or $c_A(t) - c_B(t)$ relative to clock c_B of some node B. Similarly, the *tick offset* of some node A is the difference between the real time t_0 at which a real tick occurs and the real time $t_0^{'}$ to which it is associated by A, i.e. $t_0^{'} - t_0$. A clock c_A has a *clock rate*, i.e. the speed $c_A^{'}(t)$ at which it progresses at real time t (1st derivative). The *clock skew* is the difference between the rates of the clock c_A and the perfect clock at real time t, i.e. $c_A^{'}(t) - 1$, or $c_A^{'}(t) - c_B^{'}(t)$ relative to clock c_B of some node B.

Differing clock rates lead to increasing clock offsets and are the reason why nodes have to resynchronize their clocks from time to time. Please note that real time can be defined as a real tick to which a real time value is associated. Thus, time synchronization yields strictly more information than tick synchronization. However, tick synchronization is less expensive and is sufficient for system-level applications such as multi-hop medium slotting.

BBCS is based on multi-hop tick synchronization, as accomplished, for instance, by Black Burst Synchronization (*BBS*) [GK08]. The conceptual foundation of *BBS* is shown in Figure 1. At real time t_0 (marking a real tick), a master node[1] sends a *tick frame*, a frame used for syn-

[1] For the correct operation of *BBS*, it is irrelevant which node takes the master role.

chronization, which is protected against collisions by a special encoding using black bursts (see [KI07] and Section 3). This frame carries a round number (initially: 1) and is forwarded by every node after the fixed round duration d_{round}, where the round number is incremented by 1. Because collisions of synchronized (identical) tick frames are non-destructive, nodes may transmit tick frames of the same round simultaneously. The number of rounds, and thus the duration of a synchronization phase, is limited by $n_{maxHops}$, the maximum network diameter in hops[2].

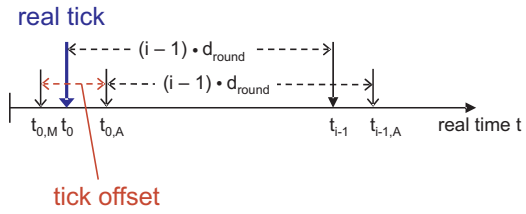

Figure 1: Tick offset.

Ideally, reception of a tick frame at some node A in round i starts at real time t_{i-1} (see Figure 1). The ideal delay $d_{0,i-1} = t_{i-1} - t_0$ is computed as product of round number (= hop distance -1) between the master node and node A and the duration of a tick frame round, i.e. $(i-1) \cdot d_{round}$. This ideal delay is increased, for each hop, by the variable signal propagation delay d_{prop}, and by d_{maxCCA}, the variable delay of recognizing the start of the tick frame reception called *clear channel assessment (CCA) delay*. Therefore, the start of a tick frame reception at node A is not recognized at real time t_{i-1}, but at $t_{i-1,A}$ (see Figure 1).

Since node A knows round number i and round duration d_{round}, it is able to calculate the ideal delay $d_{0,i-1}$. Subtracting $d_{0,i-1}$ from $t_{i-1,A}$ yields $t_{0,A}$, which is the real tick as perceived by node A. This yields the tick offset of $t_{0,A} - t_0$. For this offset, an upper bound can be deduced, depending linearly on $n_{maxHops}$, d_{prop} and d_{maxCCA} only. Note that to refer to $t_{0,A}$, node A needs its local clock value only: $c_A(t_{0,A}) = c_A(t_{i-1,A}) - d_{0,i-1}$. Thus, each node has a local reference point in time of deterministic accuracy regarding the corresponding real tick, which is, for instance, sufficient for multi-hop medium slotting.

Due to differing clock frequencies, the accuracy achieved every time a global reference tick has been established degrades over time. To keep it within deterministic bounds $d_{maxOffset}$, *BBS* resynchronizes periodically. For a required maximum tick offset, the period to be used has an upper bound that depends on maximum clock skew $r_{maxClockSkew}$, and on base tick offset $d_{maxBaseOffset}$, the tick offset achieved after resynchronization.

For a MICAz node with AT86RF230 transceiver [Atm], we have determined $d_{maxBaseOffset} = 16\,\mu$s per hop. For multi-hop medium slotting and a stationary network, it is sufficient to operate with the maximum base tick offset within 2 hops, i.e. $2 \cdot d_{maxBaseOffset} = 32\,\mu$s . We have determined the convergence time in a network with $n_{maxHops} = 10$ to be $61.4\,ms$ (duration to synchronize the whole network), yielding a synchronization overhead of 0.61%, if resynchronization is done every $10\,s$.

Apart from master-based *BBS* sketched above, there is also a decentralized version that can be used stand-alone or together with master-based *BBS* (serving as backup for master node failure). Decentralized *BBS* provides a deterministic upper bound for tick offset, too, which in addition to $n_{maxHops}$, d_{prop} and d_{maxCCA} depends on transceiver switching time (see [GK08] for details).

If networks meet, they merge into one synchronized network. Among nodes of different networks, there is a temporary lack of synchronization. This is detected by receiving unexpected tick frames, or by collisions of regular frames in reserved micro slots, and resolved (see [GK08]).

[2] The actual network diameter should not exceed this value, which can be preconfigured.

3 Cooperative and arbitrating encoding with black bursts

This section describes two encoding schemes called *cooperative and arbitrating encoding* for the wireless, collision-protected transmission of data, which are used by *BBCS* (see Section 4). With these encodings, it is possible to propagate any bit sequence, for instance, a time value, across the network in deterministic time, as derived from the maximum network diameter $n_{maxHops}$ and the properties of the transceiver hardware. Since both encodings are significantly less efficient than regular encodings for (collision-prone) data transmissions on MAC level, they are applied to certain control frames, in particular, tick frames and time frames, only. Both encodings require all network nodes to be tick-synchronized when collision-protected transmissions take place. They assume a deterministic maximum tick offset $d_{maxOffset}$ as provided by *BBS*. Furthermore, they assume that the medium is decomposed into macro slots, which are further subdivided into a fixed number of micro slots, and that all nodes have prior knowledge of (reserved) micro slots in which such transmissions may be started. This knowledge may be established during system configuration, or during network operation.

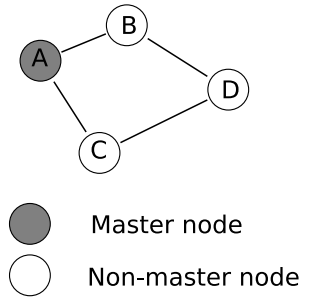

Figure 2: Example topology.

Similar to the encoding used in *BBS*, cooperative and arbitrating encoding use black bursts [KI07] (cf. Section 2). Conceptually, a black burst is a period of energy of defined length on the medium, transmitted without prior medium arbitration. When a black burst is detected, two pieces of information can be derived. First, the start of reception can be determined with high accuracy, and be used for synchronization purposes. Second, the length of a black burst can be used for encoding purposes. In case several black bursts overlap, a receiver can still derive both pieces of information, if the length is not changed significantly, i.e. if all black bursts are transmitted (almost) simultaneously. In the following, we use one black burst to encode a bit: a binary *1* corresponds to the transmission of a black burst, and "no transmission" (i.e. a black burst of length zero) stands for a binary *0*. Thus, a binary *1* is dominant, if several nodes are transmitting, yielding a (logical) OR-operation. We implement dominant black bursts as special MAC frames of defined length, with irrelevant contents, transmitted without prior clear channel assessment (CCA). Thus, we can use customary transceivers, without any hardware modifications.

To clarify the encodings introduced below, we use the topology in Figure 2 with a network diameter of $n_{hops} = 2$. Allowing for node mobility, we set the maximum diameter to $n_{maxHops} = 3$. The general frame structure of both encodings is shown in Figure 3. A frame starts with a dominant start of frame bit (SOF) to mark its beginning. This is followed by an arbitrary bit sequence, consisting, e.g., of data and checksum.

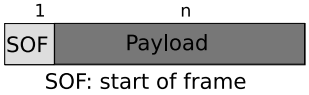

Figure 3: Frame structure with cooperative encoding.

Cooperative encoding can be used if nodes transmitting concurrently send the same bit sequence. This is, for instance, the case if a master node initiates a transmission, which is propagated hop-by-hop across the network by receiving nodes after a fixed delay. Also, it is possible

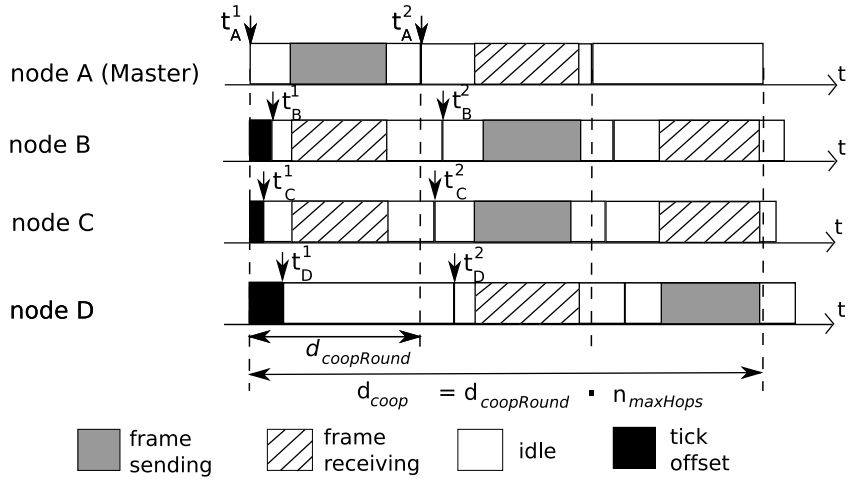

Figure 4: Cooperative encoding. A frame encoded with black bursts is sent over two hops.

that several nodes start transmission of the same bit sequence (almost) simultaneously[3], which is then forwarded accordingly. When using cooperative encoding, propagation of frames takes place in rounds, where in each round, the entire frame is sent. The duration of a round $d_{coopRound}$ is determined by the length of the frame (in bits) and the transmission time of a bit. To propagate a frame across the network, up to $n_{maxHops}$ rounds are needed.

A scenario based on topology in Figure 2 is shown in Figure 4. Recall that $n_{maxHops} = 3$, therefore, up to 3 rounds are required to propagate frames across the network. In the first round, transmission is initiated by some master node. In the scenario, node A takes this role and transmits the frame in the first round starting at local time t_A^1. Note that transmission starts in a reserved micro slot that is known to all network nodes, with a delay determined by the maximum tick offset $d_{maxOffset}$. Due to the underlying tick synchronization of BBS, this ensures that all other nodes have started listening on the medium when the frame is sent. As shown in Figure 4, nodes B, C, and D have a relative local tick offset regarding node A, yielding a local perception of the start of the first round at t_B^1, t_C^1, t_D^1, respectively.

In round 1, nodes B and C receive the frame sent by master node A and therefore become sending nodes in round 2 (see Figure 4). The start of round 2 is determined locally as $t_B^2 = t_B^1 + d_{coopRound}$ and $t_C^2 = t_C^1 + d_{coopRound}$, respectively, where $d_{coopRound}$ denotes the (fixed) round duration. This ensures that the transmissions in round 2 are sufficiently synchronized, preserving the properties of black burst encoded bit sequences regarding collision resistance. Receiving nodes are nodes A and D. A ignores the reception because it has already sent the frame in the last round and node D becomes the sending node in the final round 3.

Next, we analyze the duration d_{coop} for network-wide propagation of frames using cooperative encoding. This duration is given as

$$d_{coop} = n_{maxHops} \cdot d_{coopRound} \tag{1}$$

where $n_{maxHops}$ and $d_{coopRound}$ denote the maximum network diameter and the (fixed) round

[3] Points in time may either be preconfigured, or be agreed upon dynamically.

duration, which is given as

$$d_{coopRound} = (n+1) \cdot d_{coopBurst} + d_{processing} \tag{2}$$

Here, n is the length of the frame in bits, $d_{coopBurst}$ is the duration of a black burst transmission and guard intervals, and $d_{processing}$ is an additional time that a node needs to process the frame. Finally, $d_{coopBurst}$ is refined into

$$d_{coopBurst} = d_{burstTx} + d_{accessTx} + d_{pause} + 2 \cdot d_{maxBaseOffset} + 2 \cdot d_{macroSlot} \cdot r_{maxClockSkew} \tag{3}$$

$d_{coopBurst}$ depends on the black burst send delay and an additional pause duration that enables the senders to switch their transceivers between receiving/sending mode. It must also consider the maximum two-hop tick offset with an additional pause to enable receivers to distinguish two subsequent black bursts. Given that black bursts are implemented as special MAC frames, $d_{burstTx} = \frac{b}{r}$ corresponds to the duration of a MAC frame transmission with frame size b and transmission rate r (including frame preamble and checksum).

Arbitrating encoding can be used if nodes transmit concurrently, but not necessarily the same bit sequence. It has the effect that only those nodes transmitting the bit sequence with the highest value complete their transmission, and this value becomes known to all nodes as soon as the transmission is finished. Arbitrating encoding can, for instance, be used for network-wide medium arbitration, provided nodes competing for the medium send different bit sequences. When using arbitrating encoding, propagation of each bit of a frame takes place in rounds, where in each round, the current bit is propagated across the network. For each bit, $n_{maxHops}$ rounds are needed. This is repeated for each bit of the arbitration frame.

Figure 5 illustrates how arbitrating encoding works. We assume that nodes A, B, C, and D want to transmit frames consisting of bit sequences 110, 101, 011, and 111, respectively. We further assume that $n_{maxHops} = 2$, so for each bit, we need 2 rounds. Starting with the *first* bit of their bit sequences, nodes A, B, and D transmit a black burst in round 1, whereas node C stays silent because it sends 0. Thus, node C recognizes that the bit sequence of at least one other node has higher priority. As a consequence, it stops transmitting its own bit sequence, starts acting as repeater, and records the remaining bit sequence. For the first bit, this means that it forwards the received dominant bit in round 2, and records 1.

In round 1 of the transmission of the *second* bit, node C being repeater node is in receiving mode. Nodes A and D transmit a dominant bit, node B stays silent. Thus, nodes B and C receive a dominant bit, and node B becomes repeater, too. Now, both B and C act as repeaters, forward the received dominant bit in round 2, and record 1. In round 1 of the transmission of the *third* bit, D sends a dominant bit, while A stays silent. However, A is not in range of D, so it does not become repeater (yet). B and C continue acting as repeaters, send the received dominant bit in round 2, and record 1. Now, A is informed that another bit sequence of higher priority is being sent, and starts acting as repeater, too, recording 1. This ends the transmission, with the result that all nodes are informed about the bit sequence 111.

In general, it takes up to $n_{maxHops}$ rounds until a repeater recognizes a dominant bit, and starts forwarding. So, in the extreme case, a repeater has to wait until the final round until it knows whether a dominant bit has been sent. However, as soon as a dominant bit has been received

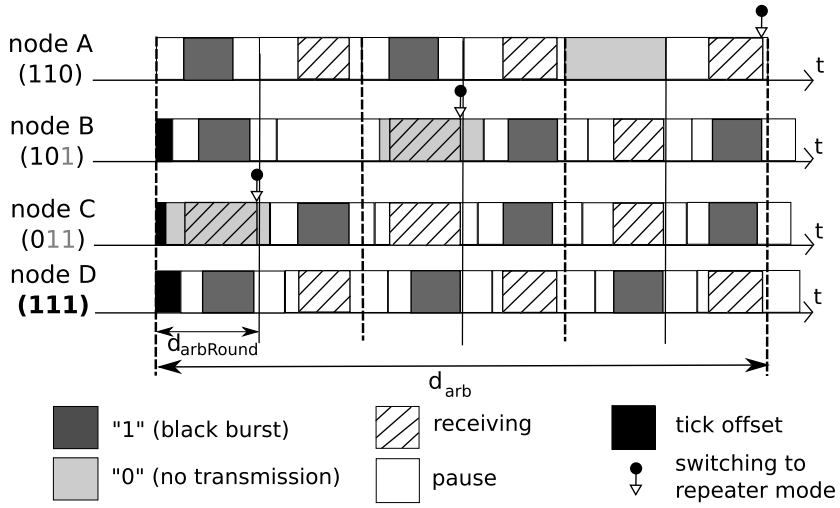

Figure 5: Arbitrating encoding. Bit sequence of node D is established, because the other nodes send recessive bits when D sends dominant bits.

and forwarded, that node may stay silent during the remaining rounds of the same bit. Note that the bit sequence with the highest value wins. If in Figure 5, node *D* would not transmit any bit sequence, all nodes would be informed about the bit sequence 110. This would also be the case if node *D* would transmit 110 itself.

Next, we analyze the duration d_{arb} for network-wide propagation of frames using arbitrating encoding. This duration is given as

$$d_{arb} = n \cdot (n_{maxHops} \cdot d_{arbRound}) \tag{4}$$

where n is the length of the bit sequence, $n_{maxHops}$ is the maximum network diameter, and $d_{arbRound}$ denotes the duration of an arbitration round which is the time required to send a black burst over one hop. The latter can be calculated as

$$d_{arbRound} = d_{coopBurst} + d_{accessRx} \tag{5}$$

4 Black Burst Clock Synchronization (BBCS)

We now introduce *Black Burst Clock Synchronization* (*BBCS*), a protocol for multi-hop time synchronization in wireless ad-hoc networks. BBCS offers master-based and decentralized time synchronization, and incorporates cooperative and arbitrating encoding (see Section 3). BBCS transmits clock values that are associated with global reference ticks, which requires a preceding network-wide tick synchronization. Below, we assume a UNIX time format with $k = 32\,bit$, however, other time formats can be used as well. The time value is placed in a time frame consisting of leading SOF bit, time value of length k, and checksum of length m (see Figure 6).

Figure 7 illustrates the conceptual foundations of BBCS. Based on tick synchronization, the medium is decomposed into macro slots starting at a real tick. This real tick is perceived locally

with an accuracy bounded by the maximum tick offset. At the beginning of each macro slot, a resynchronization takes place in a special sync slot. This phase is handled by *BBS*. In the remaining part of the macro slot, a micro slot known to all network nodes (see Section 3) marks the beginning of the transmission of the time frame. In the figure, this is the first micro slot after the sync slot, however, other placements or even the decomposition of the time frame transmission and distribution onto several macro slots (to keep intervals that are blocked for other transmissions small) are possible. After receiving the time frame, all nodes have the same time value for the last global reference tick of some node A, i.e. $c_A(t_{refTick})$, which they associate with their previous local tick. To correct their clocks, all they need to do is to compute

$$c_B(t) := c_A(t_{refTick}) + (c_B(t) - c_B(t_{localTick})) \tag{6}$$

where $c_B(t)$ is local clock reading of node B when this computation is done, and $c_B(t_{localTick})$ is the clock reading at the previous local tick.

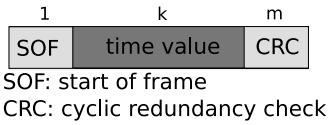

Figure 6: Time frame format.

Once a time frame has been exchanged, the time value can be used in subsequent macro slots to resynchronize the clocks, based on tick resynchronization. This is shown for the subsequent macro slot in Figure 7, and keeps clock offset within the bounds of tick offset. Thus, after an initial time synchronization, it is sufficient to exchange time values from time to time, e.g., when further nodes have joined the network, or with a fixed (long) period. Nodes detecting bit errors in a received time frame wait until the next time value exchange.

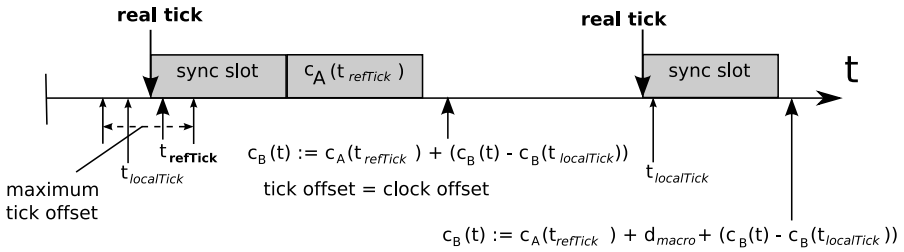

Figure 7: Foundations of BBCS.

As already mentioned, BBCS supports master-based and decentralized time synchronization. In the master-based case, some master node initiates transmission of the time frame. For network-wide propagation, cooperative encoding (see Section 3) is used. In the decentralized case, all network nodes start transmitting the time value of their local tick in a predefined micro slot. For network-wide propagation, arbitrating encoding is used. This ensures that the node with the fastest clock wins, and that all nodes are informed of the same clock value.

Based on the analysis in Section 3 and for given hardware platforms, we can now determine concrete values for maximum clock offset, time synchronization delay, and time synchronization overhead. Table 1 lists configuration and hardware parameters for the AT86RF230 transceiver [Atm], which can be found on MICAz motes. The per-hop value for $d_{maxBaseOffset}$ is derived from the the maximum CCA delay of that transceiver. We assume a UNIX time format with $k = 32\,bit$ and a checksum of $m = 4\,bit$, resulting in a payload of $n = 32 + 4 = 36\,bit$ (see Figure 6).

Variable	Value	Variable	Value	Variable	Value
b	5 byte	d_{pause}	16 μs	$d_{processing}$	300 μs
r	250 kBit/s	$d_{maxBaseOffset}$	16 μs	$d_{accessTx}$	192 μs
$r_{maxClockSkew}$	40 ppm	$d_{macroSlot}$	1 s	$d_{accessRx}$	320 μs

Table 1: Configuration and hardware parameters with AT86RF230 transceiver [Atm].

Synchronization delay for master-based time synchronization is computed by inserting values into Equations 1, 2 and 3:

$$d_{coopBurst} = \frac{5\,byte}{250\,kBit/s} + 192\,\mu s + 16\,\mu s + 2 \cdot 16\,\mu s + 80\,\mu s = 480\,\mu s \quad (7)$$

$$d_{coopRound} = 37 \cdot 480\,\mu s + 300\,\mu s = 18.060\,ms \quad (8)$$

$$d_{coop_{AT86RF230}} = 4 \cdot 18.060\,ms = 72.240\,ms \quad (9)$$

Assuming that time frames are sent with a period of 100 macro slots of length 1s, this adds an overhead o_{coop} of 0.072% on top of tick synchronization.

Synchronization delay for decentralized time synchronization is computed by inserting values into Equations 4 and 5:

$$d_{arbRound} = 480\,\mu s + 320\,\mu s = 800\,\mu s \quad (10)$$

$$d_{arb} = 36 \cdot 4 \cdot 800\,\mu s = 115.2\,ms \quad (11)$$

Again assuming that time frames are sent with a period of 100 macro slots of length 1s, this adds an overhead o_{arb} of 0.115% on top of tick synchronization.

5 Related Work

In this section, we survey selected approaches to network-wide time synchronization in ad-hoc networks. A comparative assessment of these time synchronization protocols is shown in Table 2. A thorough survey and comparison of time synchronization protocols can be found in [SBK05].

Reference Broadcast Synchronization (RBS) [EGE02] exploits the property of broadcast media that nodes within single-hop distance of a sender receive a MAC layer frame at approximately the same time. By recording the reception times of reference beacon frames and exchanging their observations, receivers can compute their mutual clock offsets. Multi-hop synchronization is achieved using time routing through clock conversion. RBS requires static configuration or dynamic (re-)election of sender nodes with network coverage. To synchronize them, too, redundant sender nodes and a sufficiently dense network topology are needed.

The Timing-Sync Protocol for Sensor Networks (TPSN) [GKS03] performs iterative sender-receiver synchronization w.r.t. one reference node, using late time-stamping. The protocol works in two phases. In the level discovery phase, reference node election and establishment of a hierarchical network structure take place. In the (re-)synchronization phase, all nodes along this hierarchy perform a pair-wise synchronization by handshakes. The use of handshakes provides some protection against frame collisions and node failures.

	Accuracy (precision/hop; deterministic bound)	Communication complexity; structural complexity	Convergence delay (deterministic bound)	Robustness against topology changes	Experiments (MAC layer; network diameter)
RBS	very high ($6\,\mu$s ; no)	$O(n)$ to $O(n^2)$ sender coverage	high (no)	low	802.11; 4 hops
TPSN	high ($17\,\mu$s ; no)	$O(n)$ node hierarchy	high (no)	low	802.15.4; 5 hops
TSP	medium (>1ms; yes)	$O(n)$ node hierarchy	high (no)	low	802.11b; 5 hops
TDP	medium (>1ms; no)	$O(n)$ elections per round	high (no)	high	-
Syncob	very high ($4\,\mu$s ; no)	$O(n)$ -	low (no)	high	proprietary; 1 hop
BitMAC	high ($20\,\mu$s ; yes)	$O(d)$ designated master	low (yes)	low	proprietary; 2 hops
BBCS	high ($16\,\mu$s ; yes)	$O(d)$ -	low (yes)	high	802.15.4; 4 hops

Table 2: Comparison of time synchronization protocols.

The Tiny-Sync Protocol (TSP) [SV03] uses probe handshakes that are time-stamped at each send/receive point, yielding a data point. Based on collected data points, two nodes can estimate their clock offset and their clock skew within deterministic bounds. For network-wide synchronization, a (logically) hierarchical topology that determines all pairs of nodes to be synchronized has to be established.

The Time-Diffusion Protocol (TDP) [SA05] is a collection of several protocols that together achieve network-wide time synchronization in mobile ad-hoc networks with good accuracy. To deal with clock skew, TDP resynchronizes all network nodes periodically. Each period starts with the re-election of master nodes, which then repeatedly diffuse timing information that is forwarded by diffused-leader nodes. Since master nodes are reelected in each period, TDP is robust against topology changes, at the expense of substantial structural overhead.

BitMAC [RR05] uses synchronized on/off keying to achieve collision-protected transmissions. The main focus here is on medium arbitration; time synchronization is only sketched. Beacon broadcasts by a designated master node are used to synchronize all receivers within range of the sender. In subsequent rounds, the beacon is propagated by receivers simultaneously, until network-wide time synchronization is achieved. The protocol is robust against node movements, but not protected against master node failure.

Syncob [KBDR07] proposes a collaborative time synchronization scheme, exploiting occasional collision-protected transmissions of sync-symbols. As BitMAC, Syncob relies on synchronized on/off modulation for collision-protected transmissions. A problem seems to be the non-deterministic nature of sync-symbol transmissions, which may lead to situations where synchronization is lost despite stable network topology.

We notice that depending on the application context, each of the above protocols has its particular strengths (see Table 2). As it turns out, all protocols yield an accuracy that is sufficient for data fusion in many realistic scenarios. *BBCS* and BitMAC achieve a worst case precision

per hop, while all other protocols yield an average precision, which is adequate for data fusion in general. *BBCS*, BitMAC, and Syncob have low convergence delay and communication complexity, and are robust against topology changes.

A drawback of BitMAC is that it uses a centralized algorithm, relying on a designated master node. This may be adequate in sensor networks with a single sink, but is a disadvantage in ad-hoc networks in general. A drawback of Syncob is the non-deterministic nature of sync-symbol transmissions, which may lead to situations where synchronization is lost despite stable network topology. Both BitMAC and Syncob use synchronized on/off modulation for collision-protected transmissions. This technique has the advantage that it supports very accurate timing. On the other hand, it is vulnerable against small timing errors, e.g., due to oversized loops [KBDR07], and is only rarely supported by transceivers.

6 Conclusions and future work

In this paper, we have presented *Black Burst Clock Synchronization* (*BBCS*), a protocol for multi-hop time synchronization in wireless ad-hoc networks. BBCS achieves time synchronization with low and deterministic maximum clock offset and convergence delay at low cost. As foundation for BBCS, we have introduced two encoding schemes called *cooperative and arbitrating encoding*. BBCS is based on tick synchronization, as provided by *BBS* [GK08], and in addition propagates clock values of real ticks among all network nodes. We have formally specified *BBCS* with SDL, and have implemented a subset on MICAz motes.

As compared to other protocols for time synchronization in wireless ad-hoc networks, the features of *BBCS* are unique. The collision-resistant encoding enables clock synchronization with deterministic convergence delay, and with a deterministic upper bound for clock offset that depends linearly on maximum network diameter and clear channel assessment jitter only. Number of nodes, node mobility, and network topology (apart from maximum network diameter) have no impact on the performance and complexity of *BBCS*.

Future work includes the complete implementation of cooperative and arbitrating encoding on the MICAz and Imote2 motes with CC2420 and AT86RF230 wireless transceivers to perform measurements under real world conditions. This has already been achieved for tick synchronization with *BBS*, on top of which *BBCS* is built. Additionally, there is ongoing research to lower the overhead of both cooperative and arbitrating encoding, while retaining the advantage of collision-resistant transmissions.

Bibliography

[ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. Wireless sensor networks: a survey. *Computer Networks* 38(4):393–422, 2002.

[Atm] Atmel. AT86RF230 data sheet. http://www.atmel.com/dyn/resources/prod_documents/doc5131.pdf

[BGK07] P. Becker, R. Gotzhein, T. Kuhn. MacZ - A Quality-of-Service MAC Layer for Ad-hoc Networks. In *HIS*. Pp. 277–282. Proceedings of the 7th Conference on Hybrid Intelligent Systems (HIS), Kaiserslautern, Germany, 2007.

[Cro] Crossbow Technology Inc. MicaZ data sheet. http://xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf

[EGE02] J. Elson, L. Girod, D. Estrin. Fine-Grained Network Time Synchronization Using Reference Broadcasts. In *OSDI*. Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, USA, 2002.

[GK08] R. Gotzhein, T. Kuhn. Decentralized Tick Synchronization for Multi-hop Medium Slotting in Wireless Ad Hoc Networks using Black Bursts. Proceedings of the 5th Annual IEEE ComSoc Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON 2008), San Francisco, USA, June 16-20 2008.

[GKS03] S. Ganeriwal, R. Kumar, M. B. Srivastava. Timing-sync protocol for sensor networks. In Akyildiz et al. (eds.), *SenSys*. Pp. 138–149. ACM, Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys 2003, Los Angeles, California, USA, November 5-7, 2003.

[ITU07] ITU-T Recommendation Z.100 (11/2007). *Specification and description language (SDL)*. International Telecommunication Union (ITU), 2007.

[KBDR07] A. Krohn, M. Beigl, C. Decker, T. Riedel. Syncob: Collaborative Time Synchronization in Wireless Sensor Networks. Fourth International Conference of Networked Sensing Systems, June 2007.

[KI07] T. Kuhn, J. I. de Irigon. An experimental evaluation of black burst transmissions. In Zomaya and Zeadally (eds.), *MOBIWAC*. Pp. 163–167. Proceedings of the Fifth ACM International Workshop on Mobility Management & Wireless Access, MOBIWAC 2007, Chania, Crete Island, Greece, October 22, 2007, 2007.

[RR05] M. Ringwald, K. Rmer. A Deterministic, Collision-Free, and Robust MAC Protocol for Sensor Networks. Proceedings of the Second European Workshop on Wireless Sensor Networks, February 2005.

[SA05] W. Su, I. F. Akyildiz. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Trans. Netw.* 13(2):384–397, 2005.

[SBK05] B. Sundararaman, U. Buy, A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks* 3(3):281–323, May 2005.

[SV03] M. L. Sichitiu, C. Veerarittiphan. Simple, Accurate Time Synchronization for Wireless Networks. Pp. 1266–1273. Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2003), New Orleans, LA, USA, 2003.

[YHE02] W. Ye, J. S. Heidemann, D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *INFOCOM*. Pp. 1567–1576. 2002.