Proceedings of the 2008 IEEE
International Conference on Information and Automation
June 20 -23, 2008, Zhangjiajie, China

A Precision Adaptive Average Time Synchronization Protocol in Wireless Sensor Networks

Li Li, Yongpan Liu, Huazhong Yang and Hui Wang
Department of Electronic Engineering
Tsinghua University
Beijing, 100084, P. R. China
lli03@mails.tsinghua.edu.cn,{ypliu, yanghz, wangh}@tsinghua.edu.cn

***Abstract*— Time synchronization provides an essential service for wireless sensor networks. Most of previous works focus on improving synchronization accuracy. However, other design concerns, such as global time continuity, power efficiency and adaptability, arise from real applications' requirements and receive more and more attentions. During synchronization, most of approaches choose a specific sensor node's (denoted as leader node) local time to be the global time, which would be disturbed easily due to network dynamics (leader node's re-elections or power down). The disorder global time reference could confuse other sensor nodes and even crash applications based on temporal sequence relationship. This paper first present an average based time synchronization algorithm, which can provide continuous global time in a cluster with similar synchronization precision during leader re-elections. Furthermore, in order to meet power efficiency and adaptability requirements from applications, a self measure and query based adaptive mechanism is integrated into it to adjust synchronization period dynamically. Compared with previous adaptive algorithms, the query table method provides one-step convergence while satisfying synchronization error tolerance.**

***Index Terms*— Sensor networks, Adaptive time synchronization, Power consumption**

I. Introduction

In wireless sensor networks (WSN), time synchronization (TS) is the key enabling technique and provides essential services to support applications to operate properly. For example, precise time is needed to switch neighboring sensor nodes into sleeping mode or to wake up simultaneously for communication in duty-cycling schedule protocols [1], [2]. Acoustic-aided location protocols [3] utilize timing information to measure time-of-flight of sound for distance estimation between nodes. In object tracking [4] and data fusion protocols [5], it is also used to estimate velocity and direction of objectives and to make sense of samples from different sensors. Since WSNs are mainly used to monitor in an embedded, untethered way, their main different features from traditional networks include resources limited (limited power supplies and communication bandwidths), applications specific (different WSN protocols are designed for different monitor tasks) and dynamics (topology changes frequently due to disabled nodes).

This work was sponsored in part by NSFC under Grant#90707002 and in part by 863 program under contract #2006AA01Z224.

A. Related Work

Most of previous time synchronization works focused on reducing synchronization errors, which mainly come from two aspects: uncertainty of message delays and inaccuracy of physical clocks. Ref. [6] realized their protocol using a two-way phase offset compensation method, while ref. [7], [8] and [9] estimated clock frequency through linear regression and extended set-valued estimation. Ref. [10] synchronized nodes with help of time interval transformation and bounded drift clock models. Ref. [11] measured various delays during message delivery and subtracted them from time stamps. However, many other design metrics required by applications running on WSNs are receiving more and more research attentions.

Maintaining monotonous and consistent global time in dynamic wireless sensor networks becomes a practical requirement from applications those rely on temporal sequence [4], [5], but few TS protocols have noticed this. Most TS protocols choose root's local time to be the global time [6], [8], which usually is the reference timescale of the network. It would be inconsistent if the root is disabled or is re-elected, which happens frequently in WSN [12], [13], [14]. If the discontinuous global time confuses the temporal sequence of data from two nodes, the tracking protocol might estimate a wrong direction of moving objects. Therefore, TS protocols should consider how to maintain global timing continuity.

Furthermore, due to limited resource of WSNs, power consumption becomes an important metric, so does the adaptability. In practice, various applications have to share the service provided by the same TS protocol. A non-adaptive TS protocol has to keep on achieving the best synchronization accuracy and can not switch to longer sync period for coarse precision requirement, which will waste energy. What's more, even for one certain application, the adaptive mechanism can help to find the best sync period. Given error bounds, ref. [15] used statistical methods to predict sync error and adjusted sync period multiplicatively. However, their approach was too complicated for tiny nodes' microprocessors. As a result, the authors have to port Soft-Float library on sensor nodes. Ref. [16] presented a probabilistic algorithm to adjust sync period additively. Besides computational complexity, their recursive methods suffered from slow convergence speed, which consumed more power and delayed response.

Averaging technique was used to synchronize clocks in the presence of faults in traditional distributed systems [17], [18]. For sensor networks, ref. [19] proposed a diffusion-based sync protocol by averaging neighboring nodes' time in an ad-hoc network. But the iterative averaging resulted in a extreme slow convergence speed (up to 18000 rounds for a sparse network in simulation) and was hard to be utilized in a real network. In this paper, we propose a precision adaptive average time synchronization protocol for cluster-based networks. The leader averages cluster's members' local time and broadcasts the mean value periodically to synchronize nodes in the same cluster. It decides the global time in a cluster using all nodes' time information. Furthermore, an adaptive mechanism is integrated to increase the protocol's flexibilities and power efficiency.

B. Contribution

Our main contributions are mainly listed as below:

- We propose a new TS protocol utilizes averaging and broadcasting for cluster-based networks. Compared with existed in-cluster time synchronization approaches, our averaging protocol can maintain consistent global time during leader re-elections in a dynamic network, which is particularly useful for those applications using global time stamps to compose data in temporal sequences. Furthermore, we discuss some factors which lead to discontinuous global time.
- We integrate a computational efficient adaptive scheme into the proposed protocol to reduce energy consumption. Compared with previous prediction approaches, measuring sync errors reduces various error sources, such as inappropriate prediction models, aging and environment factors. On the other hand, querying table can find the correct initial sync period using only one step. The fast convergence is crucial to adaptive algorithms.

The remainder of the paper is organized as follows. Section II discusses basic theory of time synchronization and its related models. After that, we propose a precision adaptive average TS protocol in section III. Experimental results of our TS protocol are presented to show its superior in section IV. Finally, section V concludes our work and points out some future directions.

II. Basic Time Synchronization Theory

TS protocols help distributed nodes to establish a common acknowledgement of time in WSNs. Clock models, communication models and transformation functions are designed to analyze time synchronization procedures. In this section, we will introduce them separately.

A. Clock Models

Clock models are used to describe local time behavior of nodes. Typically, a crystal and a counter maintain the local physical clock of a node. The counter reading of node i at real world time t could be denoted as t_i. Its frequency f_i, is one order derivative of t_i: $f_i = dt_i/dt$, while phase offset p_i is the deviation of t_i from t: $p_i = t_i - t$.

Ideally, frequencies of local clocks which are synchronized to real world time should be 1 and phase offsets should be 0 all the time. However, due to crystal inaccuracies and environment variants (temperature, noise, aging and supply voltage), there would be slight differences between frequencies, which are called frequency skew Δf. There are various models based on different frequency skew assumptions [20].

- Constant-frequency model: though different, the frequency is assumed to be constant.
- Bounded-drift model: frequency would change over time, but clock's drift ρ_i ($\rho_i = f_i - 1$, the deviation of local frequency from real world time frequency) would be bounded by ρ_{max}: $-\rho_{max} \leq \rho_i \leq \rho_{max}$.
- Bounded-drift-variation model: the variation $\theta_i = d\rho_i/dt$ of clock drift is assumed to be bounded:$-\theta_{max} \leq \theta_i \leq \theta_{max}$.

B. Communication Models

Communication models describe the sync message exchange procedure between two nodes. During the progress, messages undergo various delays, which include send, access, propagation and receive time. The arbitrary uncertainties during these delays would decrease precision of TS protocols. Therefore, better understanding communication models help to reduce such uncertainties. Presently, the most popular solution is to time-stamp after accessing the media successfully [6] during the period of sending start frame detect byte [21]. This method can remove delay uncertainty effectively before accessing to the media. Ref. [8] illustrated decode/encode delay uncertainties and how to avoid them. Compared with transmit-receive mode, ref. [7] pointed out that receive-receive mode can remove delay uncertainties from sender side. For brevity, interested readers can refer to ref. [6] and [8] for detail descriptions of communication models.

C. Transformation Functions

Transformation functions define the conversion from a node's local time to the reference one. They are designed based on different clock models and communication models, including two-way offset delay function [6], interval transformation function [10], [22] and linear polynomial function [7], [8] and [9].

- Two-way offset delay function: it uses a two-way message exchange to get four timestamps, and computes phase offset and transmission delay with them.
- Interval transformation function: this function employs time intervals instead of common time instants. Bounded-drift clock assumption allows node j to estimate lower and upper bounds of node i's local time.
- Linear polynomial function: it employs constant frequency clock model and describes the relation between local time of node i and node j as $t_i = f_{ij}t_i + p_{ij}$. In which, t_i and t_j denote local time of node i and node j, f_{ij} and p_{ij} represent relative frequency and phase offset.

TS protocols using offset delay and interval functions can maintain an instantaneous synchronization, which means sync errors would increase quickly over time during the duration between sync points. As a result, they need exchange sync messages frequently to keep sync errors staying below the specified level. When using linear polynomial function, designers usually import linear regression technique to estimate frequency and offset precisely. By considering frequency variation, linear polynomial function maintains synchronization longer than the other two. But it needs extensive computation.

In our work, we use a simplified linear transformation function by assuming different nodes' clocks run at the same frequency. The function is denoted by (1). In which O_{ij} is temporal phase offset between node i and j computed at the sync point.

$$t_j = t_i + O_{ij} \tag{1}$$

III. A Precision Adaptive Average Time Synchronization Protocol

In this section, we first illustrate the basic average time synchronization (ATS) protocol implementation. After that, we integrate a measure and query based adaptive mechanism to increase its flexibility and power efficiency. The proposed precision adaptive average TS protocol mainly focuses on the cluster-based sensor networks.

A. Description and Realization of ATS protocol

Given a cluster, the proposed average time synchronization algorithm can be expressed as follow:

Algorithm 1 ATS implementation in a cluster

```
 1: Leader broadcasts a sync time message
 2: Records leader's send-out time $t_{leader}$
 3: for each node $i$ in the cluster do
 4:     Records the sync time message arrival time $t_i$
 5:     Sends $t_i$ back to leader node
 6: end for
 7: $t^{global} = (t_{leader} + \sum t_i)/n$
 8: Leader broadcasts computed global time $t^{global}$
 9: Records leader's send-out time $t'_{leader}$
10: for each node $i$ in the cluster do
11:     Records the global time message arrival time $t'_i$
12:     $O_i = t'_i - t^{global}$
13: end for
14: $O_{leader} = t'_{leader} - t^{global}$
```

At every sync point, ATS protocol can be realized by two steps: sync time broadcast period and global time broadcast period. As lines 1-2 have shown, leader broadcasts a sync time message and records its send-out time t_{leader}. In lines 3-6, every node i records arrival time of sync time message t_i and reports it back to leader. In line 7, leader averages all these recorded time stamps to get current global time t^{global}. Then the global time broadcast period starts. In lines 8-9, leader broadcasts a global time message which contains the computed global time t^{global} and records its send-out time t'_{leader}. In lines 10-13, every node i records its arrival time t'_i and computes its phase offset and line 14 shows the leader's phase offset calculation. With these offsets, nodes can estimate global time locally using $t_i^{global} = t_i^{local} - O_i$.

It seems more reasonable to compute phase offsets by subtracting global time t^{global} from sync time arrival time t_i because both of them are recorded and computed during sync time exchange period. It is true theoretically, but in practical realization, this method result in a loss in precision. Due to message delays and frequency skew, the phase offsets would increase when the leader node waits for a short but non-neglectable duration to collect all replies, Therefore, we compensate the increased offsets by subtracting global time from t'_i, the arrival time of global time message.

Due to the limited nodes we have, only in-cluster experiments are carried out at present. But network-wide time synchronization can be achieved based on in-cluster synchronization by sync message exchange between leaders or gateways. Our in-cluster time synchronization protocol is independent of how global synchronization finds a route to deliver sync messages.

B. Integrating Adaptive Mechanism into ATS protocol

Adaptive mechanism could help TS protocols self-adjust its sync precision and periods. For one certain service user, it finds the best fit sync period to exchange fewer sync messages. For different users, it helps TS protocols to switch according to their sync requirements. It could choose a shorter sync period when fine-grained precision is required and a much longer one when coarse-grained sync accuracy is needed. Therefore, energy could be saved due to less unnecessary sync messages. Fig. 1 shows our adaptive control loop integrated in a TS protocol.

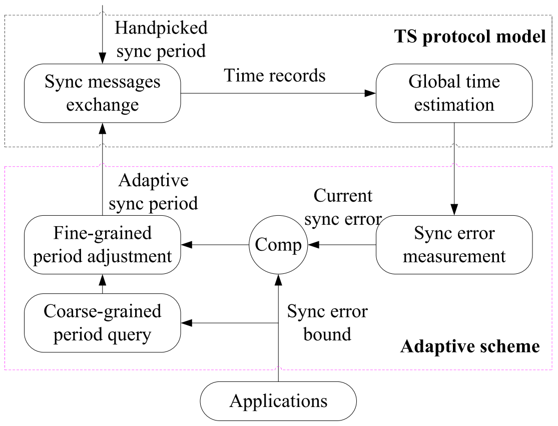

Fig. 1. Adaptive time synchronization control loop.

TS protocol model represents a general synchronization procedure, while adaptive scheme part provides a feedback for the TS protocol model to judge whether current sync period is suitable for required sync error bound and adjust the period adaptively. Traditional adaptive methods predict sync error and adjust period recursively, which suffer from both extensive computation and slow convergence. Our approach picks up a coarse-grained period from a period-error (PE) table to insure that adaptive scheme converges quickly. What's more, direct measuring can avoid computation errors.

1) Sync period adjustment: Adjusting sync period adaptively means two problems to solve: one is where to start and the other is how to adjust. Common solutions to the second problem is to increase or decrease the period recursively, either by additive or multiplicative operations. To the first one, there are no specific solutions presented yet. However, the initial period should be chosen carefully because it has a crucial impact on the convergence speed. To start from one period which is far away from the goal period consumes more power and could not satisfy applications in time.

In order to find a correct initial period quickly, we design a PE table to store a list of sync periods and corresponding maximal sync errors. Every time when given a new sync error bound, we query the table to find a suitable period (the one whose corresponding sync error is the first smaller one than bound). This method is reasonable because there is an approximate determinate dependence relationship between sync period T and its sync error e for one certain synchronization protocol. The relationship could be denoted as (2):

$$e = e_0 + \Delta f_{max} * T \tag{2}$$

In which, e_0 is the small instantaneous sync error at sync point, and Δf_{max} is maximal skew among nodes' local frequencies that would be unchanged for a relative long time given a cluster of nodes. Therefore, the sync error is proportional to sync period. This scheme would provide one-step convergence speed to find an initial period. Due to the limited store space on the nodes, we should establish a short table according to applications' requirements. This is realizable and extensible because the number of applications using time synchronization service is limited and we can add new items into the table when a new requirement arises. It should be noted that environment variants, such as noises, temperature and aging, could influence the period-error corresponding relationships. Therefore, the laboratorial table might be not suitable for other cases, such as a forest. In such a case, we can create a new table by self-training before querying them.

2) Sync error measurement: Once a period is chosen, error prediction helps to judge whether sync errors are staying below the bound or not. The answers guide us to adjust period further. However, most precise prediction methods are not suitable for WSN nodes because they are too complex to be realized on them. Instead of prediction, we measure sync errors at sync points.

For example, at the kth sync point, supposing the sync time message exchange time recorded by a cluster node i locally is $t_i^{k,local}$, it could estimate the global time at the moment using (3), and reports it to the leader, where O_i^{k-1} is phase offset computed during $(k-1)$th sync exchange by node i.

$$\hat{t}_i^{k,global} = t_i^{k,local} - O_i^{k-1} \tag{3}$$

Then the leader compares all estimated global time, the maximal difference among them would be the measured error under current sync period, it could be denoted by (4).

$$e_{max}^k = max|\hat{t}_i^{k,global} - \hat{t}_j^{k,global}|, \forall i, j \in \{1, \ldots, n\} \tag{4}$$

Given a new error bound E_{max}, the procedure of the adaptive-ATS protocol is shown as below.

Algorithm 2 Adaptive-ATS protocol in a cluster

```
1: Leader chooses initial sync period $T^{k-1}$ from PE table
2: Records sync time message send out time $t_{leader}^{k,local}$
3: for each node $i$ in the cluster do
4:     Records the sync time message arrival time $t_i^{k,local}$
5:     Computes current global time $\hat{t}_i^{k,global} = t_i^{k,local} + O_i^{k-1}$
6:     Reports $\hat{t}_i^{k,global}$ to leader
7: end for
8: $e_{max}^k = max|\hat{t}_i^{k,global} - \hat{t}_j^{k,global}|, \forall i, j \in \{1, \ldots, n\}$
9: if ($e_{max}^k < E_{max}$) then
10:     $T^k = T^{k-1}$
11: else
12:     leader chooses new period $T^k$
13: end if
```

IV. Experimental Results

In this section, we will evaluate the performance of proposed adaptive protocol. Firstly we test sync precision and global time continuity during leader re-elections. As a reference protocol, we realize the direct broadcast time synchronization (DBTS) protocol used by most cluster based TS protocols. In the DBTS protocol, the root node simply broadcasts its local time as the cluster's global time, then members compute their offsets accordingly. After that, we will compare power consumption of ATS and adaptive-ATS, and convergence speed of adaptive-ATS and traditional recursive methods.

A. Experiment Platform

Our experiment network is organized based on clusters. How to divide the whole network into clusters will not be discussed in this paper, and there are many algorithms which can do that [12], [13], [14]. We suppose that in a cluster all nodes could be heard by others in the same cluster. There are four Micaz nodes in our experiment cluster, one leader and three members, and their local clocks' resolution is set to 1 μs (microseconds). Usually, faster logical clock frequency leads to more precise time stamps but consumes more energy.

B. Synchronization Precision

Precision indicates the smallest sync error, and Fig.2 represents the sync error comparison between DBTS and ATS. The two protocols have comparable sync precision. It is reasonable because the global time broadcast period of our protocol is a similar procedure like DBTS. The main difference is that we use the average value of cluster nodes' replies as the global time. As we can see later, both global time continuity and adaptive mechanism would benefit from this reply collection and averaging procedure. In our experiment, both protocols run once every 5 seconds, the minimal sync errors achieved are about 4-5μs. However, as we can see from Fig. 2, some outrange points exist in both of them because the simplified communication model does not consider the influence of uncertainty of message delays. This instability could be solved by introducing new communication models considering message delay, which will be covered in our future work.

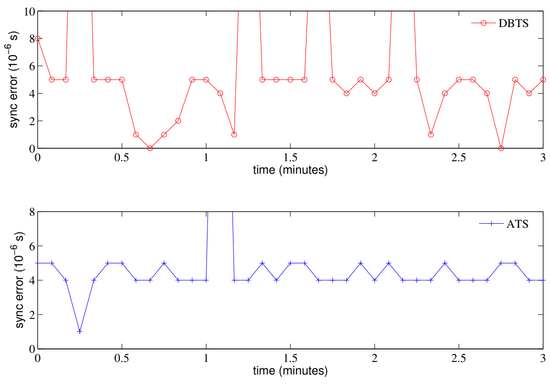

Fig. 2. Synchronization error comparison between DBTS and ATS.

C. Global Time Continuity

Global time continuity is important to those applications, which rely on temporal sequence. However, WSN dynamics such as leader re-elections, disabled nodes, move-in, move-out and environment variants might influence it. Fig. 3 represents the experiment result of global time continuity during leader re-elections.

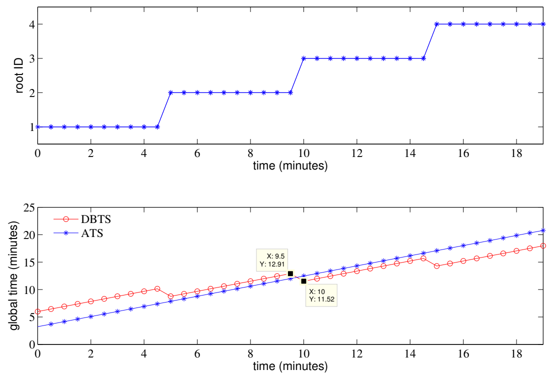

Fig. 3. Global time continuity during leader re-elections. Leader ID changes from 1 to 4 in the first figure, and global time of DBTS and ATS is presented in the second figure.

Leader re-election is a common strategy used by many energy-aware topology protocols to prolong the network's lifetime [12], [13], [14]. When leader node is re-elected, global time of DBTS changes from old leader's local time to new leader's local time. The alteration depends on the initial phase offset and increased offset during the duration between re-elections. In this experiment, the leader changes from 1 to 4 orderly, and nodes are turned on sequentially at 0, 2, 4 and 6 minute respectively. Global time is measured once every 0.5 minute. During the first re-election, we can see from Fig. 3 that the global time changes from 12.91 minutes (node 1's local time) to 11.52 minutes (node 2's local time). That means retro gradation on time. If nodes record time stamps during this period, their stamps recorded before and after leader re-election would be reversal. On the other hand, ATS can maintain stable continuous global time. Global time of two protocols appears different features because of their different global time strategies. ATS chooses average value of nodes' local time as global time while DBTS chooses only one node's time.

D. Precision Performance of Adaptive-ATS

We will test the correctness of adaptive-ATS to verify that it can find the appropriate sync period effectively. The given sync error bound will switch among $100\mu s$, $300\mu s$ and $600\mu s$ orderly, and every bound lasts for 5 minutes. The upper figure of Fig. 4 represents the sync errors under three different error bounds. Adaptive-ATS can always keep sync errors staying below different error bounds all the time. It is noted that there is a difference between sync error (0.436ms, millisecond) and error bound (0.6ms). This is because the given error bounds might not happen to be those stored in the PE table, so the sync periods are not their perfectly corresponding ones. There are two solutions to further refine the sync period. Firstly, we can self-train to get the best sync period for a given error bound. This method is effective when we know all applications' requirements. Secondly, we could iteratively improve the sync period after choosing a coarse one from the table. The later method is more general but may be also slower.

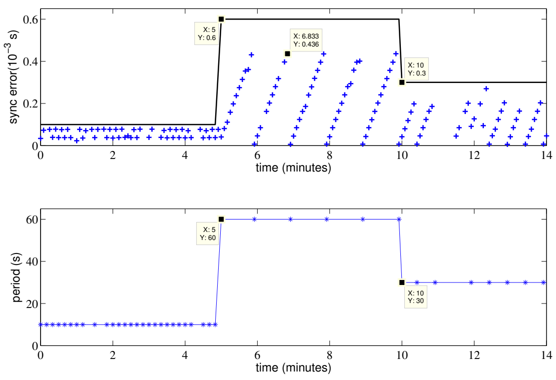

Fig. 4. Sync performance and period change under different error bounds. The first figure plots all errors during all three different error bounds (0.1ms, 0.6ms and 0.3ms). The second figure represents period change when error bound switches to 0.6ms at 5 minutes and 0.3ms at 10 minutes, it represents quickly to error bound change.

E. Power Consumption and Convergence Speed

In this section, we will first compare our adaptive ATS protocol with original ATS protocol to analytically prove that the adaptive mechanism decreases power consumption effectively. Secondly, we compare its convergence speed with previous recursive methods to show that our one-step convergence is quite power efficient. The second part of Fig. 4 presents the sync period change procedure during switching among various bounds. For three bounds (bound changes from $100\mu s$, to $300\mu s$ and $600\mu s$), adaptive-ATS protocol picks up corresponding period which is 10s, 30s and 60s, respectively. There are 45 times synchronizations totally in the adaptive ATS procedure. However, 90 times sync message exchanges are observed in original ATS protocol, since it has to synchronize every 10 seconds to ensure all error bounds can be satisfied. In this case, our adaptive-ATS method lowers the communication power to a half. Besides synchronization times reduction, Fig. 4 also illustrates that adaptive-ATS approach has a very quick response to a new error bound, because adaptive-ATS protocol gets the appropriate initial period immediately by querying PE table. Whereas, previous recursive methods [15], [16] need dozens of steps, which brings up more message exchanges. For example, assuming the error bound is $300\mu s$, the recursive method starts from period which is 1 seconds, and multiplies it by 2 every step. As a result, it needs 5 steps to find our initial period (30s), which uses additional 31 seconds and 5 sync message exchanges.

V. Conclusion

TS protocols are designed to serve other applications that need time synchronization, so we should pay more attentions to make them adapt to requirements from real WSN applications rather than improving their own performance only. In our opinion, global time continuity, power efficiency and adaptability to application are three design metrics which should be concerned about. We propose an ATS protocol keeping monotonous global time during leader re-elections, as well as achieving comparable precision with DBTS. What's more, we introduce an adaptive mechanism into our ATS protocol. It is shown that our protocol effectively adapts to different sync accuracy requirements. As far as we know, the novel querying table method provides the fastest convergence speed to find appropriate initial period, which reduces node's sync message exchange number, and thus power consumption. In the future, we would extend this work into a long-term and networked-wide adaptive time synchronization protocol.

References

[1] M. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," in *IEEE Infocom*, 2004.

[2] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2002.

[3] L. Girod and D. Estrin, "Robust range estimation using acoustic and multimodal sensing," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 3, 2001.

[4] R. Gupta and S. Das, "Tracking moving targets in a smart sensor network," in *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, vol. 5, 2003.

[5] W. Yuan, S. Krishnamurthy, and S. Tripathi, "Synchronization of multiple levels of data fusion in wireless sensor networks," in *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, vol. 1, 2003.

[6] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the first international conference on Embedded networked sensor systems*. ACM Press New York, NY, USA, 2003, pp. 138–149.

[7] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 147–163, 2002.

[8] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM Press New York, NY, USA, 2004, pp. 39–49.

[9] M. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 2, 2003.

[10] K. Römer, "Time synchronization in ad hoc networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. ACM Press New York, NY, USA, 2001, pp. 173–182.

[11] S. Ping, "Delay Measurement Time Synchronization for Wireless Sensor Networks," Intel Research, IBR-TR-03-013, Tech. Rep., June 2003.

[12] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM Press New York, NY, USA, 2001, pp. 70–84.

[13] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, and C. MIT, "An application-specific protocol architecture for wireless microsensor networks," *Wireless Communications, IEEE Transactions on*, pp. 660–670, 2002.

[14] P. Santi and J. Simon, "Silence is golden with high probability: Maintaining a connected backbone in wireless sensor networks," in *Wireless Sensor Networks: First European Workshop, EWSN 2004, Berlin, Germany, January 19-21, 2004, Proceedings*. Springer, 2004.

[15] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, and M. Srivastava, "Estimating clock uncertainty for efficient duty-cycling in sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM Press New York, NY, USA, 2005, pp. 130–141.

[16] D. Macii and D. Petri, "An adaptive-rate time synchronization protocol for wireless sensor networks," in *Instrumentation and Measurement Technology Conference Proceedings, 2007 IEEE*, 2007.

[17] L. Lamport and P. Melliar-Smith, "Synchronizing clocks in the presence of faults," *Journal of the ACM*, vol. 32, no. 1, pp. 52–78, 1985.

[18] J. Welch and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," *Information and Computation*, vol. 77, no. 1, pp. 1–36, 1988.

[19] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *Computers, IEEE Transactions on*, vol. 55, no. 2, pp. 214–226, 2006.

[20] K. Romer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks," in *Handbook of Sensor Networks: Algorithms and Architectures*, 2005, pp. 199–237.

[21] D. Cox, E. Jovanov, and A. Milenkovic, "Time synchronization for zigbee networks," in *System Theory, 2005. SSST'05. Proceedings of the Thirty-Seventh Southeastern Symposium on*, 2005, pp. 135–138.

[22] L. Meier, P. Blum, and L. Thiele, "Internal synchronization of drift-constraint clocks in ad-hoc sensor networks," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. ACM Press New York, NY, USA, 2004, pp. 90–97.