PAPER

Lightweight Precision-Adaptive Time Synchronization in Wireless Sensor Networks*

Li LI[†a)], ***Member***, **Yongpan LIU**[†], **Huazhong YANG**[†], ***and*** **Hui WANG**[†], ***Nonmembers***

SUMMARY Time synchronization is an essential service for wireless sensor networks (WSNs). However, fixed-period time synchronization can not serve multiple users efficiently in terms of energy consumption. This paper proposes a lightweight precision-adaptive protocol for cluster-based multi-user networks. It consists of a basic average time synchronization algorithm and an adaptive control loop. The basic average time synchronization algorithm achieves $1\,\mu$s instantaneous synchronization error performance. It also prolongs re-synchronization period by taking the average of two specified nodes' local time to be cluster global time. The adaptive control loop realizes diverse levels of synchronization precision based on the proportional relationship between sync error and re-synchronization period. Experimental results show that the proposed precision-adaptive protocol can respond to the sync error bound change within 2 steps. It is faster than the exponential convergence of the adaptive protocols based on multiplicative iterations.

key words: *sensor networks, time synchronization, precision adaptability, energy efficiency*

1. Introduction

Wireless sensor networks (WSNs) have been gaining great attention over the last few years. They can be seemlessly integrated into various environments to monitor the objects of interest closely. A sensor network is usually composed of many low-cost nodes equipped with low-performance processors, short distance transceivers, and limited energy supplies. Therefore, protocols proposed for WSNs should be lightweight, energy efficient, and robust.

Time synchronization is a key service of WSNs. Various applications rely on the time agreement among nodes to realize their own functions, such as location protocols [1], TDMA protocols [2], power management protocols [3], objective tracking protocols [4], [5] and data fusion protocols [6]. Time synchronization protocols developed for traditional distributed systems are not suitable for WSNs because they require sufficient energy supplies and high-performance infrastructures. What they concern most is synchronization performance improvement. On the other hand, energy efficiency is important to WSNs' time synchronization protocols.

Time synchronization aims at decreasing time error among diverse nodes. Time-stamping delay uncertainty and frequency skew are two error sources of time synchronization protocols. To reduce time-stamping delay uncertainty, Su et al. [7] measured various time delays during communications and subtracted them from time stamps. Alternatively, Timing-sync Protocol for Sensor Networks (TPSN) [8] used a two-way exchange to compute transmission delay. It also utilized a MAC layer time stamping technique to avoid time latencies from MAC layer and the layers above. To compensate frequency skew, Reference Broadcast Synchronization protocol (RBS) [9] and Flooding Time Synchronization Protocol (FTSP) [10] estimated the global time frequency by linear regressions. Recently, Noh et al. [11] and Chaudhari et al. [12] proposed frequency skew and phase offset estimators based on the maximum likelihood theory with diverse delay assumptions.

Most proposed time synchronization protocols were designed under a constant sync error bound assumption, therefore their re-synchronization periods were fixed. However, when there are multiple synchronization users, the error bound may change because the users often have different lifetime requirements on time synchronization. For instance, a distributed beamforming protocol requires $100\,\mu$s on-demand synchronization, while a TDMA protocol may require $500\,\mu$s always-on synchronization. Obviously, fixed-period time synchronization protocols are not suitable for a changeable error bound. Firstly, they can not realize qualified synchronization precision when an error bound is improved. Secondly, the fixed-period working style will waste a lot of energy because it has to fulfill the best precision requirement even if a user only utilizes the synchronization service for a while (on-demand). The latter drawback is especially troublesome for WSNs.

An alternative solution is to let time synchronization self-adjust its re-synchronization period to realize diverse levels of precision. However, the existing period-flexible protocols are not designed for changeable sync error bounds. For example, Rate-adaptive Time Synchronization (RATS) [13] adapted to the changing clock drift and environmental conditions while Adaptive-Rate Time Synchronization Protocol (ARSP) [14] adapted to the percentage of synchronized nodes in a network. Moreover, both RATS and ARSP searched for a qualified re-synchronization period by multiplicative iterations, which converged slowly and could not ensure finding the optimal re-synchronization period when the sync error bound changed.

A lightweight precision-adaptive time synchronization

Manuscript received September 2, 2009.
Manuscript revised March 3, 2010.

[†]The authors are with the Department of Electronic Engineering, TNList, Tsinghua University, P.R. China.

[*]This work was supported in part by the 863 Program award under 2009AA01Z130 and in part by the NSFC awards under 60976032.

a) E-mail: lli03@mails.tsinghua.edu.cn

DOI: 10.1587/transcom.E93.B.2299

protocol for cluster-based multi-user networks is presented in this paper. It improves our previous work [15] in two aspects. Firstly, it employs two-way exchanges to achieve better precision performance. Secondly, it refines the way to define cluster global time to prolong re-synchronization period. In this paper, we first reformulate the synchronization problem and find the approximate proportional relationship between sync error and re-synchronization period through sync error decomposition. Further, we propose a precision-adaptive time synchronization protocol consisting of a basic time synchronization algorithm and an adaptive control loop. The basic time synchronization algorithm reduces the instantaneous error to $1\,\mu$s, which ensures the proportional relationship between sync error and re-synchronization period. Besides, the algorithm also prolongs re-synchronization period by taking the average of two specified nodes' local time to be cluster global time rather than using conventional methods. The adaptive control loop queries a table for an initial re-synchronization period and refines the period according to the proportional relationship between sync error and re-synchronization period. Experimental results show that the precision-adaptive protocol can respond to the sync error bound change in 1–2 steps. It is faster than the exponential convergence of the adaptive protocols based on multiplicative iterations.

The remainder of the paper is organized as follows. Section 2 introduces the related time synchronization theories. The precision-adaptive time synchronization protocol is presented in Sect. 3. Section 4 evaluates the protocol's performance. Finally, Sect. 5 concludes our work and points out future research directions.

2. Related Time Synchronization Theory

In this section, we introduce related time synchronization theories by explaining a basic time synchronization procedure and techniques used by its constituents.

2.1 Basic Time Synchronization

As shown in Fig. 1, basic time synchronization consists of local time stamping, message exchange control and global time estimation. Stamping local time mainly deals with eliminating delay uncertainty during sync message transmission. Message exchange control takes charge of sync route establishment for effective sync message spreading. After receiving sync messages, global time estimation computes the related conversion function parameters. Then a node can convert its local time to global time via conversion functions. Usually, time synchronization is executed periodically to prevent sync error increasing with time.

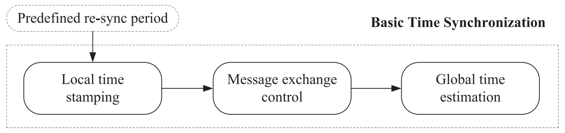

Fig. 1 Fixed-period time synchronization.

2.2 Local Time Stamping

When sync messages are processed, transmitted or received, there are delays between the stamped time and the practical execution time. These delays are hard to be removed completely because they are uncertain. As a result, sync errors are produced when using the delayed time-stamps to compute conversion function parameters. Basic delays include the latencies during sending, accessing, propagating and receiving durations [16]. More delay uncertainties during message transmission/reception, interrupt, decoding/encoding and byte alignment durations were illustrated in [10]. In WSNs, the propagation latency is negligible because it is at most tens of nanoseconds, which does not contribute significantly to the microsecond-scale error budget [9].

To remove the delay uncertainty from sender side, a receiver-receiver mode was utilized in RBS [9] to replace the conventional sender-receiver mode. In TPSN [8] and FTSP [10], timestamps were made after a node had accessed the medium to avoid latencies from MAC layer and the layers above.

2.3 Message Exchange Control

Message exchange control takes charge of route construction to deliver sync messages. Some synchronization users, such as TDMA and sleep-wake schedule protocols, only need local (single-hop) synchronization. In such a case, a synchronization coordinator (a leader node in a cluster, or a master node in the master-slave mode) can directly broadcast sync messages. However, users such as objective tracking and data fusion protocols require time agreement in a large scale. In this situation, network-wide routes should be established. Popular route architectures include level/tree hierarchy [8], [17], overlapping clustering [9], flooding [10] and diffusion [18].

2.4 Global Time Estimation

Global time estimation is responsible for converting a node's local time to global time based on local time models and conversion functions.

2.4.1 Local Time Model

At real world time t, a node v_i's local time $c_i(t)$ can be denoted by a polynomial function as (1), where frequency $f_i(t)$ is one order derivative of $c_i(t)$, and initial phase offset $p_i(t_0)$ is deviation of $c_i(t)$ from the real world time when it is turned on.

$$c_i(t) = f_i(t) * t + p_i(t_0) \tag{1}$$

Usually, two nodes will have different frequencies due

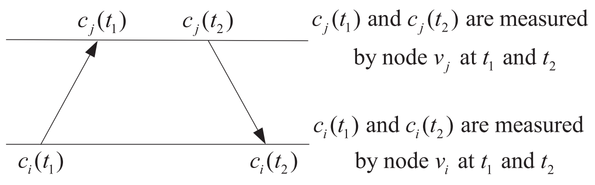
$c_j(t_1)$
$c_j(t_2)$
$c_j(t_1)$ and $c_j(t_2)$ are measured by node v_j at t_1 and t_2
$c_i(t_1)$ and $c_i(t_2)$ are measured by node v_i at t_1 and t_2
$c_i(t_1)$
$c_i(t_2)$

Fig. 2 Two-way message exchange.

to oscillator inaccuracies and environmental factors. The frequency difference between two nodes is called frequency skew. There are several clock models based on different frequency skew assumptions [19] (here f denotes instinct frequency).

- Constant-frequency model: every node's frequency is time invariant and different from others' ($f_i(t) = f_i \neq f$).
- Bounded-drift model: the local frequency will change over time, but its deviation from the instinct frequency $\rho_i(t) = f_i(t) - f$ is bounded by a limited constant: $|\rho_i(t)| \leq \rho_{max}$.
- Bounded-drift-variation model: the frequency skew variation is bounded by a constant: $|\frac{\partial \rho_i(t)}{\partial t}| \leq \theta_{max}$.

2.4.2 Conversion Functions

Several functions are available to convert a node's local time to global time. The most popular ones include interval conversion function [20], linear polynomial function [9], [10], [17] and two-way offset delay function [8], [11], [12].

Interval conversion mainly aims at pair-wise synchronization, in which a node estimates a bound of a remote node's time. It is suitable for high-dynamic networks.

Linear regression function utilizes the constant-frequency clock model and a least-square estimator to compute the relative frequency skew and relative phase offset between nodes. It is computational, but can prolong re-synchronization period.

Two-way offset delay function computes the relative phase offset between two nodes through message exchanges as shown in Fig. 2 [8]. If the duration of a two-way exchange ($t_2 - t_1$) is short enough, the phase offset between nodes v_i and v_j would not change ($O_{ij}(t_1) = O_{ij}(t_2)$), then $O_{ij}(t_1) = c_j(t_1) - c_i(t_1)$ and the transmission delay d can be computed as (2).

$$O_{ij}(t_1) = \frac{(c_j(t_1) - c_i(t_1)) - (c_i(t_2) - c_j(t_2))}{2}$$
$$d = \frac{(c_j(t_1) - c_i(t_1)) + (c_i(t_2) - c_j(t_2))}{2} \tag{2}$$

3. Precision-Adaptive Time Synchronization

In this section, we present a precision-adaptive time synchronization protocol. For consistency, we first describe the network model and definitions used in this paper. Then we analyze the composition of sync error and find the relationship between sync error and re-synchronization period. Based on the analysis, we propose a precision-adaptive time synchronization protocol and explain its basic time synchronization algorithm and adaptive control loop, respectively.

3.1 Network Model and Definitions

We consider a network based on static clusters. There is a leader node in each cluster which takes charge of intra-cluster scheduling and communication between a cluster and a base station. Member nodes inside a cluster directly communicate with the cluster leader. The work presented in this paper solves the synchronization problem of single-hop networks. For a multi-hop cluster-based network, inter-cluster time synchronization can be achieved by exchanging cluster global time between neighbor clusters and relating one cluster's global time to its neighbor clusters'. The work presented in this paper can be extended by this way to realize multi-hop synchronization.

Cluster global time is the standard time inside a cluster. In-cluster synchronization precision is defined as below.

Definition 1: at any time, *in-cluster synchronization precision* is the maximum error between all locally estimated cluster global time and the actual cluster global time.

3.2 Synchronization Error Decomposition

Since delay uncertainty and frequency skew are two sync error sources, we can divide sync error into instantaneous error and accumulated error accordingly. The instantaneous error can be measured at the moment when a node is just synchronized. It is produced by delay uncertainty and is time-invariant between two successive sync exchanges. The accumulated error is the increased error after a sync exchange, which is introduced by frequency skew. The two kinds of sync errors reflect different aspects of time synchronization performance. The instantaneous error implies how precise a time synchronization protocol can be and the accumulated error indicates how long the protocol can keep a network stay in the synchronized state.

Suppose node v_i is synchronized at time t_{s_k}, its sync error at time $t_\Delta = t_{s_k} + \Delta t$ can be represented as (3), where $e_i^0(t_{s_k})$ and $e_i^a(\Delta t)$ denote the instantaneous error at time t_{s_k} and accumulated error after duration Δt, f_i is node v_i's local frequency and f^g is global frequency.

$$\begin{aligned} e_i(t_\Delta) &= e_i^0(t_{s_k}) + e_i^a(\Delta t) \\ &= e_i^0(t_{s_k}) + |f_i - f^g| * \Delta t \end{aligned} \tag{3}$$

For a cluster of nodes, the in-cluster synchronization precision at time t_Δ is decided by the maximum sync error $e_{\max}(t_\Delta) = \max_i e_i(t_\Delta)$.

$$\begin{aligned} e_{\max}(t_\Delta) &= \max_i(e_i^0(t_{s_k}) + e_i^a(\Delta t)) \\ &\leq \max_i e_i^0(t_{s_k}) + \max_i |f_i - f^g| * \Delta t \end{aligned} \tag{4}$$

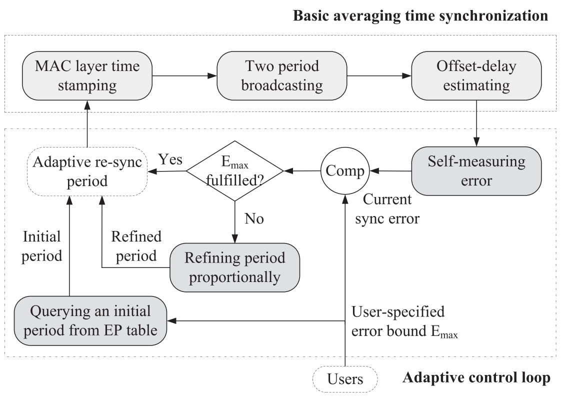

Fig. 3 Precision-adaptive time synchronization.

If the instantaneous error is small enough, the accumulated error will dominate the total sync error as time elapsing, that is

$$e_{\max}(t_\Delta) \approx \max_i e_i^a(\Delta t) = \max_i |f_i - f^g| * \Delta t. \tag{5}$$

It is well known that variations in environmental condition can affect nodes' frequencies. However, it takes time for significant effect. During this period of time, nodes' frequencies can be considered to be fixed. And the maximum frequency skew $\max_i |f_i - f^g|$ is also constant in a static cluster. Consequently, the accumulated sync error is directly proportional to the duration Δt. The proportional relationship then provides a way to decide the re-synchronization period when given a sync error bound. Let the error bound be $E_{\max}$, then the re-synchronization period T can be expressed as (6).

$$T \approx \frac{E_{\max}}{\max_i |f_i - f^g|} \tag{6}$$

However, when nodes' frequencies are almost the same, it is hard to compute the maximum frequency skew accurately. Alternatively, if we know the sync error E^* corresponding to some re-synchronization period T^*, we can compute the goal re-synchronization period under an error bound $E_{\max}$ as (7).

$$\left.\begin{aligned} E^* &\approx \max_i |f_i - f^g| * T^* \\ E_{\max} &\approx \max_i |f_i - f^g| * T \end{aligned}\right\} \Rightarrow T \approx \frac{E_{\max} * T^*}{E^*} \tag{7}$$

3.3 Precision-Adaptive Time Synchronization Architecture

The overall precision-adaptive time synchronization architecture is shown in Fig. 3. It consists of a basic time synchronization algorithm and an adaptive control loop. The basic time synchronization algorithm is responsible for prolonging re-synchronization period and reducing instantaneous error. Adaptive control loop takes charge of finding a qualified re-synchronization period quickly when user's error bound changes.

3.4 Basic Averaging Time Synchronization

For the benefit of saving energy, the basic averaging time synchronization algorithm prolongs re-synchronization period by utilizing average-style cluster global time. Moreover, the algorithm uses the MAC layer time stamping technique [21] and a two-period broadcast to reduce instantaneous error, which ensures the proportional relationship (5) between sync error and re-synchronization period.

3.4.1 Prolonging Re-Synchronization Period

Given a sync error bound, (6) indicates that the re-synchronization period is inversely proportional to the maximum frequency skew of a cluster. Therefore, we should try to reduce the maximum frequency skew to prolong re-synchronization period.

Some protocols, such as RBS and FTSP, reduced the maximum frequency skew by estimating the global frequency locally. They optimized local frequencies by turning f_i in (6) to $\hat{f}_i^g$. Alternatively, we reduce the maximum frequency skew of a cluster by optimizing the cluster global frequency f^g.

In an n-node cluster, the optimization problem can be formulated as (8). Given a local frequency series $f_1, f_2, \ldots, f_n$, we should find a global frequency to minimize the maximum global frequency skew.

$$\min_{f^g} \max_i |f_i - f^g| \tag{8}$$

The optimal solution is $f_*^g = \frac{f_{\max}+f_{\min}}{2}$, where $f_{\max}$ and $f_{\min}$ denote the maximum and minimum frequencies in a cluster.

Conventionally, a cluster leader utilizes its local time as cluster global time [22]. The global frequency then can be represented as $f^g = f_{leader}$. If the leader is chosen randomly, f_{leader} may be far from the optimal global frequency. To reduce the maximum global frequency skew, it is better to choose the node with a similar frequency as the optimal one to be a leader. [15] takes the average of all cluster members' local time to be cluster global time. Usually, it is a better choice than using the leader's local time, because it avoids choosing the maximum/minimum frequency as cluster global frequency.

Suppose v_α and v_β are the nodes with the maximum and minimum local frequencies, cluster global time can be defined as (9). This selection ensures that the global frequency is optimal.

$$c^g(t) = \frac{c_\alpha(t) + c_\beta(t)}{2} = \frac{f_\alpha + f_\beta}{2} * t + \frac{p_\alpha + p_\beta}{2} \tag{9}$$

As shown in Fig. 4, a node validation phase (NVP) is used to find nodes v_α and v_β. After two consecutive broadcasts, every node inside a cluster can record the time interval between two broadcasts $I_i(t_{v_m}) = c_i(t_{v_m}^2) - c_i(t_{v_m}^1)$. Suppose the real world time interval between two broadcasts is $\Delta t = t_{v_m}^2 - t_{v_m}^1$, the nodes' recorded time intervals can be

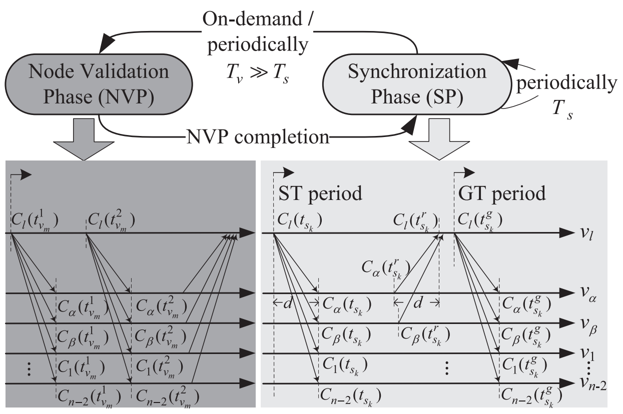

Fig. 4 Node validation phase and two-period broadcast synchronization phase.

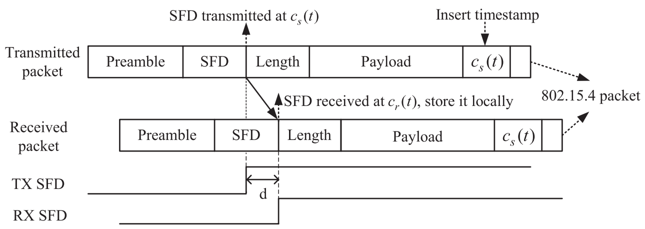

Fig. 5 Time stamping when transmitting and receiving SFD byte.

written as $I_i(t_{v_m}) = f_i * \Delta t$. It shows that a node with a faster clock will report a larger time interval value and a node with a slower clock will report a smaller value. After receiving all members' time intervals, the leader node compares them to determine nodes v_α and v_β. When synchronization phase (SP) starts, the leader node will inform nodes v_α and v_β to report their local time for cluster global time computation.

NVP can be executed in both "periodic" and "on-demand" modes to preclude the effect of environmental temperature variation on nodes' frequencies. In the periodic mode, NVP period T_v should be set according to the specific environmental condition in which a network is implemented. For example, ambient temperature of an indoor condition is more stable than that of an outdoor condition. As a result, an indoor network can have a longer NVP period than an outdoor network. In the on-demand mode, NVP is executed when ambient temperature changes[†]. Every time ambient temperature changes, the leader node records the temperature and the corresponding node pair v_α and v_β in a table. If encountering the same temperature in the future, the leader node can search the table for nodes v_α and v_β.

In the periodic mode, NVP period T_v can be set to tens of minutes (outdoors) to several hours (indoors), which is much longer than the second-scale re-synchronization period T_s. From Fig. 4, one NVP consumes $n + 1$ messages (2 broadcasts and $n - 1$ replies) and one SP consumes 4 messages in an n-node cluster. Suppose the cluster size is $n = 30$, T_v and T_s are 1 hour and 30 seconds, NVP will consume 31 messages in an hour, which is 6% of SP's message overhead (480 messages in an hour). In the on-demand mode, NVP is only executed when the ambient temperature is not recoded in the table, which consumes less message overhead than periodic mode. Therefore, NVP will not introduce significant overhead burden to the overall time synchronization algorithm.

3.4.2 Reducing Instantaneous Error

As is well known, a synchronization word will be appended at the head of a packet to indicate the starting point of data during wireless communication. MAC layer time stamping technique makes a timestamp when a sender node is transmitting some specific bit of the synchronization word and inserts the timestamp into the message being transmitted. Accordingly, a receiver node also makes a timestamp when the same bit is received.

For different types of messages, the positions of the specific bit are different. Figure 5 explains the composition of an IEEE 802.15.4 packet and interrupt signal changes on sender's and receiver's SFD pins [21], [23]. Here, the start of frame delimiter byte (SFD) plays the role of synchronization word. Timestamps can be recorded when SFD has been transmitted and received. Note that the practical transmission delay d in Fig. 5 is a constant when data rate is fixed, and it equals the time to transmit the last bit of SFD byte. Therefore, message transmissions in both directions have almost the same transmission delay and the delay can be computed using a two-way exchange.

However, the conventional two-way exchange is started by a slave node (a member node in a cluster), which consumes heavy message overhead. We modified it into a two-period broadcast as shown in Fig. 4. During the sync time (ST) broadcast period, the leader node uses time stamps from the first replied node (in Fig. 4, it is v_α) to compute transmission delay, as is shown in (10).

$$d = \frac{(c_l(t^r_{s_k}) - c_l(t_{s_k})) - (c_\alpha(t^r_{s_k}) - c_\alpha(t_{s_k}))}{2} \tag{10}$$

3.4.3 Algorithm Description

The basic average time synchronization (ATS) algorithm can be described as algorithm 1.

ATS starts with broadcasting an ST message. Nodes v_α and v_β then send reply messages to leader node, which contain their recorded ST message arrival time and the reply message send-out time. When receiving the first reply message, the leader computes transmission delay d. After collecting two replies, it takes the average of the reported ST arrival time to compute the global time $c^g(t_{s_k})$ during s_kth synchronization. Then the global time broadcast period begins with broadcasting a GT message containing $c^g(t_{s_k})$. Every member v_i computes its phase offset by subtracting global time $c^g(t_{s_k})$ from its recorded ST arrival time $c_i(t_{s_k})$. The final line shows the leader's phase offset calculation.

[†]Temperature changes can be detected by leader node's onboard temperature sensor.

Algorithm 1 ATS implementation in a cluster

```
1: During $s_k$th synchronization, leader broadcasts an ST message and
   records its send out time $c_l(t_{s_k})$
2: for Each member node $v_i$ in the cluster do
3:     Records ST message arrival time $c_i(t_{s_k})$
4: end for
5: for Nodes $v_\alpha$ and $v_\beta$ do
6:     Report $c_{\alpha,\beta}(t_{s_k})$ and reply message send out time $c_{\alpha,\beta}(t^r_{s_k})$ to leader
7: end for
8: if Leader receives the first reply then
9:     Records its arrival time $c_l(t^r_{s_k})$
10:    Computes transmission delay $d$
11: end if
12: if Leader receives both replies then
13:    Computes global time $c^g(t_{s_k}) = \frac{c_\alpha(t_{s_k})+c_\beta(t_{s_k})}{2}$
14:    Broadcasts a GT message containing $c^g(t_{s_k})$
15: end if
16: for Each member node $v_i$ in the cluster do
17:    Computes phase offset $O_i(t_{s_k}) = c_i(t_{s_k}) - c^g(t_{s_k})$
18: end for
19: Leader computes phase offset $O_l(t_{s_k}) = c_l(t_{s_k}) + d - c^g(t_{s_k})$
```

Before s_{k+1}th synchronization starts, node v_i can convert its local time to global time by (11).

$$\hat{c}^g_i(t) = c_i(t) - O_i(t_{s_k}), t \in (t_{s_k}, t_{s_{k+1}}) \tag{11}$$

3.5 Precision-Adaptive Time Synchronization

Now we add an adaptive control loop to ATS to complete the precision-adaptive time synchronization protocol (adaptive-ATS). The adaptive control loop consists of three parts: re-synchronization period initiation, error self-measurement and proportional period refinement.

3.5.1 Re-Synchronization Period Initiation

Initial period should be chosen carefully because it will influence the convergence speed when searching for the qualified re-synchronization period. However, neither RATS nor ARSP considered this issue. They started searching from the current period or a predefined period.

According to the proportional relationship between sync error and re-synchronization period, an error-period (EP) table is designed to store a list of re-sync periods and their corresponding sync errors. Whenever given a new sync error bound, adaptive-ATS queries the table to find whether the error bound is in or not. If it is in, the protocol uses its corresponding period as the initial period, otherwise it starts searching from a predefined period. Since synchronization users are limited, the EP table realization only uses a little storage space. Besides, new items can be added into the table when new users arise.

An EP table can be obtained from experiments, but the laboratorial results might not be suitable for arbitrary cases due to environmental changes. Generally, it can also be obtained by self-training.

3.5.2 Error Self-Measurement

Once a re-synchronization period is chosen, an important step is to judge whether the current period has fulfilled the sync error bound. ARSP utilized a Simple Response Method (SRM) to collect all nodes' states. However, it introduced heavy message and energy overhead because it used replies from all nodes. Differently, error measurement of adaptive-ATS can be accomplished via ATS's two replies for global time computation.

Suppose the re-synchronization period is set to $T_{s_{k-1}}$ at s_{k-1}th synchronization, ATS's maximum error before s_kth synchronization starts can be approximately denoted by (12).

$$\begin{aligned} e_{\max}(T_{s_{k-1}}) &\approx \max_i |f_i - f^g| * T_{s_{k-1}} \\ &\approx \max_i \left| f_i - \frac{f_\alpha + f_\beta}{2} \right| * T_{s_{k-1}} \end{aligned} \tag{12}$$

Obviously, ATS's two reply nodes have the maximum global frequency skew. Therefore, when s_kth synchronization starts at time t_{s_k}, nodes v_α and v_β estimate global time $\hat{c}^g_\alpha(t_{s_k})$ and $\hat{c}^g_\beta(t_{s_k})$ using their respective offsets computed during s_{k-1}th synchronization, and report them to the leader node. Then the leader node can obtain the sync error corresponding to re-synchronization period $T_{s_{k-1}}$.

3.5.3 Re-Synchronization Period Refinement

Sometimes, though the precision performance achieved by the initial period can fulfill the given sync error bound, there will be difference between the actual sync error and the required error bound. Therefore, the initial period should be refined.

RATS and ARSP refined the re-synchronization period by multiplicative iterations. However, the iterative step length is hard to define because a fixed setting might not be suitable for diverse situations. Optionally, given an error bound $E_{\max}$, the re-synchronization period can be refined from $T_{s_{k-1}}$ to T_{s_k} using (13).

$$T_{s_k} \approx \frac{E_{\max} * T_{s_{k-1}}}{e_{\max}(T_{s_{k-1}})} \tag{13}$$

The last problem is to decide when to stop period refinement. To make sure the sync error remain below the error bound, we admit the adaptive algorithm has found a qualified period and the searchings stop when the sync error is within $[B_l * E_{\max}, B_u * E_{\max}]$. In this paper, B_l and B_u are set to 0.8 and 0.9, respectively. Therefore, (13) should be multiplied by a factor as (14).

$$T_{s_k} \approx \frac{B_l + B_u}{2} * \frac{E_{\max} * T_{s_{k-1}}}{e_{\max}(T_{s_{k-1}})} \tag{14}$$

3.5.4 Protocol Description

Adaptive-ATS can be summarized as algorithm 2. Firstly,

Algorithm 2 Adaptive-ATS in a cluster

```
1: if Given an error bound $E_{\max}$ at $t_{s_{k-1}}$, $k=1$ then
2:     Run ATS once to compute phase offset $O_i(t_{s_{k-1}})$
3:     Leader chooses an initial re-sync period $T_{s_{k-1}}$
4: end if
5: while New synchronization starts at $t_{s_k} = t_{s_{k-1}} + T_{s_{k-1}}$ do
6:     Leader broadcasts an ST message and records its send out time
7:     for Each node $v_i$ in the cluster do
8:         Records ST message arrival time
9:     end for
10:    for Nodes $\alpha$ and $\beta$ do
11:        Estimate global time $\hat{c}^g_{\alpha,\beta}(t_{s_k})$ at $t_{s_k}$:
12:        $\hat{c}^g_{\alpha,\beta}(t_{s_k}) = c_{\alpha,\beta}(t_{s_k}) - O_{\alpha,\beta}(t_{s_{k-1}})$
13:        Report estimated global time, ST arrival time and the reply message send out time to leader
14:    end for
15:    if Leader receives both replies then
16:        Computes global time $c^g(t_{s_k})$ and transmission delay $d$
17:        Broadcasts a GT message containing $c^g(t_{s_k})$
18:        Computes error $e_{\max}(T_{s_{k-1}})$ under period $T_{s_{k-1}}$:
19:        $e_{\max}(T_{s_{k-1}}) = \max(|\hat{c}^g_{\alpha,\beta}(t_{s_k}) - c^g(t_{s_k})|)$
20:        if $(B_l * E_{\max} \leq e_{\max}(T_{s_{k-1}}) \leq B_u * E_{\max})$ then
21:            $T_{s_k} = T_{s_{k-1}}$
22:        else
23:            $T_{s_k} = \frac{(B_u+B_l)}{2} * \frac{T_{s_{k-1}} * E_{\max}}{e_{\max}(T_{s_{k-1}})}$
24:        end if
25:    end if
26:    for Each node $v_i$ in the cluster do
27:        Computes phase offset $O_i(t_{s_k})$
28:    end for
29:    $k = k + 1$
30: end while
```

adaptive-ATS runs ATS once for all cluster nodes to compute phase offsets and picks up an initial re-synchronization period from EP table. Whenever the next synchronization starts, adaptive-ATS judges whether the current re-synchronization period is qualified by comparing the measured sync error and the error bound. If the period is a qualified one, adaptive-ATS will keep it unchanged, otherwise it refines it according to the proportional relationship between sync error and re-synchronization period.

4. Performance Evaluations

In this section, we evaluate the performance of ATS and adaptive-ATS. Firstly, we introduce the experiment platform. Then, we test ATS's instantaneous error and accumulated error growth speed. Finally, we evaluate adaptive-ATS's performance under a changeable sync error bound.

4.1 Experiment Platform Setup

Our experimental cluster was composed of four Micaz sensor nodes [24], including one leader and three member nodes. A Micaz node has an Atmel128L MCU and a CC2420 transceiver. The local clock resolution was set to $1\,\mu$s. Two additional Micaz nodes were used as beacon node and base station node, respectively. The beacon node broadcasted beacon messages to the experimental cluster periodically. When the tested nodes received the beacon messages, they reported the estimated global time. The base station node was connected to a computer. It collected all reported messages and sent them to the computer for analysis. Since the beacon messages were received simultaneously by the tested nodes, message transmission delays would not introduce additional errors to experimental results. Additionally, due to limited available nodes, computer simulations were used as a complementarity to evaluate protocol performance when cluster size became larger.

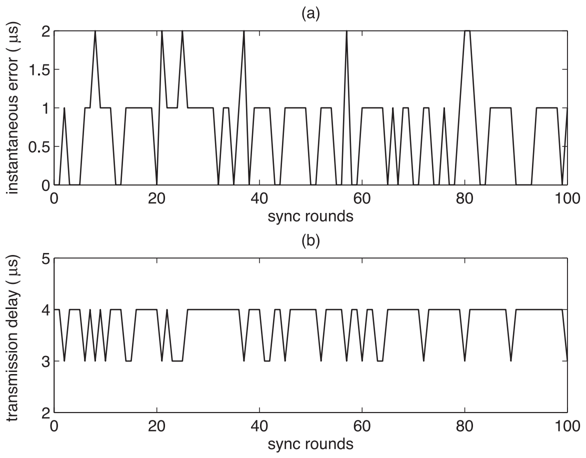

Fig. 6 (a) Instantaneous synchronization error. (b) Estimated transmission delay.

4.2 Performance of ATS

4.2.1 Instantaneous Sync Error

Firstly, we test ATS's instantaneous synchronization error. In the experiment, the re-synchronization period was set to 5 seconds, and the beacon message was sent every 1 second. Figure 6(a) represents the instantaneous sync error performance. ATS's instantaneous sync error is about $1\,\mu$s, ensuring the approximate proportional relationship in (5). Further, since the effective data rate of CC2420 is 250 kbps, the time to transmit the last bit of SFD byte is $4\,\mu$s, which coincides with ATS's estimated delay as shown in Fig. 6(b).

4.2.2 Accumulated Sync Error Growth Speed

Now we evaluate the growth speed of ATS's accumulated sync error. A reference protocol was also realized in the experimental cluster. It used the leader's local time as cluster global time and periodically broadcasted it for member nodes to compute their phase offsets. We call it the direct broadcast time synchronization (DBTS).

In the experiment, all nodes were synchronized only once and the sync error growth is recorded in Fig. 7(a). We can see that DBTS's sync error increases faster than ATS's. The ratio of error growth speed $DBTS/ATS$ is about 1.3.

We would like to find the typical error growth speed ratio of DBTS and ATS through the following simulations. In a 30-node cluster, the nodes' frequencies were assumed

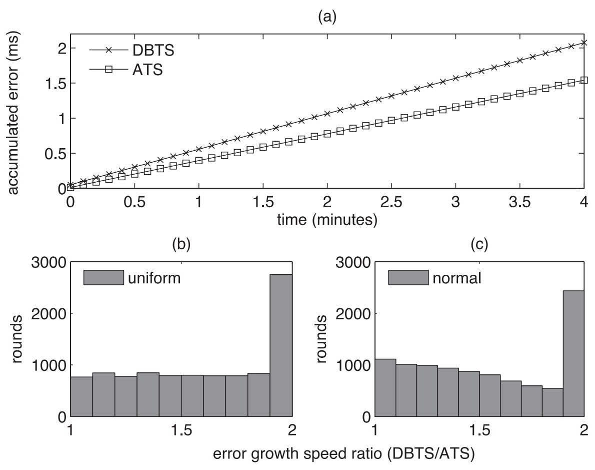

Fig. 7 Growth speed comparison of DBTS's and ATS's accumulated sync errors. The results in (a) are measured from the Micaz experimental cluster and the results in (b)&(c) are from simulations.

Table 1 Synchronization requirements.

Synchronization users	Error bound	Lifetime
SMAC [2]	$500\,\mu$s	always-on
Distributed beamforming [25]	$100\,\mu$s	on-demand

to be

- independently and uniformly distributed: $f_i \sim U(f - \rho, f + \rho)$, $\rho = 30$ ppm;
- independently and normally distributed: $f_i \sim N(f, \rho^2)$, $\rho = 10$ ppm.

The simulations were repeated 10^4 times and all error growth speed ratios are plotted in Fig. 7(b)&(c). In each case, DBTS's errors increase faster than ATS's and the possibility that the speed ratio is close to 2 exceeds 20%.

4.3 Performance of Adaptive-ATS

Further, we tested the performance of adaptive-ATS under a changeable error bound. Table 1 shows synchronization requirements from two users. Since SMAC and distributed beamforming have different lifetime requirements on synchronization service, the error bound will switch between $500\,\mu$s and $100\,\mu$s.

4.3.1 Precision Performance

Precision performance of adaptive-ATS under a changeable error bound is presented in Fig. 8. In this experiment, the error bound changed from $100\,\mu$s to $500\,\mu$s, then started again from $100\,\mu$s. Figure 8 shows that adaptive-ATS can adapt to the precision bound change.

4.3.2 Convergence Speed

When the measured error $e_{\max}(T)$ is within $[B_l * E_{\max}, B_u * E_{\max}]$, the re-synchronization period T is considered to be qualified. Therefore a qualified period T_{goal} should be within the bounds of (15).

$$\frac{B_l * E_{\max}}{\max_i |f_i - f^g|} \leq T_{goal} \leq \frac{B_u * E_{\max}}{\max_i |f_i - f^g|} \tag{15}$$

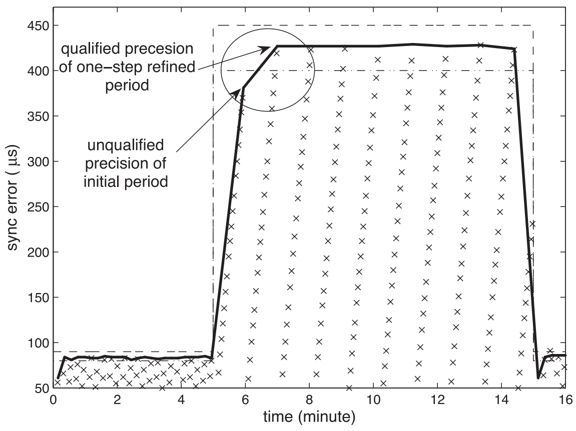

Fig. 8 Precision performance of adaptive-ATS under a changeable error bound. The dashed curves represent the lower and upper bounds: $B_l * E_{\max}$ and $B_u * E_{\max}$. The crosses mark the real-time sync errors and the solid line plots the maximum errors corresponding to the chosen periods.

For adaptive-ATS, even if its initial period $T_{s_{k-1}}$ is not located in the above bounds, the refined period will be. Recalling (14), note that $\frac{T_{s_{k-1}}}{e_{\max}(T_{s_{k-1}})} \approx \frac{1}{\max_i |f_i - f^g|}$, T_{s_k} can be denoted as

$$T_{s_k} \approx \frac{(B_l + B_u)}{2} * \frac{E_{\max}}{\max_i |f_i - f^g|}.$$

It is located in the bounds of (15) obviously. Therefore, adaptive-ATS will converge to the qualified period in 1–2 steps. Results in Fig. 8 also confirm this declaration. When the error bound changed from $100\,\mu$s to $500\,\mu$s, the maximum synchronization error corresponding to the queried initial period is not within the error bound $[400, 450]\,\mu$s. After one-step adjustment, the maximum error of the refined period can meet the error requirement.

As a reference method to search for a qualified re-synchronization period, the multiplicative increase, multiplicative decrease method (MIMD) [13] was also realized. When given a new error bound, MIMD begins searching for the qualified period from a fixed initial period. If the measured error exceeds $B_u * E_{\max}$, it divides the current period by κ, else if the error is smaller than $B_l * E_{\max}$, it multiplies the current period by κ.

The period refinement of MIMD can be represented by (16). However, it can not ensure that MIMD will converge to a qualified period within the bounds of (15). MIMD's searchings might vibrate around the bounds. Besides, even if MIMD finally finds a qualified period T_{goal}, it will use $\log_\kappa \frac{T_{goal}}{T_{initial}}$ steps. A large κ might enable MIMD to approach the qualified period quickly, but it will also add to the risks of vibrating around the goal period.

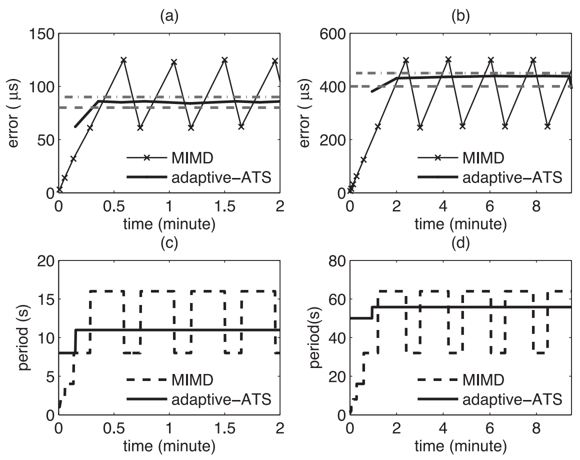

Fig. 9 Convergence comparison between MIMD and adaptive-ATS. (a)&(b) plot error convergence performance of MIMD and adaptive-ATS under error bounds 100 μs and 500 μs. (c)&(d) represent synchronization period convergence performance of MIMD and adaptive-ATS under error bounds 100 μs and 500 μs.

$$T_{s_k} = \begin{cases} \kappa * T_{s_{k-1}}, & \text{if } e_{\max}(T_{s_{k-1}}) < B_l * E_{\max}, \\ \frac{1}{\kappa} * T_{s_{k-1}}, & \text{if } e_{\max}(T_{s_{k-1}}) > B_u * E_{\max}, \\ T_{s_{k-1}}, & \text{others.} \end{cases} \tag{16}$$

Figure 9 presents convergence comparison between MIMD and adaptive-ATS. In the experiment, precision bound was fixed to 100 μs and 500 μs respectively. The step adjustment parameter of MIMD was $\kappa = 2$ as in [13], and the initial period was set to 1 second. In both situations, adaptive-ATS can find the qualified periods within 2 steps and its error performance keeps stable within the bounds. Conversely, MIMD vibrates around the qualified periods and so do its errors.

5. Conclusion

This paper proposes a lightweight precision-adaptive time synchronization protocol (adaptive-ATS) for multi-user networks. It consists of a basic average time synchronization (ATS) algorithm and an adaptive control loop. ATS decreases the instantaneous error to 1 μs, which ensures the proportional relationship between sync error and re-synchronization period. Moreover, it also prolongs re-synchronization periods by defining the cluster global time in an average-style. Adaptive-ATS can respond to the user error bound change quickly and accurately. It is assured of finding the optimal re-synchronization period in 1–2 steps, and so is faster than the traditional multiplicative iterative methods. In the future, we would like to introduce the linear regression estimator into the proposed protocol to reduce frequency skew and realize multi-hop time synchronization.

References

[1] L. Girod and D. Estrin, "Robust range estimation using acoustic and multimodal sensing," Proc. 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1312–1320, Maui, Hawaii, USA, Oct.-Nov. 2001.

[2] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," Proc. 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), pp.1567–1576, New York, NY, USA, June 2002.

[3] M. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," Proc. 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), pp.1740–1750, Hong Kong, March 2004.

[4] R. Gupta and S. Das, "Tracking moving targets in a smart sensor network," Proc. 58th IEEE Vehicular Technology Conference (VTC 2003-Fall), pp.3035–3039, Orlando, Florida, CA, USA, Oct. 2003.

[5] K. Römer, "Temporal message ordering in wireless sensor networks," Proc. 2nd IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2003), pp.131–142, Mahdia, Tunisia, June 2003.

[6] W. Yuan, S. Krishnamurthy, and S. Tripathi, "Synchronization of multiple levels of data fusion in wireless sensor networks," Proc. 2003 IEEE Global Telecommunications Conference (GLOBECOM 2003), pp.221–225, San Francisco, CA, USA, Dec. 2003.

[7] S. Ping, "Delay measurement time synchronization for wireless sensor networks," Tech. Rep. 1, Intel Research, IBR-TR-03-013, June 2003.

[8] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," Proc. 1st International Conference on Embedded Networked Sensor Systems (SenSys 2003), pp.138–149, Los Angeles, CA, USA, Nov. 2003.

[9] 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, Winter, 2002.

[10] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," Proc. 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004), pp.39–49, Baltimore, MD, USA, Nov. 2004.

[11] K. Noh, Q. Chaudhari, E. Serpedin, and B. Suter, "Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks," IEEE Trans. Commun., vol.55, no.4, pp.766–777, April 2007.

[12] Q. Chaudhari, E. Serpedin, and K. Qaraqe, "On maximum likelihood estimation of clock offset and skew in networks with exponential delays," IEEE Trans. Signal Process., vol.56, no.4, pp.1685–1697, April 2008.

[13] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. Srivastava, and D. Ganesan, "Estimating clock uncertainty for efficient duty-cycling in sensor networks," IEEE/ACM Trans. Netw., vol.17, no.3, pp.843–856, June 2009.

[14] D. Macii and D. Petri, "An adaptive-rate time synchronization protocol for wireless sensor networks," Proc. 2007 IEEE Instrumentation and Measurement Technology Conference Proc. (IMTC 2007), pp.1–6, Warsaw, Porland, May 2007.

[15] L. Li, Y. Liu, H. Yang, and H. Wang, "A precision adaptive average time synchronization protocol in wireless sensor networks," Proc. 2008 IEEE International Conference on Information and Automation (ICIA 2008), pp.65–70, Changsha, Hunan, China, June 2008.

[16] H. Kopetz and W. Ochsenreiter, "Clock synchronization in distributed real-time systems," IEEE Trans. Comput., vol.C-36, no.8, pp.933–940, Aug. 1987.

[17] M. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," Proc. 2003 IEEE Wireless Communications and Networking (WCNC 2003), pp.1266–1273, New Orleans, LA, USA, March 2003.

[18] W. Su and I. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," IEEE/ACM Trans. Netw., vol.13, no.2, pp.384–397, April 2005.

[19] K. Römer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks," in Handbook of Sensor Networks: Algorithms and Architectures, pp.199–237, John Wiley &

Sons, 2005.

[20] K. Römer, "Time synchronization in ad hoc networks," Proc. 2nd ACM international symposium on Mobile ad hoc networking and computing, pp.173–182, Long Beach, CA, USA, Oct. 2001.

[21] D. Cox, E. Jovanov, and A. Milenkovic, "Time synchronization for zigbee networks," Proc. 37th Southeastern Symposium on System Theory (SSST 2005), pp.135–138, Tuskegee, AL, USA, March 2005.

[22] D.S. Long and X. Tao, "Cluster-based power efficient time synchronization in wireless sensor networks," Proc. 2006 IEEE International Conference on Electro/information Technology, pp.147–151, East Lansing, MI, USA, May 2006.

[23] IEEE Standard 802.15.4-2003, http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

[24] MicaZ datasheet, 2004, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf

[25] K. Yao, R. Hudson, C. Reed, D. Chen, and F. Lorenzelli, "Blind beamforming on a randomly distributed sensor array system," IEEE J. Sel. Areas Commun., vol.16, no.8, pp.1555–1567, Oct. 1998.

Li Li was born in Jiangxi Province, P.R. China. She received B.S. degree from the Department of Electronic and information Engineering at Huazhong University of Science and Technology in 2003. She is now a Ph.D. student in Circuits and Systems Division in Electronic Engineering Department, Tsinghua University. Her research interests include time synchronization and distributed estimation in wireless sensor networks.

Yongpan Liu was born in Henan Province, P.R. China. He received his B.S., M.S. and Ph.D. degrees from Electronic Engineering Department, Tsinghua University in 1999, 2002, and 2007. He worked as a research fellow in Tsinghua University from 2002 to 2004. Since 2007, he became an assistant professor in Electronic Engineering Department, Tsinghua University. He has published over 30 peer-reviewed conference and journal papers, supported by NSFC, 863, 973 Program. His main research interests include embedded systems, power-aware architecture and VLSI design and electronic design automation. Specifically, his projects consist of ultra-low power wireless sensor network and heterogeneous MPSoCs for software defined radio. He is an IEEE member and served as a reviewer and TPC of several IEEE conferences and TVLSI.

Huazhong Yang was born in Sichuan Province, P.R. China, on Aug. 18, 1967. He received B.S., M.S., and Ph.D. Degrees in Electronic Engineering from Tsinghua University, Beijing, in 1989, 1993, and 1998, respectively. Now, he is a Professor and Head of the Institute of Circuits and Systems in Electronic Engineering Department, Tsinghua University, Beijing. His research interests include CMOS radio-frequency integrated circuits, VLSI system structure for digital communications and media processing, wireless sensor network, low-voltage and low-power circuits, and computer-aided design methodologies for system integration. He has authored and co-authored over 30 patents, 7 books, and over 200 journal and conference papers. He was granted National Palmary Young Researcher Fund of China in 2000 and the Special Governmental Subsidy from the Chinese State Council in 2007.

Hui Wang was born in Chongqin, P.R. China, on Feb. 20, 1947. She received B.S. degree in radio technology from the Radio Department, Tsinghua University, Beijing, in 1970. In 1970, she joined the Department of Electronic Engineering, Tsinghua University, where she is now a Full Professor. She was the vice Chairman of the Department of Electronic Engineering, Tsinghua University, from 1996 to 1998. Now, Prof. Wang has been appointed as the deputy chief of the Educational Administration Section, Tsinghua University. Prof. Wang has authored and co-authored over 50 technical papers and 5 books, among them a textbook Methods for Computer-Aided Circuit Analysis and Design.