Syncob: Collaborative Time Synchronization in Wireless Sensor Networks

Albert Krohn
Particle Computer GmbH
Germany

Michael Beigl
DUS
Universität Braunschweig
Germany

Christian Decker, Till Riedel
TecO
Universität Karlsruhe
Germany

***Abstract*— This paper analyses the problem of time-synchronization of distributed sensor nodes. We present a Syncob, a method for synchronizing an arbitrary number of nodes in a distributed setting without the requirements of an infrastructure, master node or time and resource consuming protocol overhead. Syncob is therefore also very good suited for highly mobile settings with ad-hoc communication. Syncob is implemented as a physical layer protocol and provides a time synchronization deviation of max. 4μs between any participating node. Our implementation on low-cost pPart sensor nodes shows that Syncob requires very low overhead and very low complexity for hardware and software.**

***Keywords*— time-synchronization, distributed synchronization, superimposed radio signal, collaborative protocols, cooperative transmission**

I. Introduction

Wireless Sensor Networks are an attractive way to monitor and control the physical world in an unobtrusive and ad-hoc manner. In a wireless sensor network, a potentially high number of sensor nodes are connected through wireless ad-hoc and multi-hop communication. Many new applications have been proposed using wireless sensor networks for precision agriculture, border security, geophysical monitoring and logistics. Wireless sensor networks are an excellent example for a distributed system: they consist of small, independent entities collaborating for a superior purpose.

Sensor nodes normally have an independent power supply like a battery and use methods of energy harvesting like solar cells. To control the limited energy resources efficiently, wireless sensor networks typically undergo periodic sleep-cycles to save energy. To collaborate for a common application, wireless sensor nodes have to be precisely coordinated including calibration and coordination for multi-hop communication. Much of the coordination relies on *time synchronization* among the nodes as e.g. multi-hop communication and periodic sleep times is not possible, when nodes' wake-up cycles are not synchronized. Monitoring of physical events often also requires marking sensor values with according time stamps of a global clock. Besides these two examples, there are many other examples for the necessity of time-synchronization in wireless sensor network, which is widely agreed to be a central building brick for these systems [1], [2].

In this paper, our target application platform is the low-end area of sensor networks. The methods we propose should work on simple sensor nodes and do not require sophisticated signal processing or handling of large amount of data. Before we present a solution we will analyze which network layer is suitable for such a time synchronization method. We contribute with a solution for time-synchronization on the physical layer that can synchronize groups of sensor nodes with high accuracy, low overhead and outstanding speed. The time-synchronization we present is a method for establishing a common time-frame among participating nodes and to control the local offsets against this global time-frame. Based on this synchronization, some applications are discussed.

A. Levels of time synchronization

Generally, time synchronization can be implemented on any layer of the network stack introducing various difficulties. When exchanging signals or packets between synchronization partners, each layer that is traversed will introduce uncertainties for the synchronization. Figure 1 shows this. When exchanging synchronization symbols

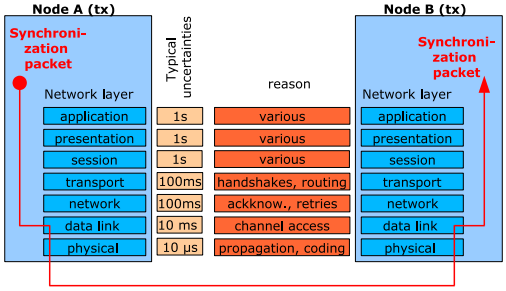

Fig. 1. Timings for deviation according to network layer

only on the physical layer, then the only influence on the synchronization is the coding, signal generation and propagation time on the medium. In most cases this is very small compared to the required accuracy.

As soon as the synchronization is implemented on

higher layers, the synchronization suffers from more uncertainties due to time delays added by the underlying layers that may not be known or predictable. We will now shortly discuss these delays layer by layer. In the *link layer*, packet delays due to transmission or reception will be introduced. The total amount of time added on the packets then depends heavily on the link layer protocols. If the channel access is e.g. a CSMA with random back off time, it will be hard to retrieve what the actual delay was from a higher layer. In case the channel access is simple transmission ("fire and forget"), the delay will be more deterministic, but the reliability of the synchronization suffers. In general a accuracy of about 10ms is to be expected. At the *network layer* packet processes like queuing and routing will further introduce unpredictable time delays. Additionally, as packets may come from several hops away, the transmission time may have serveral effects on the overall delay which is mostly unpredictable because most routing algorithms choose the route transparently. The *transport layer* adds buffering, transparent resending and other transport control mechanism to the stack. Delay from these mechanisms are only partially controlled by the own node thus adding to slightly higher delay uncertainty. All *application near layers* contain various application specific functionalities that provide high-level abstract functionality. Delays introduced here are typically in the area of seconds.

From this perspective, it seems advantageous to implement synchronization methods as low a possible in the network layers. We now want to quantify the expected accuracies for implementations on different layers as a rule of thumb. On the physical layer, only the signal propagation and processing will influence the accuracy, which will therefore result in accuracies in the μs area.[1].

II. Application scenarios and requirement analysis

In this section, we want to analyze some concrete application scenarios to motivate time synchronization for practical usage in wireless sensor networks. Based on the scenarios, we also define the required distributed operation parameters such as speed and accuracy for time synchronization.

A. Sensor value correlation

Sensor fusion has found a lot of attention in the research on sensor networks. In many cases, the low-level combination of sensor values and measurements series is helpful or necessary when monitoring events in the physical world. It is often necessary to fuse sensor values that occur e.g. at different places but at the same time. We want to take the "Smart-its friends"[3] as an example for time synchronization:

Two sensor nodes locally measure their acceleration values and record them in a sliding window. They compare their last series of values with each other, and when they recognize a similar movement pattern, they assume to be attached to the same object and thus declare themselves as "friends".

Similar ideas have also been used for more advanced user interaction and recognition applications in [4] and [5]. To compare movements patterns between objects, the series of acceleration values must be highly time-synchronized; the tolerable derivations are typically smaller then 100 ms (in [5] typically 50 ms).

B. Location systems

Location information is one of the most interesting attribute to sensor values. In this paper, we want to take a closer look at ultrasound indoor positioning systems with high accuracy which in many cases use time synchronization as a system basis. Besides early research prototypes and systems such as the Active Bat system [6], Cricket System [7], Dolphin [8] and RELATE [9], [10], there are many commercial systems available. The time synchronization necessary for ultrasound location systems is in the area of $10\mu s$[2]. For a location system based on ultrasound (or even audible signals) the time synchronization accuracy should be in the area of μs. Referring back to figure 1, it is clear that the synchronization must be implemented on the physical layer to achieve such high accuracies.

III. Distributed Signaling

The time synchronization mechanism presented in this paper is based on the idea of distributed signaling. We do not consider a central instance or master node that serves as a time reference beacon. Instead, we understand synchronization as a general task supported by each participant in the network. We propose a fully distributed mechanism based on random beacon transmission. The time in our proposal is divided into frames to form a TDMA system. At the beginning of each frame there is a beacon transmission to re-synchronize nodes in the network. A beacon contains a reference point in the time, such that any node receiving a time-synchronization beacon can re-synchronize its local clock according to the received reference point in time. In our system, every synchronized node will send out beacons to re-synchronize its neighbors. During idle times, every node is requested to also listen to beacons of other nodes to re-synchronize its own local clock. Using distributed operation, we run into the problem of coordination. If we do not establish a certain controlled coordination, it is unclear which node would transmit re-synchronization beacon at what times or in which frames. In figure 2 we can see that

[1]calculation example: at a distance of 100m, the radio propagation time is only around $0.3\mu s$

[2]Calculation example: with 10 μs time accuracy, one can expect a minimum location error of: $x = v \cdot t = 340\frac{m}{s} \cdot 10\mu s = 3mm$

the random operation leads to collisions. These collisions could be taken care of by introducing random accesses for beaconing processing in a CSMA or TDMA fashion. The distributed random operation introduces even more problems known from wireless networks: hidden and exposed terminals can severely affect the synchronization quality, such that nodes could often miss resynchronization events and therefore loose system connection.

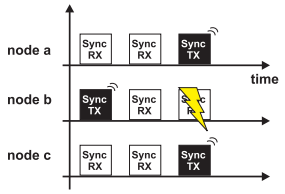

Fig. 2. Distributed beaconing process without sophisticated control mechanisms: collisions can occurr

To overcome the problem of coordination of beaconing, hidden and exposed terminals and other obstacles in the protocol design, we want to implement a system operation as described in figure 3. Nodes transmitting at the same time do not cause a destructive collision. The channel access for the beaconing process does not need to be controlled by CSMA-style methods and simultaneous beaconing does not harm the system. There are several ways how such a

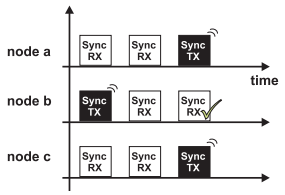

Fig. 3. Distributed beaconing process without collisions

system could be established. A classical approach would be to use multiple access schemes known from radio communications. As the nodes are located nearby and transmit during the same times[3] the only choices left are FDMA and CDMA systems. The network would be partitioned into clusters and different frequencies or codes would be used to avoid colliding transmissions. Such an approach leads to follow-up problems: how to choose the correct codes or frequencies to avoid overlapping? These questions result in the graph coloring problem and are known to be only efficiently solved for static topologies. In mobile scenarios the coordination overhead explodes.

We shortly motivated that it is not trivial to coordinate the distributed beaconing process using classical approaches such as FDMA, CDMA or TDMA (with e.g. token ring or CSMA access). We therefore like to propose a radical new approach: A system, where simultaneous transmissions on the same frequency band do not result in a collision. Two stations sending a normal narrowband modulated synchronization-packet at the same time will *collaborate* in such a way so that other stations will still receive the message. Figure 4 shows an architectural view of such a process. The key to the operation are

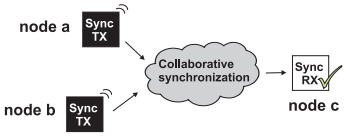

Fig. 4. Collaborative mechanisms on the network layers enable the collision-free, simultaneous synchronization

collaborative aspects on the network layers, especially *superimposed radio signals*. In section V, we explain in details how such an operation can be realized.

IV. Analysis of Related Work

We like to review some relevant protocols in the sensor network area that address time-synchronization for wireless sensor network and evaluate them against the assumptions and requirements we derived in our analysis in section II. Important aspect are accuracy, speed, overhead and usability for our target scenarios.

A. LTS

The "Lightweight Time Synchronization "-protocol [11],uses pair-wise synchronization, which is then expanded into multi-hop networks using tree-structures. For the pair-wise synchronization, a packet is exchanged and the transmit and receive times are locally stored. After the circulation of the packet between the partners, the unknown transmission time can be subtracted and the clocks can be re-synchronized. This method works on a fairly high level of the networking layers and can only produce good results, when the unknown delay due to transmission is (according to [11]): a sum of transmission delay, signal propagation on the channel, reception delay and channel access delay) and protocol implementation is identical for all nodes. In theory, the authors expect to achieve a time synchronization accuracy of around 100 μs with a certainty of 99% for typical cases. In simulations they achieve a accuracies at about 100ms. Both values are too large for our envisioned target scenarios.

B. RBS

Reference broadcast synchronization [12] is similar to the synchronization methods proposed in this paper. It is based on the exchange of synchronization packets on the physical layer and achieves typical accuracies better than $5\mu s$. The RBS system is receiver centric, such that if two or more nodes receive the identical sync-packet from a remote node, they would compare their local reception times among each other in a following communication process and then correct their clocks to a common view.

[3]at least for the case of a collision; otherwise there won't be a collision

The drawback of RBS is that the transmitted synchronization packet does not contain an absolute time reference of the transmitting node. Only the relative times between receivers are corrected. The authors further propose an extension to multi-hop operation that included many nodes and various topologies. The authors do not discuss optimal coordination for the pervading of synchronization which – especially for dense settings – will introduce relevant overhead.

C. Mini-sync und Tiny-sync

The synchronization methods *tiny-sync* and *mini-sync* are mainly dealing with the drift of local oscillators, which is caused by hardware variations due to production or environmental influences such as temperature differences. The authors model the correlation between a local time t_i and the reference time t as linear:

$$t_i(t) = a_i t + b_i \qquad [13, formula(1)] \tag{1}$$

now, the relation between two local clock can be formulated as:

$$t_1(t) = a_{12} t_2(t) + b_{12} \qquad [13, formula(2)] \tag{2}$$

If now node 1 and node 2 want to correct their local clocks, a packet is exchanges from nodes 1 to 2 and back to 1. As the causal order of the events is clearly determined ($t_0 < t_b < t_r$), one can assemble inequations to find the upper and lower bounds for the unknown a_{ij} and b_{ij}. The more packets are exchanged, the more converges the estimation. The difference between *mini-sync* and *tiny-sync* is that tiny-sync always uses only the minimum number of times to determine the bounds whereas mini-sync memorized more than the necessary number of values. The authors show in simulation that typically 40 of such time values are enough to achieve 3ms of synchronization for single-hop and 30ms for multi-hop networks. For the envisioned scenarios and use-cases of this paper, these accuracies will not be sufficient.

D. BitMAC

The BitMAC [14] Protocol proposes the use of a collision-free system for time-synchronization. We extend the time-synchronization systems of BitMAC, with focus on distributed operation and issues on the physical layer. BitMAC is based on distributed bit transmission from different sources. BitMAC claims, that preambles and other packet parts can be left out. The use of BitMAC for synchronization is as follows: Nodes simultaneously transmit a certain bit-sequence. Doing so, collision can be avoided, as BitMAC assumes that the communication channel has an "OR"-characteristic. This is illustrated in figure 5. Similar assumption on the physical layer can also be found in [15]. However, the nature of radio waves imposes problems with this assumption. In this paper, we want to clarify the problems and contribute with a solution. Secondly, BitMAC uses a centralized architecture where the synchronization is always initiated by a single, central node. We instead present a fully distributed process without centralized coordination. BitMAC also assumes a natural averaging through the simultaneous transmission. Several authors ([16], [17], [18], [19]) have looked into this subject and it will also be a subject in our paper.

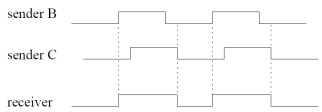

Fig. 5. "Identical transmissions by two senders with small synchronization errors. The receiver will see slightly stretched "1"bits and slightly compressed "0"bits ". From [14, Figure 5, page 6]

V. Syncob: Collaborative Time Synchronization

Our proposed collaborative synchronization *Syncob*[4] depends on various collaborative aspects. We will now separate them by the network layers and explain in details what mechanisms lead to the radial new approach.

A. collaborative physical layer

Referencing back to figure 4, we now want to explain how a collision-free synchronous transmission in the same frequency band can be realized. In figure 6, node a and node b send a synchronization packet at the same time. In this simple example, the synchronization corresponds to the emission of a synchronization packet; a symbol plus a preamble which are modulated binary. The binary sequence of the synchronization packet is in this case "10101010101011001". In the example, the binary modulation is mapped to the modulation symbols S_0 and S_1 which are then transmitted as $S_1S_0S_1S_0S_1S_0S_1S_0S_1S_0S_1S_0S_1S_1S_0S_0S_1$ to the channel. Channel and source coding mechanisms like whitening of data stream, manchester encoding, bit-stuffing etc. have to be disabled. With the collaborative concept of our mechanism, the two simultaneous emitted packets can superimpose on the wireless medium and can still be received by other nodes. This – of course – implies certain restriction on physical layer and also on the link layer of the used radio communication. But the effect of our proposal is immediately clear: the distributed beaconing process will be supported by the collaborative aspects of the network layers such that a coordination of beacons is not necessary. Whether one or more nodes transmit synchronization packets at the same time does not matter to the system. Receiving nodes can still receive the packet

[4]the name syncob is inspired by the *syncope* in music theory. A syncope is a stressing of a normally unstressed beat to produce a certain rhythm. In the same way, our protocol uses stressing of certain symbol transmission to produce a certain rhythm: the time synchronization for a time-framed protocol

that was emitted by one, two or more stations. The organization of the beaconing is therefore free of protocol overhead and can be based on local random processes independent of the status of partner nodes.

Collaborative aspects on the physical layer are a known and established field in the research on wireless and wired communication. It can normally be found under the term *cooperative transmission*. For our case, we are interested in a very special case of cooperative transmission where two or more stations use narrow band signals that are simultaneously transmitted and can constructively superimpose in the radio channel. Similar mechanisms have been proposed for ultra wide-band operation [16] and also for certain FSK systems [17], but not for narrow band systems. We want to take a closer look at those mechanisms from the application and implementation perspective. We need to design a radio front-end, that can transmit and receive modulation symbols (for our example only binary modulation with the two symbols S_0 and S_1) in a cooperative way, such that multiple, synchronous transmitted symbols on the channel will not cause a non-deterministic collision. At the same time, we need to respect the strong restrictions that come along with our target platform, the low-cost sensor nodes. Sophisticated phase manipulations or real-time filtering and generating the radio waves in the time domain with software radio techniques will not be possible. The typical radio front-end of sensor nodes leaves very little design space for cooperative transmission; the modulation will have to be *narrowband* and a *FSK, ASK or OOK* and will also have neither *phase-tuning of the carrier* nor *software radio function*. Therefore, we decided to use a narrowband binary OOK/ASK to realize the superimposed radio signals. With OOK, S_1 corresponds to an active transmission of a carrier and S_0 corresponds to no transmission. But when receiving two or more carriers/signals from different sources, the received signal will be the *sum* of all signals. Such a summation can lead to constructive and destructive interference. The latter would be crucial for the system, as the signals would not be detected correctly by the receiver. To understand this problem, we have to go on an excursion of signal processing theory on the physical layer of radio communication:

Excursion: superimposed radio signals When two or more radio signals (in a simple case only the pure carrier) are transmitted simultaneously, they will sum up on the channel. When the carrier frequencies and phases of the emitted waves are not perfectly aligned and synchronized, the superposition will cause alternating destructive and constructive interference. These interferences will vary in time and space. Figure 6 shows an extreme example of a resulting received signal as a superposition of two incoming signals from different sources. The carrier frequency are slightly different causing a slowly interference pattern in the receiver. Only when monitoring the received signal for more than a period of the frequency of the envelope modulation, a robust

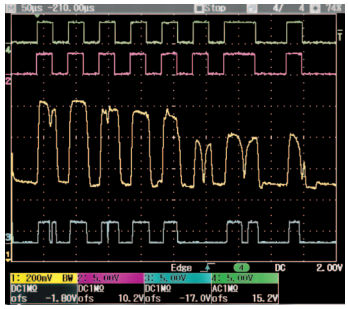

Fig. 6. Superimposed radio signals can result in destructive interference. The time-signal top down: Two base band TX amplitudes of two OOK transmitters, the received base band amplitude (RSSI) signal as a sum of the transmitted signal, the bit-decision output of the transceiver.

detection can be guaranteed. As the symbol duration of S_1, S_0 is typically only some $10\mu s$ and the envelope period can be as high as several $10ms$, the system is not well designed. The observation time (duration of Symbols) must be longer than the period of the observed signal (period of the incoming signal)!

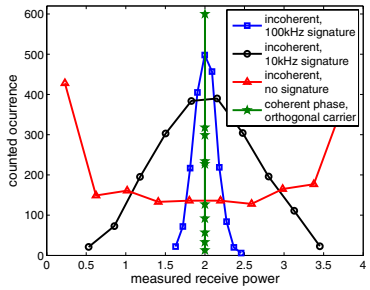

Fig. 7. Applying signatures to the emitted signal improves the short-time statistic of the received amplitude of superimposed radio waves

To solve this, we propose to modify the emitted signal by spreading the spectrum with a signature. With this, we can achieve that the period of the received signal envelope is shortened. We simulated several signatures as band-limited noise and applied them to the transmitted signal. Figure 7 shows the pdfs of the received signal power. We apply two signals with power one and superimposed them in the channel – we would then in the receiver expect to detect a power of two. Without signatures and with incoherent carriers (which will be the case for our scenario of simultaneous emitted symbols), one can see that the pdf (denoted with Δ) is fairly unusable. We get a very broad distribution with even high differential probabilities at extremes. Then, with increasing bandwidth of the signature, we approach the ideal case of coherent phases. The detection is improved and can form the basis of our collaborative physical layer.

As the time-synchronization is not perfect, the start times of the signals will not be perfectly aligned. In figure 8 we show an extreme example of such an offset Δt. With

these time-shifts between the transmitters, the received signal will get fuzzy edges and the times, where waves are emitted (during S_1, whereas during S_0 there is no emission) will be dominant. Figure 6 is a good practical proof for the assumption in [14] and [15] that the channel will have and "or"-behavior, if we modulate with OOK like described above. The higher the time-shift Δt between the nodes is, the harder will it be for the receiver to guarantee correct detection as the signal borders and signal durations will get more and more fuzzy. We will discuss the theoretical limit in section V-D. For further discussions on signal formation and detection, we refer the interested reader elsewhere: We showed the effect of superimposed radio signals already in an implementation in [19] and [20] and also derived a new modulation system for collaborative physical layers in [21]. For the detection of the simultaneous signals, a Maximum-Likelihood decision on an energy-detector can be used.

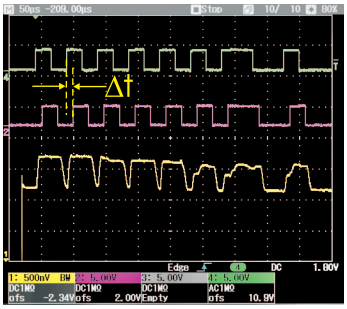

Fig. 8. Time-shifts can harm the collaborative physical layer. The time- signals from top-down: Two base band amplitudes of transmitters with a relevant time-offset Δt; the received base-band amplitude (RSSI)

B. collaborative link layer

Our proposal of a link layer includes all other aspects of our Sycob protocol. We will now separate them and explain collaborative aspects for each of the listed topics:

Building up a synchronization. Each node is equally responsible for establishing synchronization. Nodes that are left alone actively start synchronizations and actively listen for existing synchronizations. Whenever they find a remote synchronization, they accept the synchronization and collaborate with the remote station. Or they wait for other nodes accepting their own synchronization

Stabilizing a synchronization. Each node in a synchronized network is responsible for keeping up the synchronization. There are no master nodes nor cluster heads or any other preferred nodes taking care of this important task. It lies in the responsibility of each node to accept, re-synchronize to incoming synchronizations packet and promote synchronization and re-synchronization to other. With the effect of smoothing and averaging on the physical layer (see section V-A) and the collision free emission and reception, the distributed operation is stable an efficient.

Automatic rate control. Each node locally decides on its own synchronization transmit rate. When many nodes are around, the number of necessary transmits per time and node is reduced. Nodes control this behavior locally and such collaboratively find a stable solution for the distributed operation.

Distributed operation. The nodes have no additional communication channel to organize and control the synchronization process. All active exchange is the emission and reception of the explained synchronization sequence. The process of establishing and stabilizing of the synchronization is free of hierarchies or pre-defined individual parameterization. Each node has the exact same starting point and the same rights and responsibilities.

Through these distributed and collaborative parts of the system, the nodes can build up and stabilize the synchronization efficiently. Figure 9 shows a simplified flow chart of the synchronization state machine in the nodes.

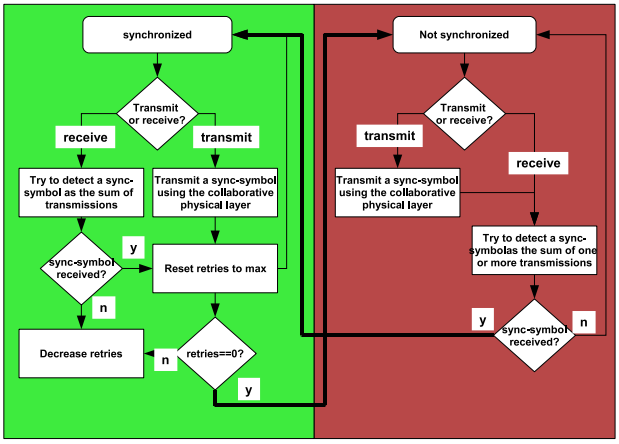

Fig. 9. The flow chart of the collaborative synchronization

C. collaborative network layer and topological issues

The Syncob protocol naturally extends to multi-hop scenarios such as the distributed operation is not limited to a fully connected topology. As long as sensor nodes detect and transmit synchronization symbols, the time-synchronization is stabilized. Nevertheless, there are certain limitations that we consider as future work for syncob:

Oversized loops. When a propagating time-frame returns to the originator over a very long path, the error propagation can then lead to a time-shift that is not acceptable for the collaborative physical layer. Then, two transmissions of neighbors would lead to a collision. There are various possibilities to deal with this effect. One way is to extent the bit-timings of the collaborative physical layer to accept the maximum

occurring time-shift. For this, assumptions on the maximum physical and topological extension of the network must be taken.

Concurrent islands. This situation occurs when two independent networks come into reach range. The nodes in the overlapping area then have a membership conflict. In S-MAC [22], this situation is solved by letting these nodes adopt to both time-frames, another option is to switch to different channels. Generally such situations are characterized by three fundamental processes: *conflict detection*, *conflict information propagation* and *conflict resolution*. The most interesting point here is the resolution of the situation of two synchronization schemes. Syncob aims at a common synchronization for all nodes and therefore multi-frequency approaches are not feasible. The problem of concurrent islands can finally be reduced to the common problem of a preference or leader election in distributed systems. One synchronization-scheme must be superior to the other. A classical approach to leader election is the use of a unique identifier. Then, a feasible mechanism must be found to propagate the preference or leader throughout the complete network.

D. Time synchronization considerations

As Syncob is based on distributed, synchronous transmitting of equal symbols, it is important to determine the tolerable offset between nodes with which the system can still work. When the synchronization offset between nodes is too large, then the synchronous transmission will no longer work, as the symbols cannot constructively superimpose but will randomly collide.

We are now looking at the trade-off between Quartz accuracy and resynchronization. Assuming e.g. two nodes that are initially aligned with an offset of $t_{\Delta init}$, a maximal tolerable time-offset t_{tol} and a Quartz accuracy of $k[ppm]$ on a nominal oscillation frequency f_0 and period T_0, then we can calculate the necessary period of a synchronization t_{resnyc} for a worst case scenario of two nodes with contrarily extreme Quartz inaccuracies. The difference between the oscillation periods is

$$\Delta T = T_0(\frac{1}{1-k} - \frac{1}{1+k}) \tag{3}$$

Then we find the trade-off between Quarz accuracy and re-synchronization time t_p as:

$$t_{tol} = t_{\Delta init} + t_{resync}(\frac{2k}{1-k^2}) \approx t_{\Delta init} + t_p \cdot 2k \tag{4}$$

VI. Implementation

Syncob was implemented on pPart particle computer sensor nodes[5] which is used in many applications in different labs.. There, it forms the basis of collaborative time synchronization for the AwareCon Protocol [23]. AwareCon is a TDMA protocol with a time frame of 13ms. In the beginning of each 13ms frame, Syncob is used to re-synchronize the network. This requires 4% of the frame time. With a symbol length (S_0 and S_1) of 24 μs, Syncob guarantees a time synchronization accuracy of $4\mu s$ among all nodes. The pParts carry a 10ppm Quarz and can synchronize to an existing received symbol with the accuracy of one instruction ($0.2\mu s$). So looking back to (4) with the requirement of $t_{tol} \stackrel{!}{=} 4\mu s$, we see that $t_{resync} = 4\mu s + 0.2\mu s/20 \cdot 10^{-6} = 210ms$. With the frame size of $13ms$, that means, Syncob has to be resynchronized at least every 16 time frames. For stability in the implementation, Syncob tries to resynchronize every slot and and switches the status to unsynchronized after 7 time-frames without resynchronization. Another important aspect is the total physical area that a single cell with Syncob running can cover. For this, we must extend (4) with the variable propagation delay t_{delay} on the medium:

$$t_{tol} = t_{\Delta init} + t_p \cdot 2k + t_{delay} \tag{5}$$

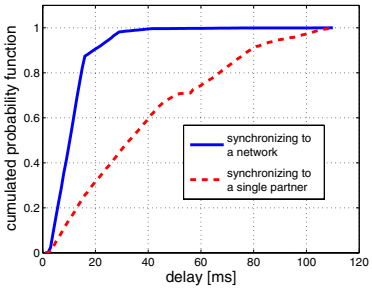

Fig. 10. Experimental results for the time needed for synchronization

Another important measure is the time needed to synchronize a node to a network. For this, we set up a test environment with particle computers and recorded the times needed for a node to enter the state "synchronized". The distribution of times is plotted in figure 10. The experiment yielded an average value of $40ms$ for two isolated nodes that come into radio range and an average of $12.6ms$ for a node that comes into range of an already established and synchronized network.

VII. Evaluation and Conclusion

With Syncob, we presented a time-synchronization protocol for wireless sensor networks based on various collaborative aspects. The major part is the collaboration on the physical layer which leads to a scalable, intrinsically stable and distributed synchronization. Syncob only required local knowledge, avoiding additional overhead through packet exchange for coordination. We evaluate the Syncob protocol using the requirement analysis from section II. Syncob can be implemented with little overhead and leads to accuracies in the area $< 10\mu s$. With the high accuracy

[5] http://www.particle-computer.de

of synchronization, Syncob is suited for the use of sensor-fusion and coordination for wireless sensor networks. It can even form the time-synchronization basis for sensible applications like location systems based on time-of-flight of ultra- or audible sound. In table I, we summarize the properties and application domains for the discussed protocols. All other proposed protocols require additional coordination packet exchanges in addition to the actual time synchronization. Under high mobility, this can lead to complicated cases, when synchronization partner lost their single-hop connection during the synchronization process. Syncob instead is ideal for the case of mobile scenarios as it naturally averages the signals of the nodes in range and locally adapts the transmission rates for an equalized load.

protocol	**accuracy, [layer]**	**sensor corr.**	**location system**	**mobile scenarios**
LTS	100 μs, [ll]	yes	no	limited
mini-sync	30 ms, [ll]	no	no	limited
RBS	5 μs, [phy]	yes	yes	limited
BitMAC	20 μs, [phy]	yes	limited	limited
Syncob	4 μs, [phy]	yes	yes	yes

TABLE I

Parametric comparison of time synchronization protocols.

Acknowledgement

The work presented in this paper was partially funded by the European Community through the projects *CoBIs* (contract no. 4270) and *RELATE* (contract no. 13790) and by the Ministry of Economic Affairs of the Netherlands through the BSIK project *Smart Surroundings* under contract no. 03060.

References

[1] J. Elson and K. Römer. Wireless sensor networks: A new regime for time synchronization. *ACM Computer Communication Review (CCR)*, 33(1):149–154, 2003.

[2] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium*, page 186, 2001.

[3] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, , and Hans-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Proceedings of the first international Conference on Ubiquitous Computing (UbiComp)*, Atlanta, USA, 2001.

[4] Jonathan Lester, Blake Hannaford, and Gaetano Borriello. äre you with me? using accelerometers to determine if two devices are carried by the same person. In *Proceedings of the 2nd international PERVASIVE Conference 2004*, pages 33–50, 2004.

[5] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Proceedings of the 2nd international PERVASIVE Conference 2004*, pages 1–17, 2004.

[6] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.

[7] Nissanka B. Priyantha, Allen K. L. Miu, Hari Balakrishnan, and Seth Teller. The Cricket Compass for context-aware mobile applications. In *Proceedings of the Seventh International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001.

[8] M. Minami, Y. Fukuju, K. Hirasawa, S. Yokoyama, M. Mizumachi, H. Morikawa, , and T. Aoyama. Dolphin: A practical approach for implementing a fully distributed indoor ultrasonic positioning system. In *Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp)*, Nottingham, England, September 2004.

[9] Mike Hazas, Christian Kray, Hans Gellersen, Henoc Agbota, Gerd Kortuem, and Albert Krohn. A relative positioning system for co-located mobile devices. In *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Seattle, USA, June 6-8 2005.

[10] Albert Krohn, Michael Beigl, Mike Hazas, Hans Gellersen, and Albrecht Schmidt. Using fine-grained infrared positioning to support the surface-based activities of mobile users. In *Proceedings of the 5th International Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, Columbus, USA, 2005.

[11] Jana van Greunen and Jan Rabaeay. Lightweight time synchronization for sensor networks. In *Second ACM International Workshop on Wireless Sensor Networks and Applications in conjunction with ACM MobiCom 2003*, San Diego, CA, USA, September 2003.

[12] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, USA, Dec 2002.

[13] Mihail L. Sichitiu and Chanchai Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, USA, March 2003.

[14] Matthias Ringwald and Kay Römer. Bitmac: A deterministic, collision-free, and robust mac protocol for sensor networks. In *roceedings of 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 57–69, Istanbul, Turkey, january 2005.

[15] Albert Krohn, Michael Beigl, and Sabin Wendhack. SDJS: Efficient statistics for wireless networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols*, Berlin, Germany, 2004.

[16] A. Scaglione and Y.-W. Hong. Opportunistic large arrays: Cooperative transmission in wireless multihop ad hoc networks to reach far distances. *IEEE Transaction on Signal Processing*, 51(8), August 2003.

[17] An swol Hu and Sergio Servetto. dfsk: Distributed frequency shift keying modulation in dense sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, Paris, France, June 2004.

[18] A.S Hu and S Servetto. On the scalability of cooperative time synchronization in pulse-connected networks. *IEEE Transactions on Information Theory*, to appear, 2006.

[19] Albert Krohn, Michael Beigl, Christian Decker, Till Riedel, and Tobias Zimmer. The implementation of non-coherent cooperative transmission for wireless sensor networks. In *Proceedings of the International Conference on Networked Sensing Systems (INSS)*, Chicago, USA, May 2006.

[20] Albert Krohn, Michael Beigl, Christian Decker, Till Riedel, Tobias Zimmer, and David Garces. Increasing connectivity in wireless sensor network using cooperative transmission. In *3rd International Conference on Networked Sensing Systems (INSS)*, Chicago, USA, May 31- June 2 2006.

[21] Albert Krohn. Optimal non-coherent m-ary energy shift keying for cooperative transmission in sensor networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France, May 14-19 2006.

[22] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks, June 2002.

[23] Michael Beigl, Albert Krohn, Tobias Zimmer, Christian Decker, and Philip Robinson. AwareCon: Situation Aware Context Communication. In *Proceedings of Ubicomp 2003*, Seattle, USA, October 2003.