An implementation of IEEE 1588 protocol for IEEE 802.11 WLAN

Article *in* Wireless Networks · August 2015
DOI: 10.1007/s11276-015-0898-z

5 authors, including:

Wu Chen
Northwestern Polytechnical University
11 PUBLICATIONS **36** CITATIONS

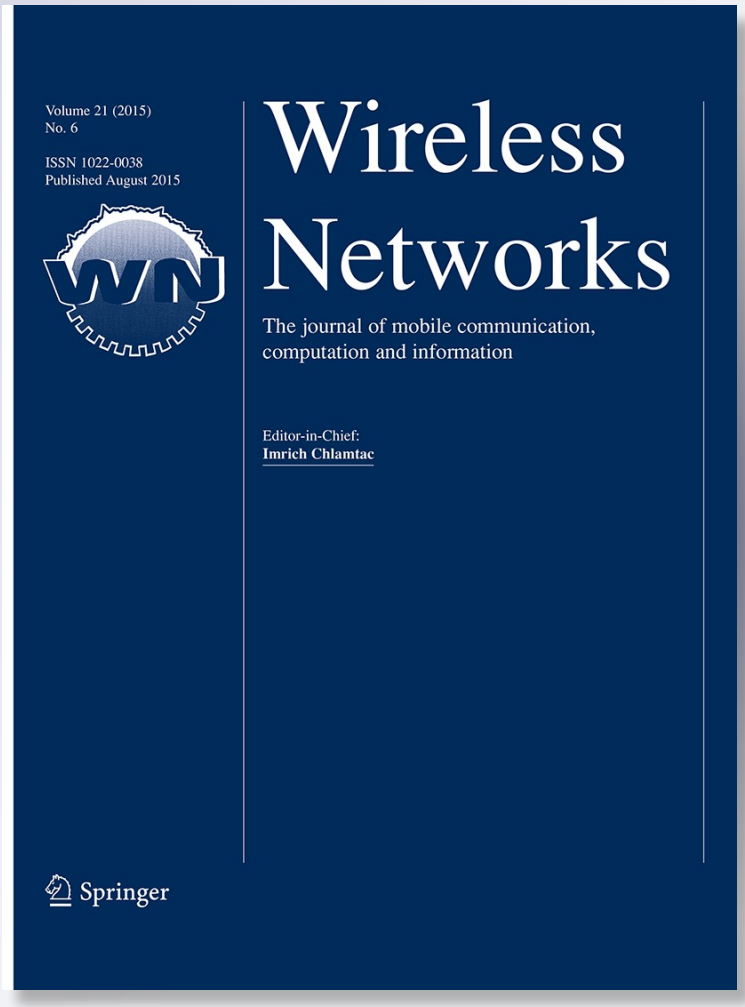
An implementation of IEEE 1588 protocol for IEEE 802.11 WLAN

Wu Chen, Jianhua Sun, Lu Zhang, Xiang Liu & Liang Hong

Wireless Networks
The Journal of Mobile Communication, Computation and Information

ISSN 1022-0038
Volume 21
Number 6

Wireless Netw (2015) 21:2069-2085
DOI 10.1007/s11276-015-0898-z



Springer

Wireless Netw (2015) 21:2069–2085
DOI 10.1007/s11276-015-0898-z

An implementation of IEEE 1588 protocol for IEEE 802.11 WLAN

Wu Chen · Jianhua Sun · Lu Zhang · Xiang Liu · Liang Hong

Published online: 31 January 2015

Abstract Clock synchronization is one of the enabling technologies for Wireless Local Area Networks (WLAN). It is crucial to perform applications such as data fusion, location detection and energy conservation. IEEE 1588 Precision Time Protocol (PTP) is a widely used clock synchronization protocol, but its accuracy is affected by bidirectional asymmetric delays in WLAN. A detailed analysis of the generation mechanism and statistical properties of the bidirectional asymmetric delays in IEEE 802.11 WLAN is conducted firstly. Then, a Kalman filter is designed for delay filtering. And based on the Kalman filter, a clock servo system is proposed using pure software-based implementation of PTP for IEEE 802.11 WLAN. Finally, the effectiveness of the implementation is verified by experiments. Experimental results show that the implementation has the virtues of high synchronization accuracy, short convergence time and small deviation error.

Keywords Synchronization · WLAN · IEEE 1588 · Bidirectional asymmetric delays · Kalman filter · Clock servo

1 Introduction

Clock synchronization is one of the key technologies for distributed systems. The purpose of clock synchronization is to maintain a consistent global clock in a distributed system and make the information, events, and time related behavior of each node have a global consensus explanation [26]. Clock synchronization is important for a distributed network measurement and control system because performance of the system largely depends on the clock synchronization accuracy [11] of the network. In order to meet the requirement of high precision synchronization, IEEE published IEEE 1588 standards [1, 3, 24] in 2002 and 2008 respectively. As a traditional distributed network measurement and control system, WLAN is widely used. And clock synchronization is a critical piece of infrastructure for WLAN in various fields.

There are some key problems that need to be resolved in clock synchronization.

1. *Accuracy* It is a core problem of clock synchronization technology. Commonly, the clock deviation among nodes should less than a millisecond.
2. *Complexity* As the node processing ability and battery energy are very limited in WLAN, clock synchronization technology cannot adopt complex algorithms even if they can achieve higher accuracy.
3. *Scalability and robustness* Due to stringent requirements on mobility and energy consumption, nodes may frequently join or leave a network operated in ad-hoc mode resulting in a varying number of active nodes at any given point in time. A clock synchronization algorithm suited for WLAN networks shall take this into account. If operated in harsh environments, clock synchronization shall remain operational even if nodes are partly malfunctioning.
4. *Cost and energy consumption* The size of nodes in WLAN is usually small and the energy capacity is very limited. Clock synchronization should consume low energy in order to prolong work time. Furthermore, the factor of terminal equipment cost should be taken into consideration.

W. Chen (✉) · J. Sun · L. Zhang · X. Liu · L. Hong
Department of Automation, Northwestern Polytechnical University, Xi'an 710072, China
e-mail: chenwu@nwpu.edu.cn

5. *Real time* In order to achieve good synchronous effect, periodic synchronization or event trigger synchronization can be adopted. No matter which mechanism is used, the clock synchronization solution should meet the real-time requirement.

With the rapid development of network and its wide use, clock synchronization is becoming one of the hottest issues in the WLAN. Numerous amounts of research work have been done in recent years. At present, clock synchronization algorithm in WLAN can be classified into two categories: sender-to-receiver synchronization [2, 37] and receiver-to-receiver synchronization [7]. In sender-to-receiver synchronization, a sender should be first designated or dynamically elected. The sender periodically sends packets with local timestamps. Receivers use timestamps as the reference clock and estimate transmission delays to synchronize the clock with the sender. A typical sender-to-receiver synchronization algorithm is the Timing-Sync Protocol for Sensor Networks (TPSN) [21]. IEEE 802.11 has its own clock synchronization mechanisms. It uses special beacon frames to synchronize the Time Synchronization Function (TSF) [2] counter on the wireless interfaces of the participating nodes in WLAN. Receiver-to-Receiver synchronization uses a sequence of synchronization messages from a fixed sender. All receivers can be synchronized with each other directly. This method uses the multicast characteristic of wireless channels to eliminate the processing delay, media access delay, transmission delay and receiving delay during transmissions. A typical receiver-to-receiver algorithm is the Reference Broadcast Synchronization (RBS) [19].

Currently, many proposed synchronization algorithms for WLAN focus on multi-hop environments because distributed multi-hop network is the development direction of the WLAN in distributed network measurement and control systems. These algorithms are required to collect a series of timing message transmissions, and estimate relative clock skews and offsets among nodes by removing the effects of random delays. In this regard, WLAN, Wireless Sensor Networks (WSN) and Mobile Ad Hoc Networks (MANET) face the same problem and have many disadvantages as follow:

1. So far, most of the synchronization algorithms must have a master node. Other nodes in the network regard the master node as a reference to realize clock synchronization. The master node can be specified, such as Gossiping Time Protocol (GTP) [23] and TPSN protocol [21, 25]. Also, it can be dynamically selected, such as PTP [42], Recursive Time Synchronization Protocol (RTSP) [4, 5], Flooding Time Synchronization Protocol (FTSP) [22, 30, 41] and Level Synchronization by Sender, Adjuster and Receiver (LESSAR) [40]. Some synchronization protocols based on TSF also depend on a master node [9, 27]. When using a master node, there is an obvious disadvantage. If the master node breaks down, designating or electing a new master node will cause a sharp decline in performance and affect the normal operation of the protocol. And the master node based algorithms rely on a spanning tree or clustered-based structure. They suffer large overhead in building and maintaining the tree or cluster structure, and are vulnerable to sudden node failures [17].
2. Currently, clock synchronization algorithms are still not perfect and mainly suitable for one-hop environment. In multiple-hop environment, the error will cumulate with the increasing of the hop count [5, 9, 15, 34]. Meanwhile, communication cost will increase and influence the performance of the protocol.
3. Clock synchronization algorithms should be able to adapt to the dynamic topology. However, current studies mostly focus on the static topology and only consider contingent topology changes caused by node failures and new node joining. Current clock synchronization cannot support network partition and combination well [39]. When a network is divided or combined, the clock synchronization error will increase. The simulation results of literature [33] illustrate the problem. Based on the Unifying Slot Assignment Protocol (USAP), Rockwell Collins Company develops a combination method of divided networks [35]. However, this method is based on multi-channel and nodes need to switch to a specific synchronous channel for receiving synchronization information. This method cannot be applied in single channel WLAN. In addition, Wolf et al. have done research on the combination of dynamic TDMA networks and developed the Asymmetric Partition Synchronization Protocol (APSP) [38]. APSP is based on the assumption that the physical layer can provide a special interruption for the Media Access Control (MAC) layer in order to detect non-synchronization. However, most physical layer components cannot realize the detection. Therefore, the above two methods cannot be applied widely at this time.
4. By now, many clock synchronization solutions are conducted under the assumption that the network topology does not change or rely on special topologies such as spanning tree and clustered-based structure. Such solutions are vulnerable to node failures.
5. The fundamental approach to clock synchronization is getting the clock offset by transmitting a series of timing messages. Random delays make the offset deviate from the true value. In this sense, clock synchronization can be regarded as the process of

random delays. The main solution to deal with random delays is the statistical estimation technique. Many estimators are designed for particular distribution of delay. The performance may vary a lot for different delay models. There is a need for estimation techniques which are robust to different delay distribution.

As a conclusion, though there is a large amount of research on clock synchronization for WLAN, all of this research remains at the exploration stage. The research to date has not produced any standard that has been adopted widely. As a kind of high precision clock synchronization protocol, IEEE 1588 is an international standard and widely used in distributed measurement and control systems. Applying PTP to WLAN can provide clock information with high accuracy and promote the applications of WLAN. In addition to WLAN, other wireless measurement and control systems also have urgent need in clock synchronization. However, PTP is mainly used in wired networks, for example, IEEE 802.3, and the application to WLAN is still in the stage of exploration. A study encompassing PTP and WLAN has been done by Cooklev et al. [13]. This study does not propose actual implementation. Instead, it spotlights potential instants where hardware timestamping can be done to provide synchronization accuracy in the range of nanoseconds. Bang et al. have provided PTP implementation over IEEE 802.11n for multimedia service using software support and statistical filtering [6]. However, the final synchronization jitter is 400 μs. Mahmood et al. [28] proposes an approach for wireless clock synchronization based on an ordinary PTP and software timestamping at driver level. PTP assumes that forward and reserve links have the same delay, but the transmission delay for uplink and downlink may be different in a single round-trip. This is the main limitation of PTP applying in WLAN. In WLAN, clock deviation obtained by the existing synchronization mechanism has large errors and it will seriously influence the accuracy of clock synchronization.

As IEEE 802.11 is the first choice of WLAN, the research results for IEEE 802.11 can be widely used. In this paper, we study how to effectively apply PTP to IEEE 802.11 WLAN. A delay filtering algorithm based on Kalman filter is designed. And based on the delay filtering algorithm, a clock servo system is achieved to realize pure software-based implementation of PTP for IEEE 802.11 WLAN.

The rest of this paper is organized as follows. The next section introduces the principle of PTP. Section 3 describes the 802.11 backoff mechanism which is the main cause of asymmetric delays. The analysis of asymmetric delays is provided in Sect. 4, and the delay filter algorithm is proposed in Sect. 5. Clock servo system design is described in Sect. 6. The experiment results are presented in Sect. 7. Finally, Sect. 8 draws conclusions.

2 IEEE 1588 protocol theory

IEEE 1588 defines an accurate clock synchronization protocol for network measurement and control systems. The basic principle of PTP is that a precise time source is used to periodically synchronize and revise all the clocks in the network through message exchange.

2.1 Synchronization principle of PTP

PTP is the core of IEEE 1588. By periodically exchanging timestamps, PTP can realize syntonization and synchronization functions completely. The former means setting clocks to read the same time at a given instant, the latter means setting clocks to the same frequency, regardless of the time they display. According to the communication relationship between clocks, clock are divided into master clocks and slave clocks in PTP. There is only one master clock in a PTP network, and all rest clocks are slaves. The master clock is the reference clock in the whole network, and it periodically multicasts the local clock information. Slave clocks receive the master clock information and synchronize to the master by computation. PTP defines four kinds of packets, namely, Sync, Delay_Req, Follow_Up, and Delay_Resp, to generate and communicate the timing information needed to synchronize. The synchronization process is divided into two steps: the first is offset measurement, and the next is delay measurement.

2.1.1 Offset measurement

Offset measurement means measuring the time deviation between the master and the slave. During the offset measurement, the master periodically (every 2 s in common) multicasts special synchronization packet Sync to slaves. The master node needs to record the exact sending time of Sync and put it in the Follow_Up packet. Figure 1 illustrates the process. It is necessary to note that the time scale in the figure does not represent real time.

In order to measure the offset, the master first sends a Sync packet and marks the exact sending time t_1. The slave receives the Sync packet and marks the receiving time t_2. In the process of calculating the deviation, the slave needs to use t_1. So, the master needs to send a Follow_Up packet carrying timestamp t_1 to the slave after the Sync packet. After obtaining timestamps t_1 and t_2, the slave can estimate the clock offset by the following equations.

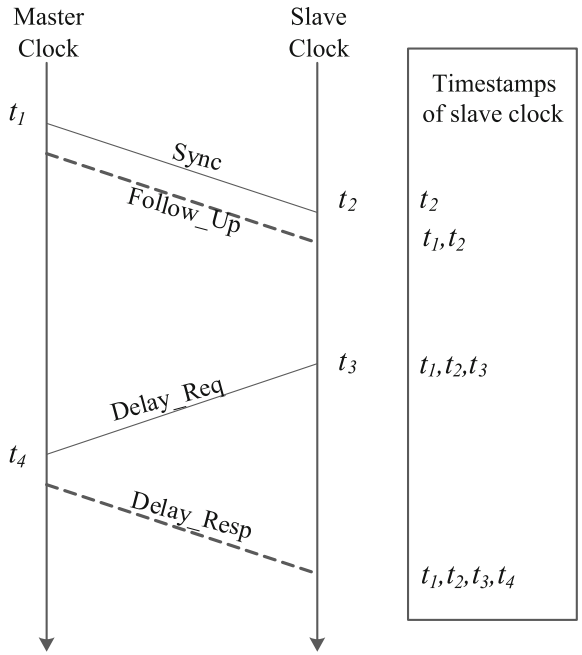


Fig. 1 Offset measurement between master and slave clock

$$t_2 - t_1 = offset + delay_1 \quad (1)$$

$delay_1$ represents the packet transmission delay from the master to the slave.

2.1.2 Delay measurement

Delay measurement is used to calculate the packet transmission delay between the master and the slave. The slave sends a Delay_Req packet to the master periodically and marks the sending time t_3. The default period of each slave is set between 4 and 60 s, and the sending time is slightly varied from node to node to avoid packet bursts. The transmission intervals shall be taken from a uniform random distribution with a minimum width of $2^{logMinDelayReqInterval}$ seconds [2]. The master receives Delay_Req and marks the receiving time t_4. Then, a Delay_Resp packet with timestamp t_4 will be sent to the slave. After obtaining t_3 and t_4, the slave can measure the delay as follow:

$$t_4 - t_3 = -offset + delay_2 \quad (2)$$

$delay_2$ represents the packet transmission delay from the slave to the master.

In the wired distributed network, we assume that the transmission delay is symmetrical. So,

$$delay = delay_1 = delay_2 \quad (3)$$

Combined with (1), (2), and (3), we get:

$$offset = [(t_2 - t_1) - (t_4 - t_3)]/2 \quad (4)$$

$$delay = [(t_2 - t_1) + (t_4 - t_3)]/2 \quad (5)$$

2.2 PTP analysis

From the workflow of PTP mentioned above, we can know that the prerequisite of PTP realizing precise clock synchronization is that the bidirectional delay between the master and the slave is symmetric. (It is necessary to note that IEEE 1588 can support the asymmetric bidirectional delay if the bidirectional delay is basically stable and can be measured. However, the measuring method of bidirectional delays is not mentioned in IEEE 1588.) In ordinary Ethernet, the prerequisite of PTP can be easily satisfied. However, it is hard to realize in WLAN because of the existence of asymmetric links. The bidirectional delay in WLAN is asymmetric and has a large variance. That is the main limitation of PTP applying to WLAN. And the asymmetric delay is mainly due to the 802.11 backoff mechanism. So we need to analyze the 802.11 backoff algorithm.

3 IEEE 802.11 backoff algorithm analysis

As mentioned above, the asymmetric bidirectional delay with a large variance is the main obstacle of PTP applying to WLAN. Here, we take the most popular WLAN protocol IEEE 802.11 protocol as an example. The main cause of the asymmetric bidirectional delay with a large variance in IEEE 802.11 is the backoff algorithm. So we need to analyze the backoff algorithm of IEEE 802.11 and find out corresponding solutions.

IEEE 802.11 mainly adopts the Distributed Coordinated Function (DCF) mechanism [2] which is similar to the competition mechanism in Ethernet. Before transmitting packets, wireless equipment needs to ensure that the channel is free. If it is free for a guard period known as the Distributed Inter Frame Space (DIFS), the node will immediately send its packets. Otherwise, the node will listen to the channel. After the channel is free, waiting for a DIFS time, the Contention Window (CW) stage will begin. CW can be divided into several slots. We use T_{slot} to represent the time length of each slot. And the initial slot number is called Backoff Initial (BI). BI determines the size of CW. Wireless equipment randomly selects one slot from 0 to BI, and starts a counter. The counter is started at the selected slot and is decremented at the end of every T_{slot} cycle. When the counter reaches zero, data will be transmitted if the channel is still free. If there are several competing nodes, some of them will select slots with small numbers and the others will select slots with large ones. The node which selects slot with a large number can detect the channel being occupied before its counter reaches zero. At this moment, the node suspends the current counter and will restart it as long as the channel is determined to free

without interruption for a period of time equal to DIFS. If data transmission fails, the size of CW will be stretched. The size of CW is $2^n - 1$ (such as 31, 63, and 127). CW stretching means $n + 1$. This is the Binary Exponential Backoff (BEB) algorithm. The size of CW is limited by the physical layer and generally is set to 1,023.

According to the above analysis, the asymmetric bidirectional delay with a large variance of IEEE 802.11 results from three aspects. The first one is the delay jitter caused by random slot selection. It is a uniform distributed random variable (0–CW) and commonly within several hundreds of microseconds. The second one is the uncertain delay due to suspended time caused by channel competition. This uncertain delay varies with the payload and is commonly above millisecond level. The third one is the bidirectional asymmetric delay jitter caused by BI changes in BEB algorithm, also commonly above millisecond level. Therefore, when directly adopting the timestamping above the MAC layer, it is very difficult to get highly precise clock synchronization. In literature [13], Cooklev has discussed the possibility of PTP applying to WLAN by using hardware timestamping. Cooklev has studied the delay jitter characteristics of different physical layer signals in IEEE 802.11 by testing on wireless network adaptor 340 of Cisco AIRONET series. Through a series of experiments, Cooklev has chosen the rising edge of TX_RDY and MD_RDY as the trigger source of timestamps. And the delay jitter can be controlled in 200 ns and thus can meet the requirement of IEEE 1588 application in measurement and control areas. Similar work has been done in [8, 10, 20].

Using hardware timestamping of the physical layer can achieve high precision clock synchronization [29]. However, using hardware timestamping to realize PTP in WLAN still has some problems. Firstly, GS1010, 88W8897 and AR9287 are currently the only chipsets of IEEE 802.11 that support physical layer timestamps as far as we know. GS1010 is already out of production and 88W8897 which supports 802.11v is still not available. TP-Link TL-WN822Nv2 with AR9287 is on sale, but it is a Universal Serial Bus (USB) adapter. Secondly, using hardware timestamping needs the support of chipsets, and this will increase the hardware cost of implementation and limit its scope of application. Software timestamping can be implemented at the application layer or MAC layer. But in order to put timestamps in the payload, the driver should be modified. Therefore, in this paper, we consider using software timestamping of the application layer to realize PTP in IEEE 802.11 WLAN. In order to reach this goal, we need to solve the problem of the asymmetric bidirectional delay and overlarge delay jitter. We will give the solution in the following sections.

4 Delay analysis of clock synchronization

The basic method of clock synchronization is to obtain the clock deviation between different nodes through the exchange of timestamps. The clock deviation includes two parts: one part is the real clock deviation and the other part is the delay resulted from transmission of timestamp packets. The error in the clock synchronization algorithm is mainly caused by the delay. Here we consider realizing PTP by pure software, so we take timestamping of the application layer as an example. As shown in Fig. 2, the delay is composed of the following six kinds of delays.

1. *Sending process delay* It represents the sending delay from the application layer to the MAC layer, that is, from the time of data generation in the application layer to the package time in the MAC layer. It is influenced by the operation system and the current load of the processor.
2. *Channel access delay* It represents the time that the MAC protocol waits for the free channel. It depends on the MAC protocol and is influenced by network traffic.
3. *Emission delay* It represents the time of transmitting frames in the physic layer and is influenced by frame length and the transmission rate of the physical layer.
4. *Propagation delay* It represents the time of data transmission in channels. As the distance between nodes in WLAN is commonly within 100 meters, the propagation delay is very small and can be ignored. However, if the distance between nodes is very large, the delay must be taken into consideration.
5. *Receiving delay* It represents the full time of the physical layer receiving a complete frame and commonly has overlap with the emission delay.
6. *Receiving process delay* It represents the time of the node processing the frame. It is the time delay of decapsulating the MAC layer package and extracting the data to the corresponding process in the application layer. Its statistical properties are similar to those of the sending process delay.

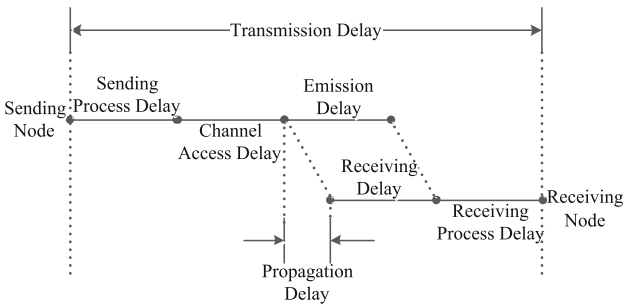


Fig. 2 Composition of transmission delay over a wireless channel

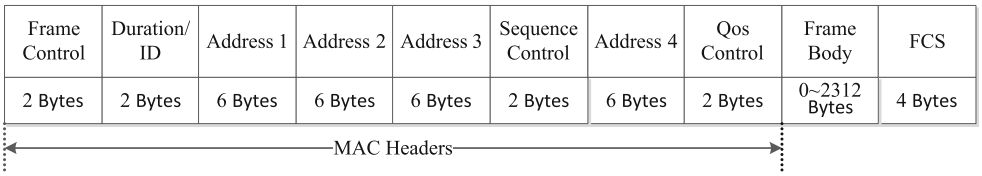


Fig. 3 MAC frame format of IEEE 802.11

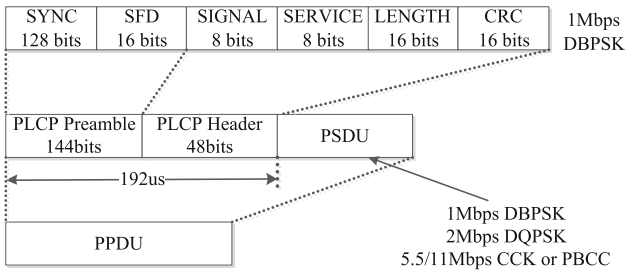


Fig. 4 Physical layer data frame structure

The transmission delay D from the sender to the receiver is a random variable. The composition of D can be described as follow:

$$D = D_{send} + D_{MAC} + D_{trans} + D_{radio} + D_{recv} \quad (6)$$

The delay estimation process is described in details as follow.

4.1 Sending process delay/receiving process delay

The sending process delay and receiving process delay are caused by the operating system and can be estimated together. We take Linux operating system as an example. With the help of [12] and [16], we can get that the total delay is <15 μs in most situations. As the goal of this paper is to achieve clock synchronization with accuracy of 100 μs, and the Kalman filter which we proposed operates at a level above the operating system, so these parts of delays can be filtered out and be negligible.

4.2 Channel access delay

The channel access delay D_{MAC} represents the waiting time from the moment that frame is completely generated to the moment of accessing the channel. In IEEE 802.11 WLAN, D_{MAC} is determined by the CSMA/CA mechanism as mentioned in Sect. 3. As described in Sect. 3, D_{MAC} is equal to the sum of a DIFS time D_{DIFS}, backoff time $D_{backoff}$ and collision time $D_{collision}$. The collision time $D_{collision}$ varies with the channel competition. The backoff time $D_{backoff}$ can be calculated as follows according to the BEB algorithm:

$$D_{backoff} = Int[CW \times Random()] \times T_{slot} \quad (7)$$

$Int(x)$ means an integer rounding function, and rounds downwards to the nearest integer of x. $Random()$ means a random number generator and generates random numbers between 0 and 1. CW means an integer between CW_{min} and CW_{max}. It is a parameter of competition window and determined by the BEB algorithm.

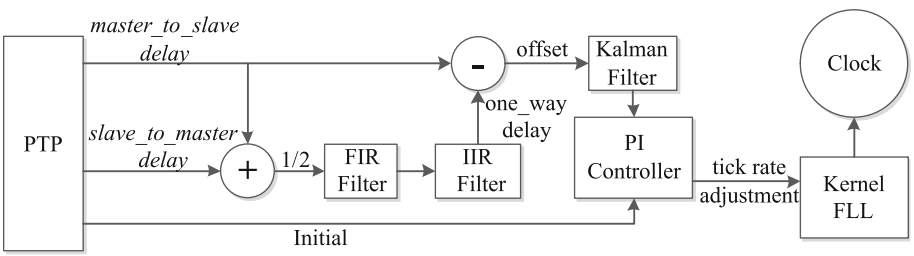
From (7), we can calculate the channel access delay as follows:

$$D_{MAC} = D_{DIFS} + D_{backoff} + D_{collision} \quad (8)$$

Table 1 Experimental conditions

Equipment	Number	Configuration
PC	3	Linux kernel 2.6.38 Intel Pentium E6700@3.20GHz, 2GB DDR3 SDRAM
Wireless LAN card	3	EW-7318Usg, 802.11b/g, for PC platform
Embedded development board	3	Linux kernel 2.6.38 S3C6410A ARM1176JZF-S@533MHz, 256M DDR RAM
Wireless LAN card	3	Mavell 88W8686, SDIO WIFI, 802.11b/g, for embedded platform
RS-485	3	Interface Converter with RS-232

Fig. 5 Clock servo system diagram



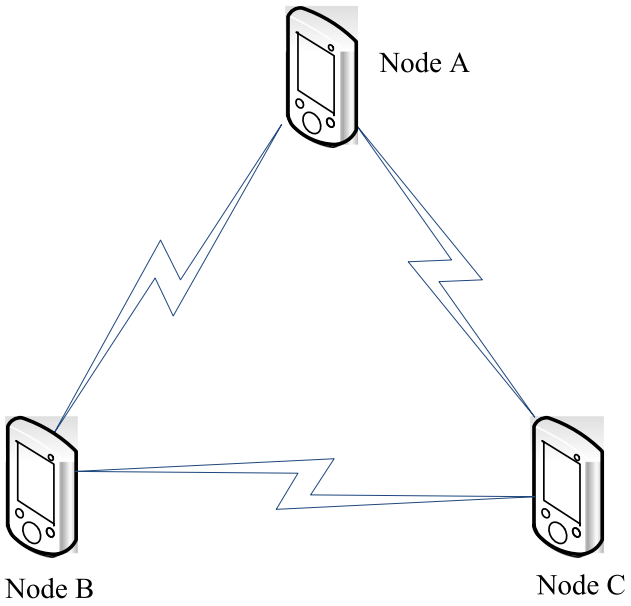


Fig. 6 Experiment setup

4.3 Propagation delay

The propagation delay D_{radio} is very small, and it can also be ignored.

4.4 Emission delay/receiving delay

The emission delay D_{trans} refers to the time of the physical layer sending data frames in the wireless channel after accessing the channel. The receiving delay D_{rev} is the time that physical layer receives the complete data packet and often overlaps with the emission delay. The emission delay D_{trans} is related to the size of data packets and transmission rate. We get the emission delay of PTP by analyzing the frame structure of PTP in the IEEE 802.11 protocol.

According to the IEEE 802.11 standard [2], IEEE 802.11 frame structure is shown in Fig. 3. And the counting unit in the figure is byte.

As shown in Fig. 3, the length of the IEEE 802.11 MAC frame header is 36 bytes. The address 3 and address 4 are

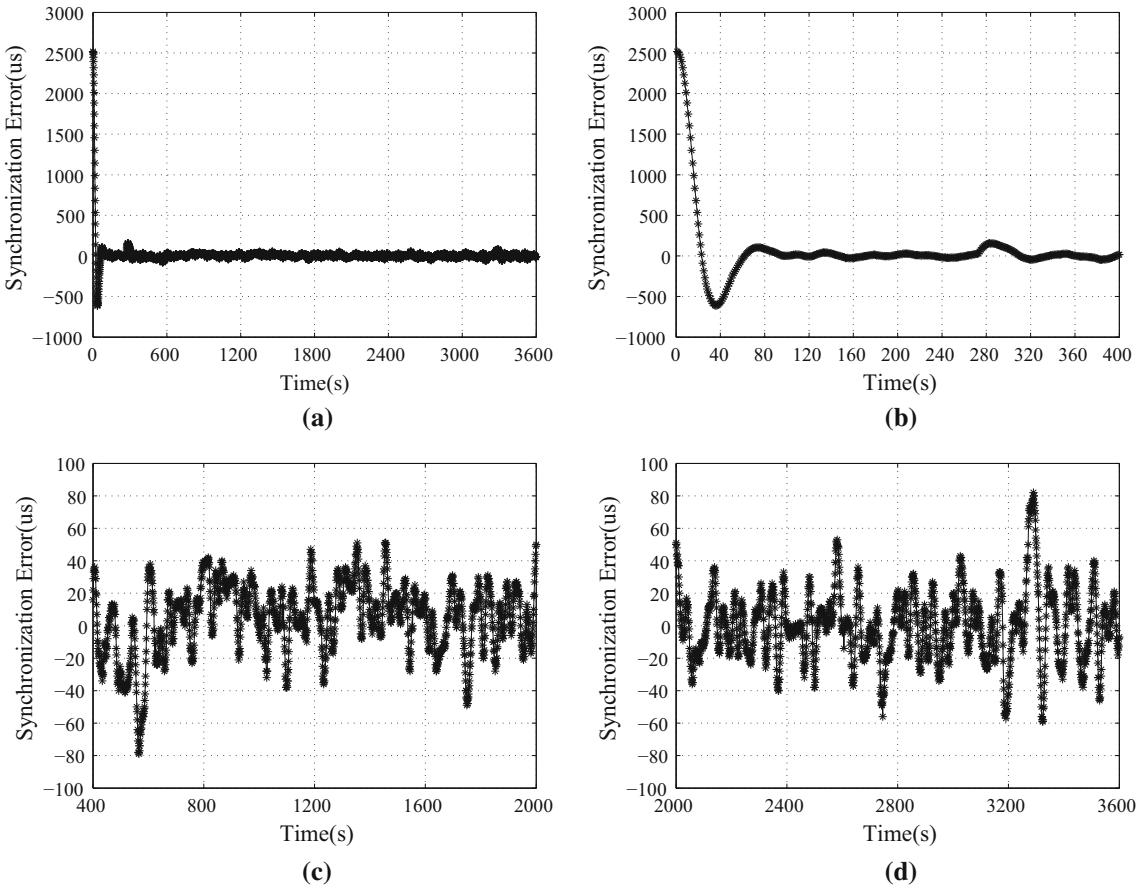


Fig. 7 Slave–slave clock offset in PC platform

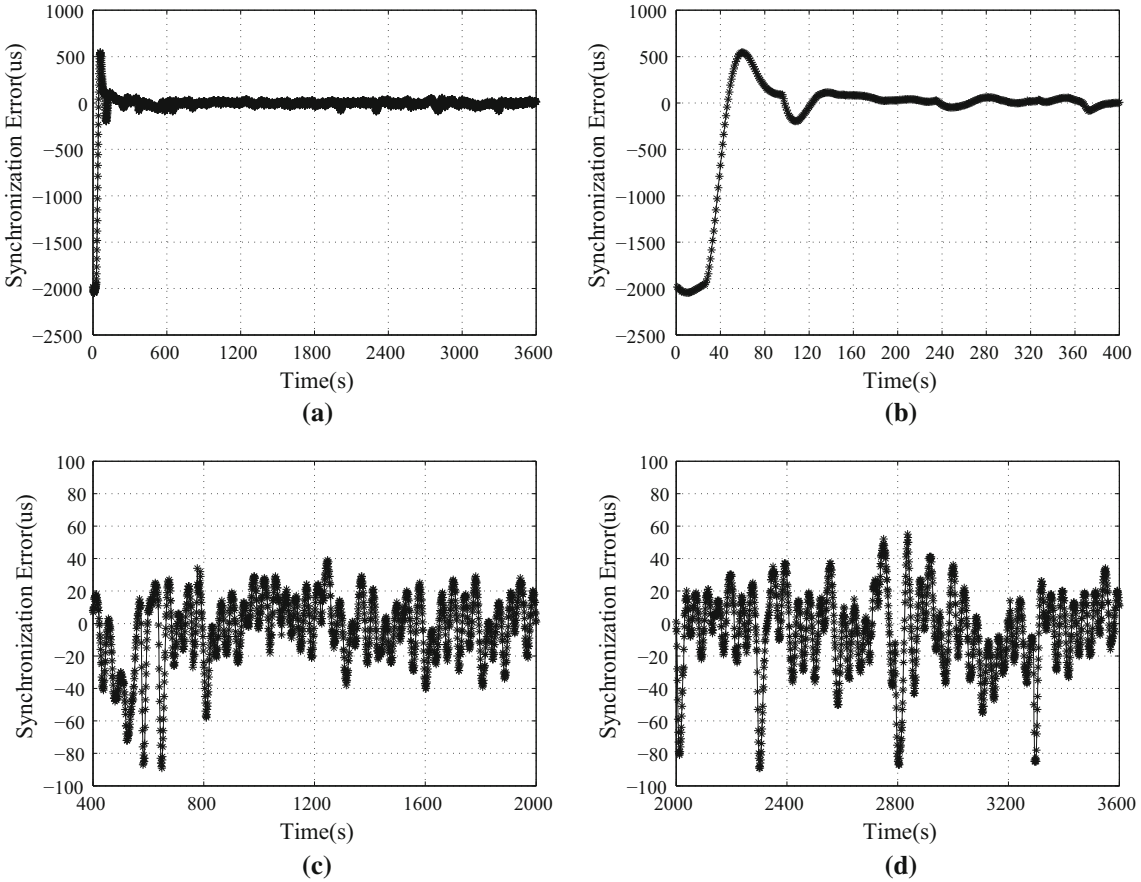


Fig. 8 Master–slave clock offset in PC platform

only used in cross BSS communication. So when using the basic business set, the length of the IEEE 802.11 MAC frame is only 28 bytes.

PTP is a type of UDP protocol and the length of PTP messages is 124 bytes at most. After adding a UDP header of 8 bytes and an IP header of 20 bytes, the packet length reaches no more than 152 bytes. Therefore, the data frame handled to the physical layer, that is, Presentation Service Data Unit (PSDU), includes 180 bytes at most.

The structure of the data frame encapsulated in the physical layer is shown in Fig. 4.

In the frame structure of the physical layer, a preamble includes a synchronization code (SYNC) of 128 bits and a start frame delimiter (SFD) of 16 bits. The Physical Layer Convergence Procedure (PLCP) header is after the preamble and includes the data transmission relevant physical parameters. The preamble and PLCP header are totally 192 bits and sent in a stable rate of 1 Mbps. And based on different business and link quality, PSDU data can be sent in the rate of 1 Mbps (Differential Binary Phase Shift Keying (DBPSK) modulation), 2 Mbps (Differential Quadrature Phase Shift Keying (DQPSK) modulation), 5.5 Mbps (Complementary Code Keying (CCK) or Packet Binary Convolutional Coding (PBCC) modulation) or 11 Mbps (CCK or PBCC modulation). The packets of PTP are multicasting and sent in the rate of 1 Mbps according to the analysis of WireShark. Therefore, we can know that emission delay/receiving delay includes the transmission delays of PSDU and the physical layer header.

$$D_{trans} = T_{slot} + PHY_{PLCP} + PHY_{PSDU} \tag{9}$$

Through the above analysis, we can conclude that the delay of PTP includes the deterministic delay and random delay. The deterministic delay is composed of the instruction cycle of the sending process delay/receiving process delay, emission/receiving delay and propagation delay. When the packet length and the distance between the nodes are known, the deterministic delay is fixed. The random delay includes two parts. One is the queuing, inquiring table and interrupts response time of the emission delay/receiving delay. And the other one is the channel access delay. The channel access delay is the main composition of the random delay. The deterministic delay can be obtained by analysis and calculation. And the main factor that

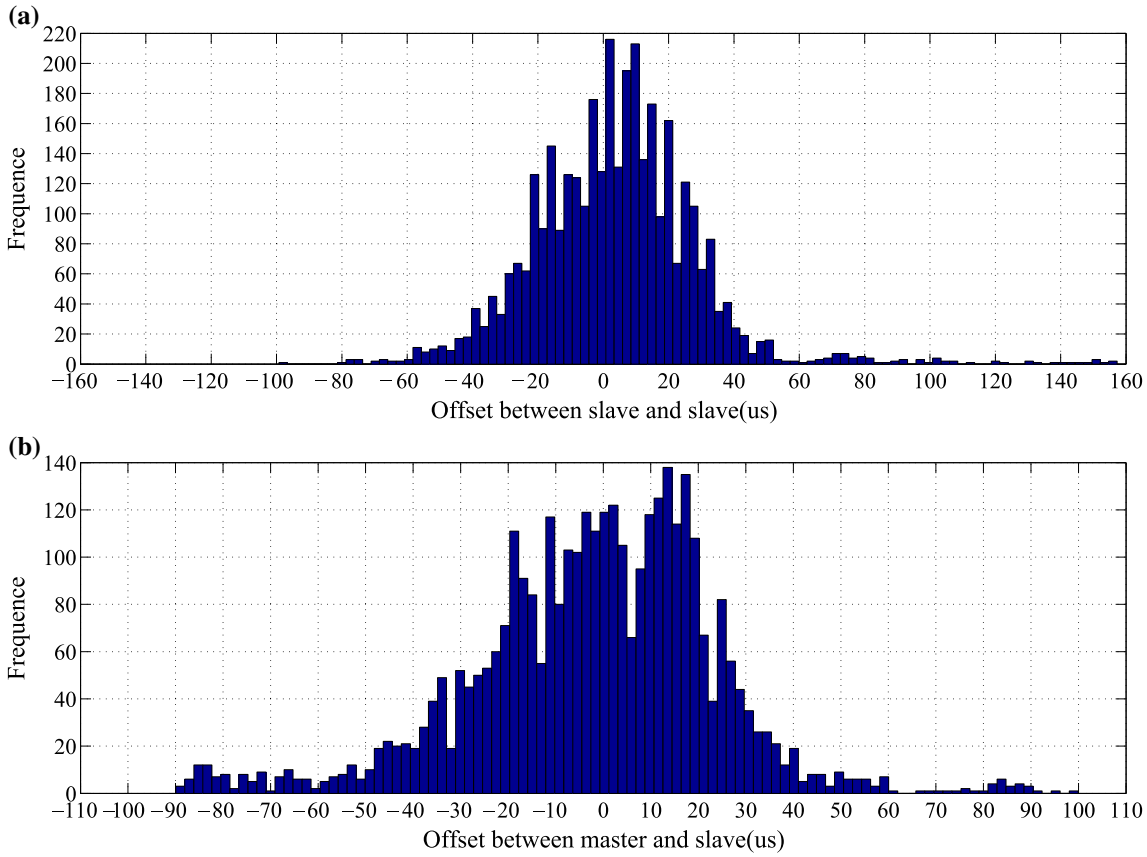


Fig. 9 Histogram of synchronization offset in PC platform

influences clock synchronization accuracy is the random delay. Therefore, how to design an appropriate filtering algorithm, reduce the influence of the random delay and achieve high precision clock synchronization is a core problem that needs to be solved in this paper. The Kalman filter is an algorithm that is used widely in parameter estimation, and has high convergence rate and high precise steady-state value. So we choose Kalman filter to achieve delay filtering.

5 Kalman filtering based delay filter algorithm

According to PTP, a high quality node is selected as the master clock in the network. All other nodes are slave nodes and need to synchronize with the master clock. Accordingly, we can assume that x is the deviation between the master and the slave. $x > 0(<0)$ represents that the slave clock leads (lags behind) the master clock.

As shown in Fig. 1, we assume that at the $k-th$ time sampling, the master node sends a Sync packet at local time t_1. And the slave node receives the data packet at local time t_2. So,

$$t_2 - t_1 = x(k) + d_1(k) \tag{10}$$

In (10), d_1 represents the packet transmission delay from the master to the slave. Then we assume that the slave node sends a Delay_Req packet at local time t_3, and the master node receives the packet at local time t_4. So,

$$t_4 - t_3 = -x(k) + d_2(k) \tag{11}$$

In (11), d_2 represents the packet transmission delay from the slave node to the master node.

Using (10) minus (11), we get

$$x(k) = \{[(t_2 - t_1) - (t_4 - t_3)] - [d_1(k) - d_2(k)]\}/2 \tag{12}$$

We assume that $\theta(k) = [(t_2 - t_1) - (t_4 - t_3)]/2$ and $\Delta d(k) = [d_1(k) - d_2(k)]/2$. And $\theta(k)$ can be measured. Therefore, the measurement equation of the clock difference between a pair of master–slave nodes is

$$x(k) = \theta(k) - \Delta d(k) \tag{13}$$

Assume that at the moment of $k-th$ time sampling, the master clock is $x_m(k)$, and the slave clock is $x_s(k)$, then

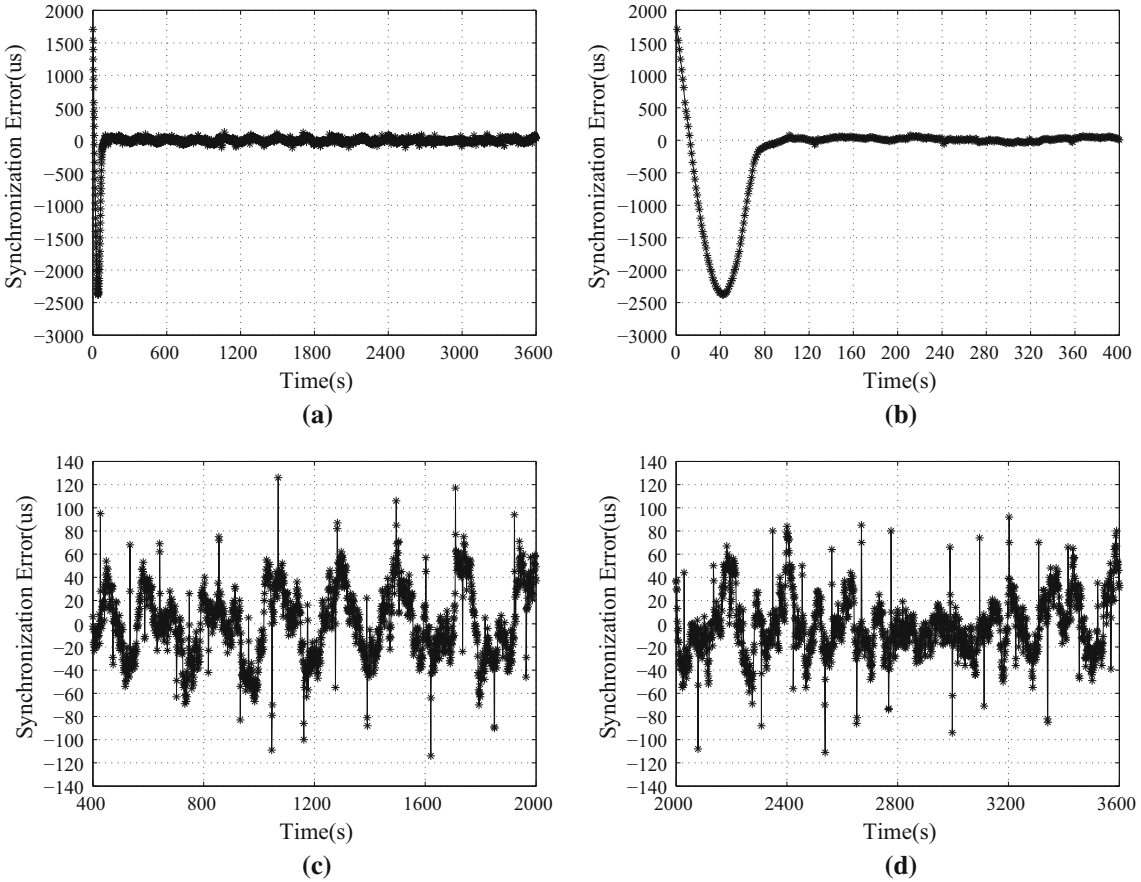


Fig. 10 Slave–slave clock offset in embedded platform

$$x(k) = x_s(k) - x_m(k) \tag{14}$$

Due to the imperfection of oscillator, a clock will drift away from the ideal time in every second. The amount of clock drifts is generally between ± 1 μs per second and ± 100 μs per second. Through large amount of statistics, the clock drift obeys normal distribution with mean 0 [32]. Since the master node and slave adopt the same clock oscillator and their work environment is almost the same, the clock offset between the master node and slave node is independent and identically distributed (i.i.d.) random variables. So, we can get

$$\begin{cases} x_s(k+1) = x_s(k) + T + \varepsilon_1, \varepsilon_1 \sim N(0, \sigma^2) \\ x_m(k+1) = x_m(k) + T + \varepsilon_2, \varepsilon_2 \sim N(0, \sigma^2) \end{cases} \tag{15}$$

In (15), T is the sampling period. According to (14) and (15), we can get

$$x(k+1) = x(k) + \varepsilon \tag{16}$$

And $\varepsilon = \varepsilon_1 - \varepsilon_2$. ε_1, ε_2 are i.i.d. random variables, so,

$$\varepsilon \sim N(0, 2\sigma^2) \tag{17}$$

So far, we can get the state equation and observation equation of implementing Kalman filter, that is,

$$\begin{cases} x(k+1) = x(k) + \varepsilon \\ \theta(k) = x(k) + \Delta d(k) \end{cases} \tag{18}$$

We have already got the statistical characteristics of ε from (18). In order to achieve the Kalman filter algorithm, we also have to know the statistical characteristics of $\Delta d(k)$.

According to the description in Sect. 4, we know that the delay between the master node and slave node includes the deterministic delay and random delay. The deterministic delay can be obtained by calculation. The random delay includes queuing delay, querying and interrupt response time of the emission/receiving delay, and channel access delay. The first two delays have been analyzed in Sect. 3. And the channel access delay is the main part of the random delay. Sakurai and Vu [31] develop a unified analytical model and obtain an explicit expression for the MAC access delay in a saturated IEEE 802.11 DCF WLAN. Simulation and experimental results show that the

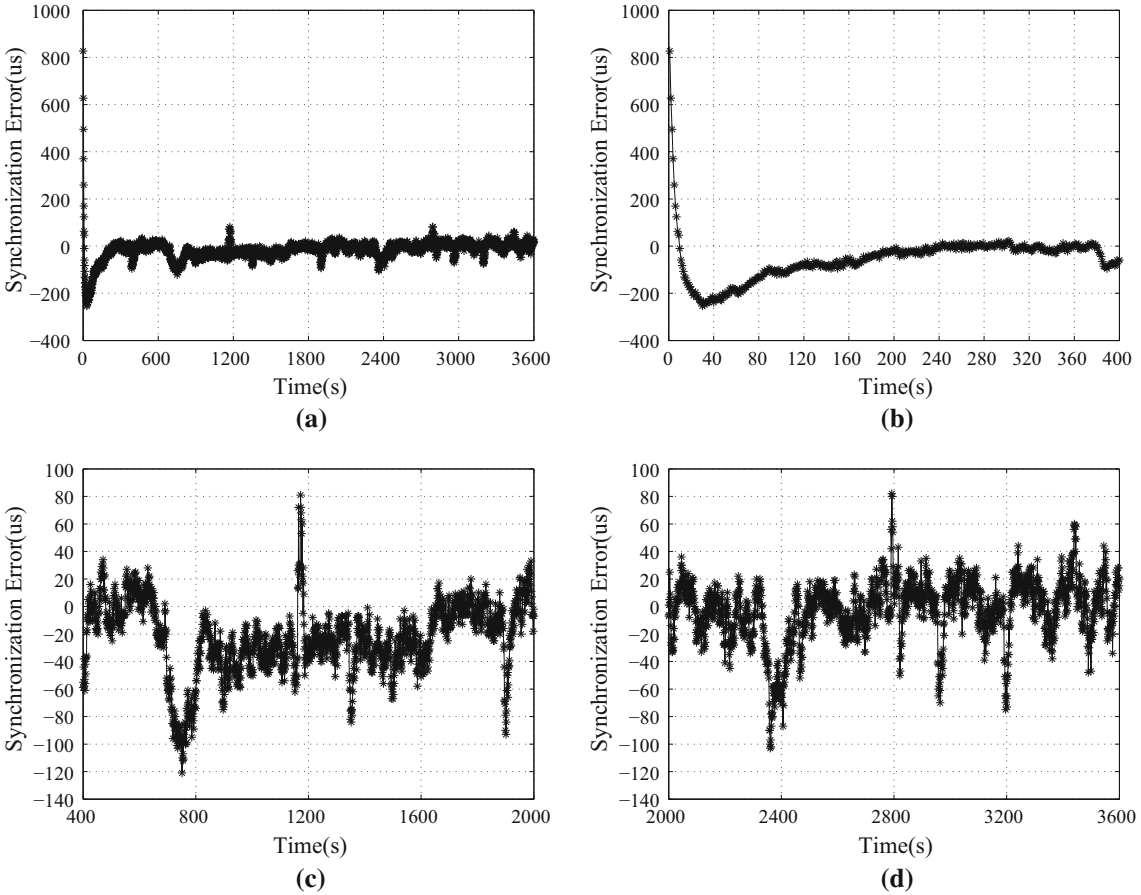


Fig. 11 Master–slave clock offset in embedded platform

mean and standard deviation of the access delay are determined when the number of network nodes and the packet size are fixed. As for PTP, the size of the clock synchronization measure packet is fixed. The number of nodes can also be obtained in a same communication subdomain. Therefore, the preceding conclusion can be applied in this study. As the magnitude larger delay caused by nodes competition or transmission failures could be identified obviously, we can filter them by range limitation. Thus, the access delay associated statistical properties from the random delay could also be obtained accurately.

According to the analysis in Sect. 4, the delay includes the deterministic delay and random delay. We can get the deterministic delay D_{def} by calculation. We can also define the mean of the access delay is D_{MACavg} and introduce a threshold value D_{thre} of the measurement error. If the delay we get satisfies $D_{delay} > D_{def} + D_{MACavg} + D_{thre}$, it will mean that there are several nodes competing or transmission failures. We regard D_{delay} in this situation as interference data and do not use it to calculate the delay. Instead, we replace it with the previous qualified sampling data by the Finite Impulse Response (FIR) filter.

We consider the queuing, querying and interrupt response time of the emission delay/receiving delay in the random delay as white noise.

So far, we have obtained the state equation, observation equation and relevant statistical characteristics which are needed in designing the Kalman filter. As for the implementation process of the Kalman filter, there are many relevant literatures and we will not describe it here anymore.

6 Clock servo system design

By adopting the Kalman filter described in Sect. 5, we can obtain relatively accurate bidirectional delays and lay a solid foundation for realizing high precision clock synchronization. In order to coordinate the local slave clock with the reference master clock time, a clock servo system is applied. Correll et al. have achieved a pure software based PTP and adopted a Proportional Integral (PI) controller based clock servo system in its implementation [14].

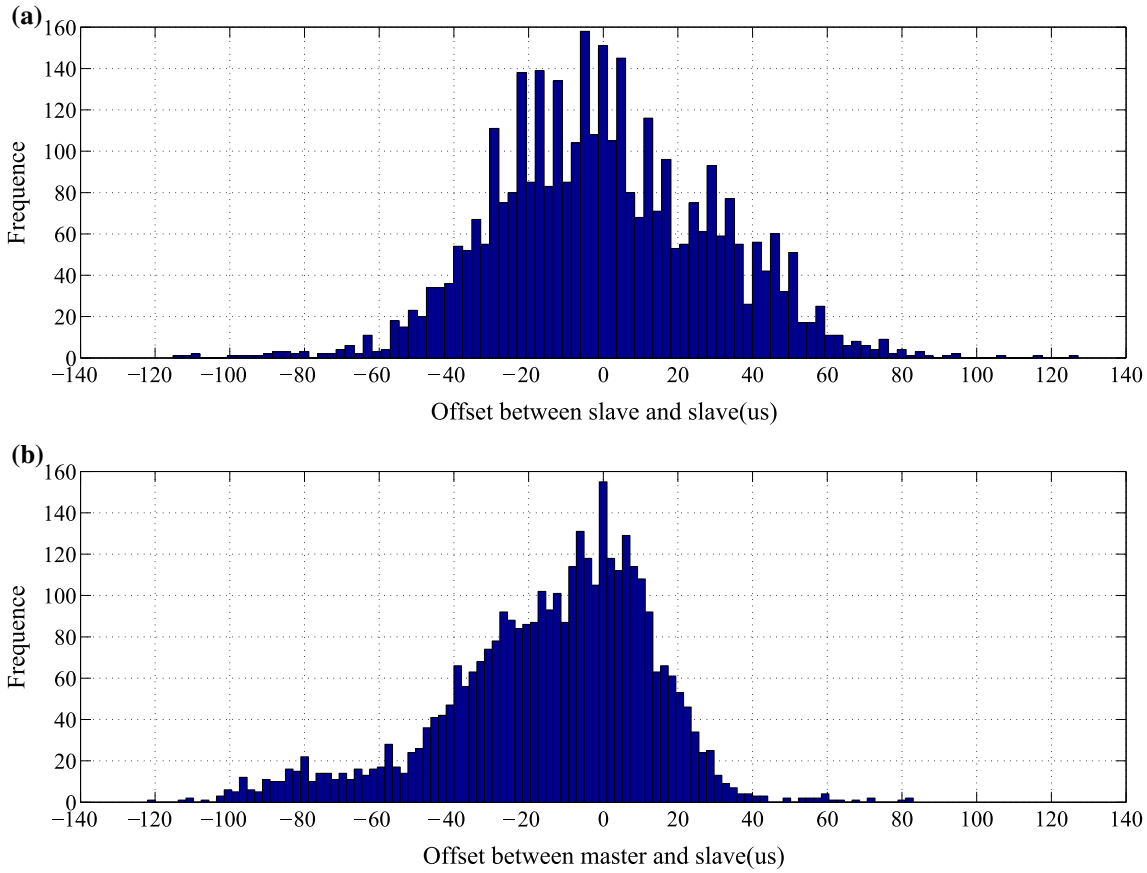


Fig. 12 Histogram of synchronization offset in embedded platform

An in-depth analysis of the clock servo system is presented in [18]. However, his protocol can only be used in wired networks. We reference and modify its PI controller design according to our research result. We use the Kalman filter to mitigate the detrimental effect of the asymmetric delay on clock coordination. The modified PI controller is shown in Fig. 5.

In Fig. 5, PTP regularly samples the communication delay of master-to-slave and slave-to-master. A FIR filer is used to mitigate input noise, especially the noise caused by the asymmetric delay. An Infinite Impulse Response (IIR) filter for one-way delays is also involved to improve the response characteristics. The Kalman filter mitigates noise of the clock offset further. The output of the PI controller is used to adjust the local clock frequency in order to realize master–slave clock synchronization. A typical PI controller is a closed-loop control system composed of a proportion term and an integration term. And the proportion term is used to eliminate input error, that is, the clock deviation between the master clock and slave clock. The integration term is used to eliminate steady-state error of the system, that is, the rate difference between the master and the slave. The PI controller cannot only reduce clock error but also adjust clock frequency. Referring to Eidson's research [18], the parameters of the PI controller are chosen by tuning.

7 Experiment results and analysis

7.1 Experiment environment and method

In order to verify the generality of the solution, we establish an experimental environment of IEEE 802.11 WLAN based on both the Personal Computer (PC) platform and the embedded platform. The experiment environment and related platform parameters are shown in Table 1. The experiment setup is shown in Fig. 6.

There are several methods to measure the accuracy of clock synchronization. The common methods are the software report, pulse per second(pps) signal comparison and clock output comparison [36]. Among them, the pps comparison is a frequently-used method in analyzing clock synchronization. This method contrasts the current clock synchronization accuracy by observing pps generated at every signal transition. The measurement method we adopted does not compare the pps signals of different

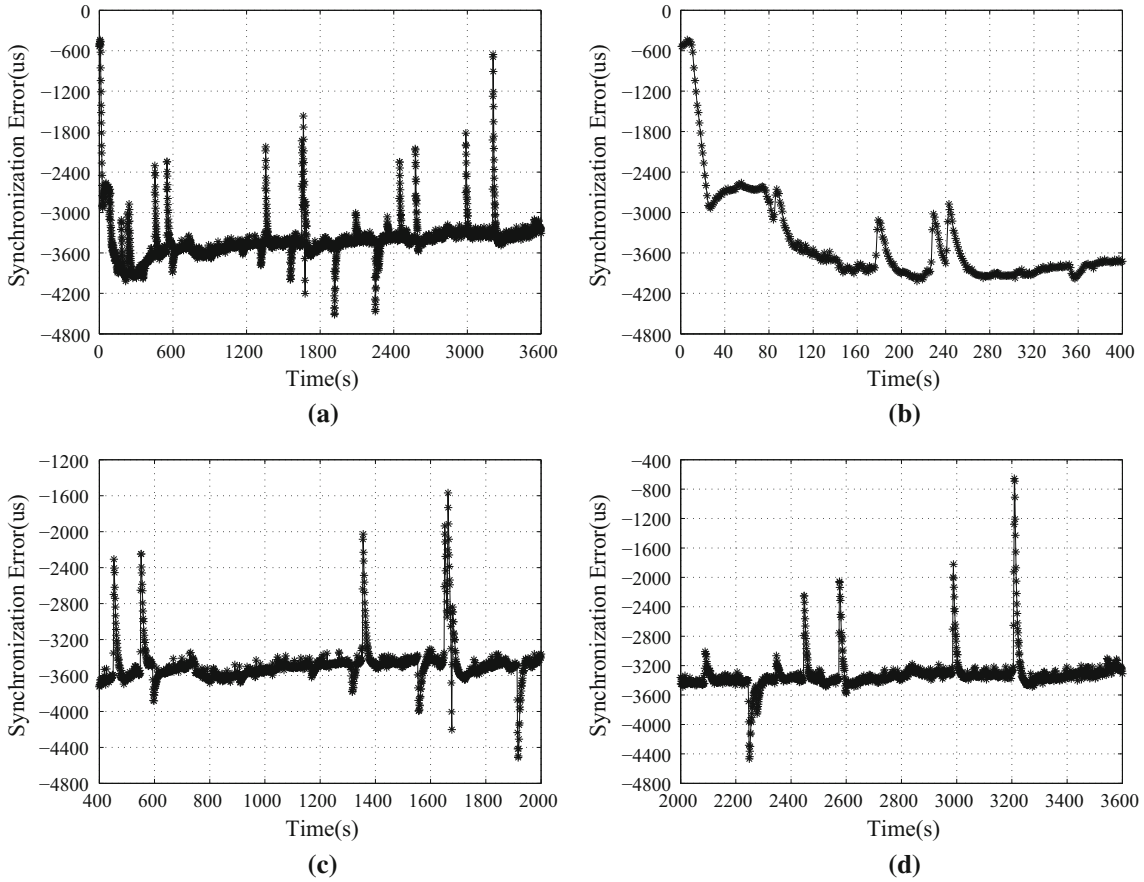


Fig. 13 Slave–slave clock offset under standard PTP in PC platform

nodes directly. We let one node generate pps signals and broadcast them over the RS-485 bus. The other two nodes receive pps signals and record the receiving time. By comparing the two timestamps, we get the clock synchronization accuracy between two receiving nodes. Measurement and clock synchronization are carried out simultaneously. The pps signal is generated based on Linux pps which is free software and has been merged into the Linux kernel.

We use the RS-485 method to measure synchronization accuracy mainly based on the following two reasons. First, the RS-485 bus supports broadcast and the broadcast signals will arrive at receivers with very little variability in their delays. Second, sending pps signals via serial buses rather than networks makes measurement and synchronization carried out simultaneously without competition for the TCP/IP protocol and channel resources. So it is suitable to measure the clock offset of the neighbor nodes.

7.2 Experiment result and analysis

In the experiment, the data measurement cycle is 1 s, and the test data is divided into 3,600 groups. We use the default period of PTP packets, and the period of Sync packet is set to 2 s, and the period of Delay_Req packets slightly varies between 4 and 60 s. The acquired data is plotted and the clock deviation of the master node and slave node is as follows.

Figure 7 shows the clock offset between slave nodes in a PC platform with the measurement range between 0 and 3,600 s. Figure 7(a) shows the offset of a full measurement range. Figure 7(b)–(d) respectively show the clock offset in the ranges of 0 and 400, 400–2,000, and 2,000–3,600 s. As can be seen from Fig. 7, the offset between the master node and slave node converges to 10 μs or less after 57 s under the action of the PI controller. The mean value of the offset is 3.701 μs, and the standard deviation is 25.332 μs.

Figure 8 shows the clock offset between the master node and slave node in a PC platform. And the offset convergence time is 143 s, mean value −2.046 μs, and standard deviation 25.883 μs. Figure 9 provides the histograms for slave–slave and master–slave clock offsets in a PC platform after it has converged.

Figures 10 and 11 respectively show the slave–slave and master–slave clock offsets in an embedded platform. In Fig. 10, the offset convergence time is 80 s, mean value 0.368 μs, and standard deviation 29.307 μs. In Fig. 11, the offset convergence time is 108 s, mean value −14.240 μs,

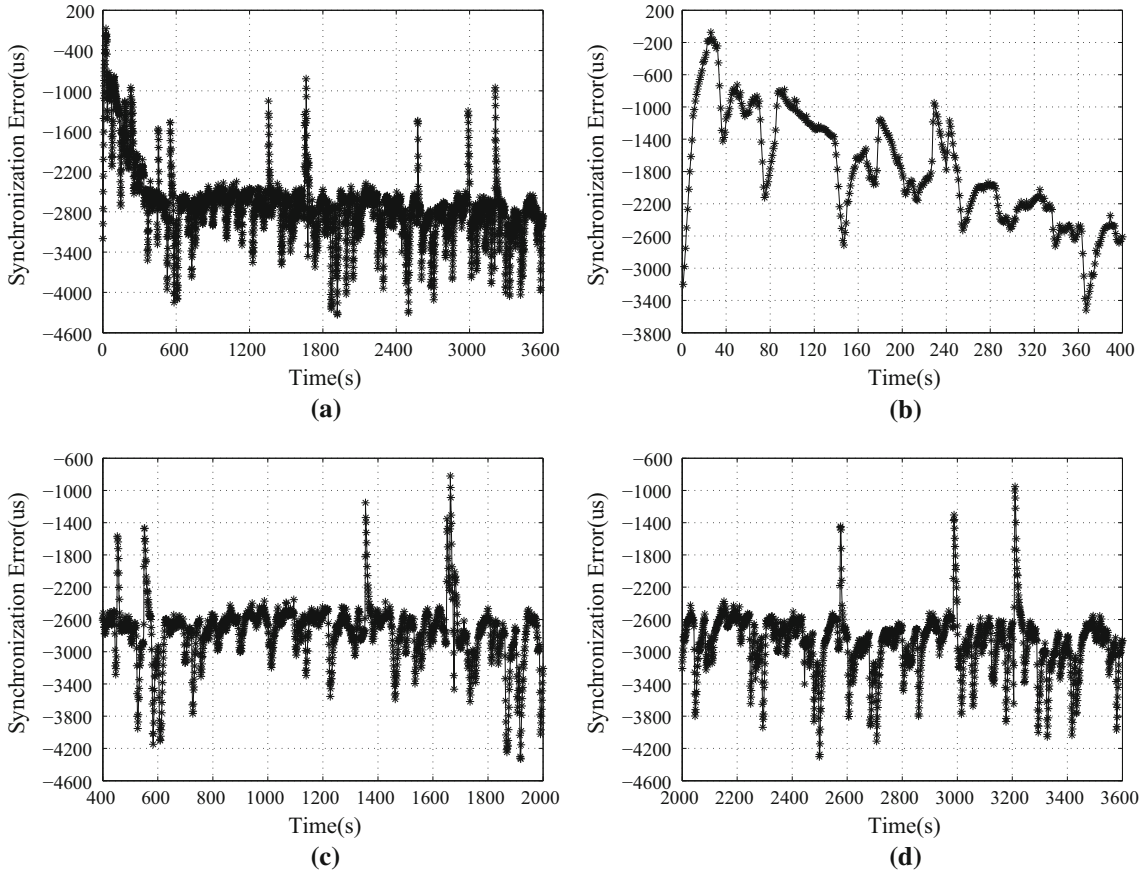


Fig. 14 Master–slave clock offset under standard PTP in PC platform

and standard deviation 27.652 μs. Figure 12 provides the histograms for the slave–slave and master–slave clock offset in an embedded platform after it has converged.

Comparing with the slave–slave convergence time, the master–slave convergence time is much longer. It is reasonable. The master node sends SYNC and FOLLOW_UP packets in multicast, the two slave nodes receive them at approximately the same time with little variability due to the reception time-stamping. On the contrary, the slave node sends Delay_Req packets in unicast. The transmission delay may vary for different slave nodes. So the master–salve convergence time is considerably longer than the slave–slave one.

From Figs. 7, 8, 9, 10, 11, 12, we can know that the performance of modified PTP is satisfied when it is applied in WLAN. The clock synchronization error is <100 μs. And the convergence time is <2 min. And the clock tracking error is <1 ms/h. So it can meet most application requirements of WLAN.

In order to compare the Kalman approach with the standard PTP, an experiment which uses standard PTP in a PC platform is conducted. The results are shown in Figs. 13, 14, 15. The slave–slave and master–slave clock offsets are illustrated in Figs. 13 and 14 respectively. The offset convergence time between the slave node and slave node is 471 s, mean value −3,403.340 μs, and standard deviation 246.015 μs. The offset convergence time between the slave node and master node is 578 s, mean value −2,839.765 μs, and standard deviation 372.383 μs. The histograms of offsets are also provided in Fig. 15. By the experimental results, we know that the Kalman approach can improve the performance obviously.

8 Conclusion

Clock synchronization is an important building block in WLAN. PTP has been effectively applied in wired network environment. But it is hard to achieve high precision in WLAN because of the existence of the asymmetric bidirectional delay. To solve this problem, a Kalman filter based delay estimation algorithm is proposed and a modified PI controller based clock servo system is designed. The validity of the system is proved by experiment.

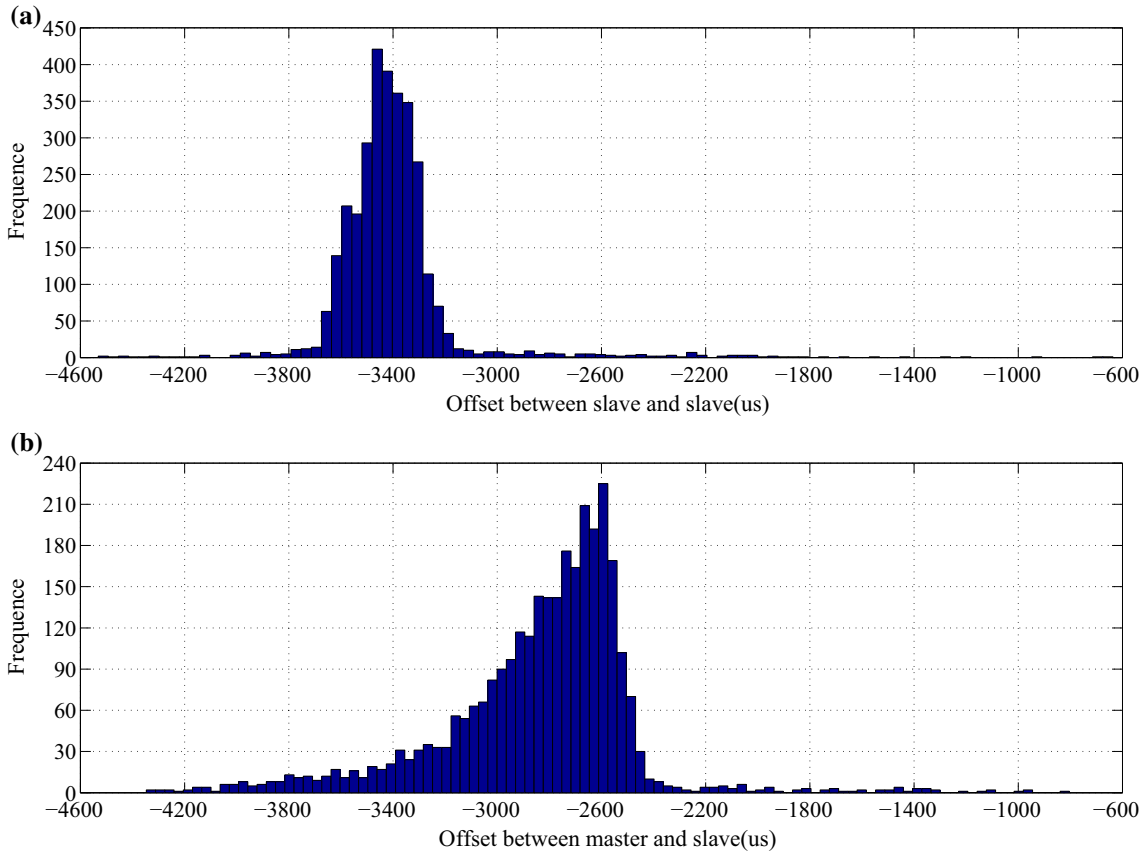


Fig. 15 Histogram of synchronization offset under standard PTP in PC platform

Experimental results show that PTP achieved high accuracy in synchronization error, short convergence time and small deviation error with the help of the proposed Kalman filter. The protocol and proposed Kalman filter are realized by pure software and clock accuracy can meet most application requirements of WLAN. In future research, we will implement the algorithm in multi-hop WLAN environment. In addition, PTP is still a master node based clock synchronization method, that is to say, it needs a central node. However, some wireless networks like WSN and Vehicular Ad-Hoc Network (VANET) are essentially distributed and decentralised network types, so we plan to use the synchronization method among several nodes to realize clock synchronization for distributed multi-hop wireless networks in future research, in order to improve the applicability and robustness of the clock synchronization protocol.

Acknowledgments This work has been carried out at the Wireless Adaptive Laboratory of Control and Networks Institute, and supported by the Fundamental Research Project of Northwestern Polytechnical University. The authors thank the anonymous referees for devoting their time to reviewing the paper and their invaluable comments and suggestions, which significantly helped to improve the quality of the manuscript.

References

1. (2002). IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. IEEE Std 1588–2002 (pp. i-144), doi:10.1109/IEEESTD.2002.94144.
2. (2007). IEEE standard for information technology—telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements—part 11: Wireless local area network medium access control (MAC) and physical layer (PHY) specifications.
3. (2008) IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. IEEE Std 1588–2008 (Revision of IEEE Std 1588–2002) pp. c1–269, doi:10.1109/IEEESTD.2008.4579760.
4. Akhlaq, M., & Sheltami, T.R. (2012). The recursive time synchronization protocol for wireless sensor networks. In *IEEE Sensors Applications Symposium (SAS), 2012*, IEEE, (pp. 1–6), doi:10.1109/sas.2012.6166318.
5. Akhlaq, M., & Sheltami, T. R. (2013). Rtsp: An accurate and energy-efficient protocol for clock synchronization in wsns. *IEEE Transactions on Instrumentation and Measurement*, *62*(3), 578–589. doi:10.1109/TIM.2012.2232472.
6. Bang, Y., Han, J., Lee, K., Yoon, J., Joung, J., Yang, S., & Rhee, J.K. (2009). Wireless network synchronization for multichannel multimedia services. In *Proceedings of the 11th International Conference on Advanced Communication Technology*, (pp. 1073–1077).
7. Chaudhari, Q. M., Serpedin, E., & Qaraqe, K. (2008). On maximum likelihood estimation of clock offset and skew in networks

with exponential delays. *IEEE Transactions on Signal Processing*, *56*(4), 1685–1697. doi:10.1109/TSP.2007.910536.
8. Chen, J., Li, Y., Song, Y., & Chen, H. (2007). Hardware-assisted clock synchronization in IEEE 802. 11 wireless real-time application. In *IET Conference on Wireless, Mobile and Sensor Networks*, 2007. (CCWMSN07). IET, pp. (363–366).
9. Chiang, J.H., & Chiueh, T.c. (2009). Accurate clock synchronization for IEEE 802.11-based multi-hop wireless networks. In *Proceedings of the 17th IEEE International Conference on Network Protocols, 2009. ICNP 2009. IEEE*, (pp. 11–20).
10. Cho, H., Jung, J., Cho, B., Jin, Y., Lee, S.W., & Baek, Y. (2009). Precision time synchronization using IEEE 1588 for wireless sensor networks. In *International Conference on Computational Science and Engineering, 2009. CSE'09.* IEEE, *2*, 579–586.
11. Cho, J. H., Kim, H., Wang, S., Lee, J., Lee, H., Hwang, S., et al. (2009). A novel method for providing precise time synchronization in a distributed control system using boundary clock. *IEEE Transactions on Instrumentation and Measurement*, *58*(8), 2824–2829. doi:10.1109/TIM.2009.2016365.
12. Chu, W. K., Zhang, F. M., & Fan, X. G. (2007). Measurement of real-time performance of embedded linux systems. *Systems Engineering and Electronics*, *29*(8), 1385–1401. doi:10.3321/j.issn:1001-506x.2007.08.041.
13. Cooklev, T., Eidson, J. C., & Pakdaman, A. (2007). An implementation of IEEE 1588 over IEEE 802.11b for synchronization of wireless local area network nodes. *IEEE Transactions on Instrumentation and Measurement*, *56*(5), 1632–1639. doi:10.1109/TIM.2007.903640.
14. Correll, K., & Barendt, N. (2006). Design considerations for software only implementations of the IEEE 1588 precision time protocol. In *Proceedings of the IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. Winterthur.
15. Djenouri, D., Merabtine, N., Mekahlia, F. Z., & Doudou, M. (2013). Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks. *Ad Hoc Networks*, *11*(8), 2329–2344.
16. Du, H. J., & Yang, N. (2007). Compression strategy for hash match algorithm in route list. *Systems Engineering and Electronics*, *29*(11), 1945–1948. doi:10.3321/j.issn:1001-506x.2007.11.038.
17. Du, J., & Wu, Y.C. (2013). Fully distributed clock skew and offset estimation in wireless sensor networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013*, IEEE, (pp. 4499–4503).
18. Eidson, J. C. (2006). *Measurement, control, and communication using IEEE 1588 (Advances in Industrial Control)*. New York: Springer.
19. Elson, J., Girod, L., & Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th Symposium on Operating systems design and implementation*, (pp. 147–163), doi:10.1145/844128.844143.
20. Exel, R. (2012). Clock synchronization in ieee 802.11 wireless lans using physical layer timestamps. In *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, 2012, IEEE, (pp. 1–6).
21. Ganeriwal, S., Kumar, R., & Srivastava, M.B. (2003). Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded networked sensor systems*, (pp. 138–149), doi:10.1145/958491.958508.
22. Huang, W., Quan, Y., & Chen, D. (2012). Improving broadcast efficiency in wireless sensor network time synchronization protocols. In *Proceedings of the International Workshop on System Level Interconnect Prediction*, ACM, (pp. 48–55), doi:10.1145/2347655.2347672.
23. Iwanicki, K., Steen, M., & Voulgaris, S. (2006). Gossip-based clock synchronization for large decentralized systems. In A. Keller & J. P. Martin-Flatin (Eds.), *Proceedings of the second IEEE international workshop on self-managed networks, systems, and services, lecture notes in computer science* (Vol. 3996, pp. 28–42). Berlin: Springer. doi:10.1007/11767886_3.
24. Jasperneite, J., Shehab, K., & Weber, K. (2004). Enhancements to the time synchronization standard IEEE-1588 for a system of cascaded bridges. In *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings. 2004*, (pp. 239–244), doi:10.1109/WFCS.2004.1377716.
25. Jiang, Y., Fan, Y., & Chen, X. (2013). Time synchronization protocol for wireless sensor networks with nodemonitoring. *Journal of Information and Computational Science*, *10*(4), 1213–1220. doi:10.12733/jics20101509.
26. Li, M. G., & Song, H. N. (2002). Research on computer clock synchronization technology. *Journal of System Simulation*, *14*(4), 477–480. doi:10.3969/j.issn.1004-731X.2002.04.020.
27. Mahmood, A., Gaderer, G., & Loschmidt, P. (2010). Software support for clock synchronization over IEEE 802.11 wireless lan with open source drivers. In *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2010*, IEEE, (pp. 61–66).
28. Mahmood, A., Gaderer, G., Trsek, H., Schwalowsky, S., & Kero, N. (2011). Towards high accuracy in IEEE 802.11 based clock synchronization using ptp. In *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2011*, IEEE, (pp. 13–18).
29. Mahmood, A., Exel, R., & Sauter, T. (2014). Impact of hard-and software timestamping on clock synchronization performance over ieee 802.11. In *Proceeding of the 10th IEEE Workshop on Factory Communication Systems (WFCS), 2014*, IEEE (pp. 1–8).
30. Maroti, M., Kusy, B., Simon, G., & Ledeczi, A. (2004). The flooding time synchronization protocol. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems, ACM Press*, (pp. 39–49), doi:10.1145/1031495.1031501.
31. Sakurai, T., & Vu, H. L. (2007). MAC access delay of IEEE 802.11 DCF. *IEEE Transactions on Wireless Communications*, *6*(5), 1702–1710. doi:10.1109/twc.2007.360372.
32. Shao, L., & Roy, S. (2005). Rate-one space-frequency block codes with maximum diversity for MIMO-OFDM. *IEEE Transactions on Wireless Communications*, *4*(4), 1674–1687. doi:10.1109/twc.2005.850374.
33. Sheu, J. P., Chao, C. M., & Sun, C. W. (2007). A clock synchronization algorithm for multi-hop wireless ad hoc networks. *Wireless Personal Communications*, *43*(2), 185–200. doi:10.1007/s11277-006-9217-4.
34. Shin, Y.J. & Lee, J.R. (2013). Time synchronization protocol in ad hoc network. In *International Conference on Information Networking (ICOIN), 2013*, IEEE, (pp. 375–378).
35. Snodgrass, T.E., & Stevens, J.A. (2008). Net formation-merging system and method. http://www.google.com.br/patents/US7430192, uS Patent 7,430,192.
36. Sun, Z.W. (2010). Research on the application of the high precision time synchronization of IEEE1588. Master's thesis, National Time Service center, Chinese Academy of Sciences.
37. Sundararaman, U., Buy, U., & Kshemkalyani, A. D. (2005). Clock synchronization for wireless sensor networks: A survey. *Ad Hoc Networks*, *3*(3), 281–323. doi:10.1016/j.adhoc.2005.01.002.
38. Wolf, B.J., Russell, H.B., & Wang, K.C. (2007). Synchronizing transmission schedules of partitioned ad hoc networks. In *Proceedings of the IEEE Military Communications Conference*, (pp. 1–7), doi:10.1109/milcom.2007.4455105.
39. Yang, H. D., & Deng, Y. (2008). Research on time synchronization in ad hoc networks. *Computer Engineering and Design*, *29*(21), 5447–5450.
40. Ye, Q., Zhang, Y. & Cheng, L. (2005). A study on the optimal time synchronization accuracy in wirelesssensor networks.

Computer Networks *48*(4):549–566, doi:10.1016/j.comnet.2004.10.018, http://www.sciencedirect.com/science/article/pii/S1389128605000006X

41. Yildirim, K. S., & Kantarci, A. (2013). Drift estimation using pairwise slope with minimum variance in wireless sensor networks. *Ad Hoc Networks*, *11*(3), 765–777. doi:10.1016/j.adhoc.2012.09.003.
42. Zou, C., & Lu, Y. (2012). A time synchronization method for wireless sensor networks. In B. Liu, M. Ma, & J. Chang (Eds.), *Information computing and applications, lecture notes in computer science* (Vol. 7473, pp. 221–228). Berlin: Springer. doi:10.1007/978-3-642-34062-8_29..

Wu Chen received the B.S. and M.S. degrees in Navigation Guidance and Control from Northwestern Polytechnical University, China, in 1992 and 1997. He also received the Ph.D. degree in Control Theory and Control Engineering from the Northwestern Polytechnical University, in 2000. Currently, he is a associate professor of School of Automation, Northwestern Polytechnical University, China. His main research interests include ad hoc network, intelligent control, embedded system technology and information security. He has published more than 20 research papers in refereed international conferences and premier journals. His research now is supported by the Fundamental Research Project of Northwestern Polytechnical University.

Jianhua Sun is a current master student of School of Network and Information Security, Northwestern Polytechnical University. His research interests focus on ad hoc networks.

Lu Zhang is a current master student of School of Network and Information Security, Northwestern Polytechnical University. He is working on topics related to ad hoc networks.

Xiang Liu is a current master student of School of Network and Information Security, Northwestern Polytechnical University. His research interests are in area of ad hoc networks.

Liang Hong is a associate professor of School of Secrecy, Northwestern Polytechnical University, China. His main research interests include wireless network technology and information security.