Time Synchronization for Multi-hop Surveillance Camera Systems

Hyuntae Cho

Abstract In recent years, surveillance systems designed for public safety have become more intelligent by providing context awareness. Traditional surveillance camera systems require access to energy and networking infrastructure in order to operate and to transmit the recorded video data. Since such requirements can increase the costs incurred when installing and maintaining surveillance systems, a wireless surveillance camera system is hereby introduced. The system can operate with low power consumption and also provides network connectivity. The battery life of the system is improved by separating the system into master and slave subsystems. The master subsystem provides Wi-Fi connectivity and records video while the slave-subsystem provides low-power event detection with ZigBee connectivity. The system uses Wi-Fi mesh networks to transmit video data and ZigBee networks to define the network topology and to synchronize multiple surveillance camera systems. Time synchronization is a fundamental issue for distributed surveillance camera systems, so this chapter details a method to synchronize time among multiple surveillance camera systems by using ZigBee radio communications.

Keywords Time synchronization · Clock synchronization · Wireless mesh networks · Wireless surveillance systems · Zigbee

1 Introduction

The increase in crime in residential areas and in public spaces has resulted in an increase in demand for surveillance systems, such as those provided by CCTV or by security services [1, 2]. Recently, the market for surveillance camera systems has shifted from CCTVs to IP-based cameras because IP-based cameras offer advantages over CCTVs in terms of resolution, cost, potential applications, etc. [3]. Furthermore,

H. Cho (✉)
Center for Integrated Smart Sensors, ITC Building(N1), KAIST, Daehak-ro 291, Yuseong-Gu, Daejeon 305-701, Republic of Korea
e-mail: phd.marine@kaist.ac.kr

C. Kyung (ed.), *Theory and Applications of Smart Cameras*, KAIST Research Series, DOI 10.1007/978-94-017-9987-4_13

big data, cloud, and IoT services have extended the potential applications of IP-based cameras [4]. However, these camera systems have many technical requirements, so it can be difficult to use them under specific circumstances. In particular, surveillance systems that are based on traditional cameras require access to power and network infrastructure, which results in high installation and maintenance costs for surveillance systems.

This chapter describes a wireless surveillance camera system that provides IP connectivity through use of a wireless networking platform. The proposed system captures images and video and then transmits the recorded data to a remote point through a self-organized wireless network. Since wireless networks consume large quantities of energy to provide network connectivity and to transmit video data, the proposed system is equipped with a dual radio system to conserve energy [5]. The system comprises two subsystems: a master subsystem and a slave subsystem. The master subsystem records and processes video and then transmits the recorded video by using a Wi-Fi mesh network. The slave subsystem turns the master subsystem off to maintain the entire system in a low-power mode as long as possible when no events require video to be captured and transmitted. The slave subsystem also determines the network topology, including synchronizing time, by using a control channel based on energy-efficient ZigBee communications. Time synchronization affects the performance of the entire system and the network for such distributed surveillance camera systems, and so it is a fundamental issue. Time synchronization can be generally achieved by exchanging time information. The exchange of time information via wireless networks can result in uncertainty due to signal delay and jitter, particularly when using a ZigBee network. Therefore, this chapter analyzes uncertainties in the ZigBee network protocol stack and introduces basic techniques to eliminate or minimize them.

This chapter is organized as follows. In Sect. 2, we present conventional approaches used for surveillance camera systems with wireless networking. In Sect. 3, we describe the proposed system and the corresponding network platform, and then we present with basic techniques to synchronize time across distributed wireless surveillance camera systems.

2 Related Work

The recent introduction of low-cost CMOS image sensors has resulted in an increase in the range of applications of wireless video networks [6]. Devices can use these sensors to capture pictures or video from the environment, and one such use case involves wireless sensor networks that provide surveillance and security by using a network of nodes to identify and track objects according to visual information. Wireless video sensor networks can also greatly improve applications in the area of environmental monitoring [7, 8]. Visual information from the environment is important for such applications, including for precision farming or habitat monitoring. Wireless video sensor networks will also enable new forms of

entertainment where real-time visual information can be provided at a large scale from a remote location, such as for a digital zoo [9].

Wireless cameras can be used to monitor and track objects in the field, such as in construction sites, harbors, forests, and campuses, and Firetide Inc. [10] and Strix Systems [11] are well-known commercial vendors of traditional wireless mesh products. Current products focus on conventional problems and focus on features that improve performance in terms of coverage, quick mobility, reliability, security, and solid networking. In such products, the camera is simply mounted on a wireless mesh platform.

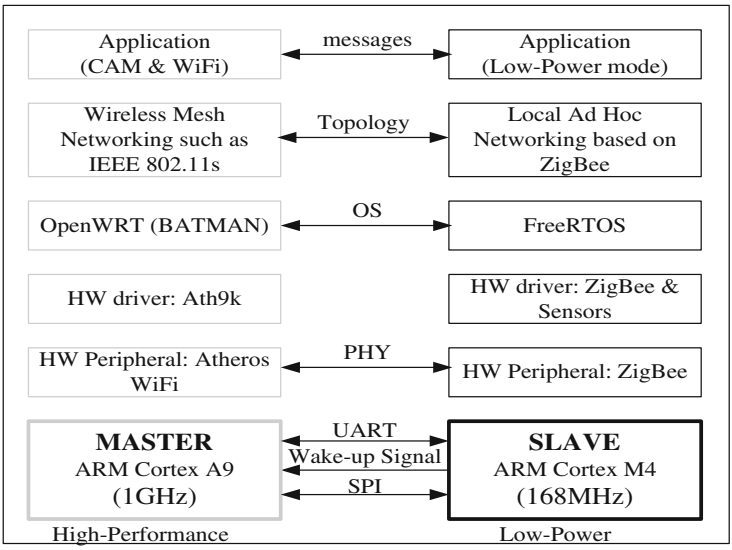
Several studies [12, 13] have investigated issues relevant to these systems, including video transmission, multi-channel operation, and improvements in network bandwidth. Raniwala and Chiueh [12] presented a multi-channel WMN architecture that effectively improves the bandwidth by exploiting nonoverlapped radio channels available through IEEE 802.11 standards. S. Yang constructed multi-radio, multi-hop wireless mesh networks by developing a Linux-based implementation of a WLAN mesh system [13]. The main design goal for our system is to fully exploit link layer characteristics in order to improve the configuration flexibility as well as the network performance. Although some research has been performed to date for wireless mesh networks for surveillance purposes, such studies have only focused on traditional mesh networking problems.

3 Wireless Surveillance Camera System for Public Safety

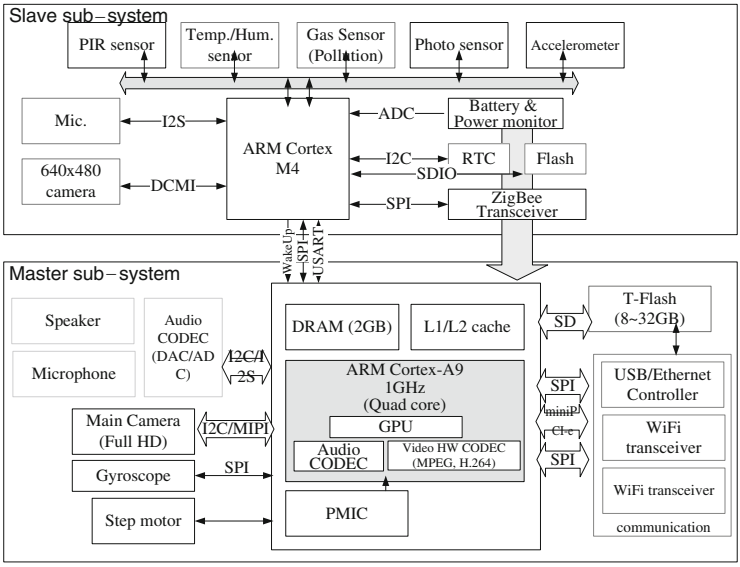
Low-power surveillance systems and network platforms require different approaches from those used in conventional systems. This section describes new approaches to reduce power consumption to provide wireless communications for surveillance camera systems. The proposed system consists of two subsystems: a high-performance master subsystem and a low-power slave subsystem, as shown in Fig. 1. The master subsystem is based on an ARM Cortex A9 processor [14] and uses OpenWRT [15] to manage the system. The master subsystem records video and transmits video data by using an FHD camera, a microphone, a high-speed application processor, and Wi-Fi network interfaces. The Ubiquiti Networks SR71 WLAN card [16] and Ath9k are used to provide Wi-Fi communications. For load balancing, the communications radio is separated into four modules: two up-links and two down-links. The master subsystem can also be connected to the Internet via Ethernet, which is more reliable than wireless networks. In addition, the GPU on the main processor helps the system recognize objects and mitigates the load on the main processor core.

The slave subsystem is responsible for topology management and low-power maintenance of the entire system. It includes a VGA camera, a low-power MCU, a microphone, an RTC, memory, and a power management circuit with a solar cell. It also includes gas, temperature, humidity, ozone, UV, and smoke sensors to detect external events and a ZigBee transceiver to provide channel control. The slave

subsystem uses an ARM Cortex M4 processor (STM32F407 [17]) as its main processor and FreeRTOS [18] to manage tasks. The detailed architecture is presented in Fig. 1 with (a) the network protocol stack of the entire video sensor system and (b) the hardware block diagram of the system.



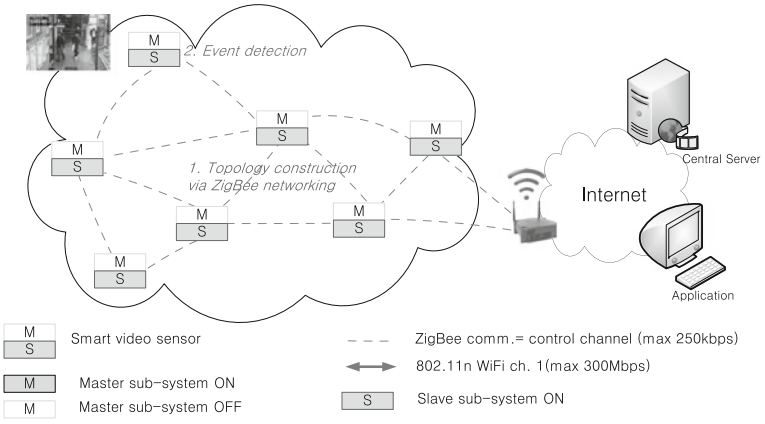
Overview of the system stack



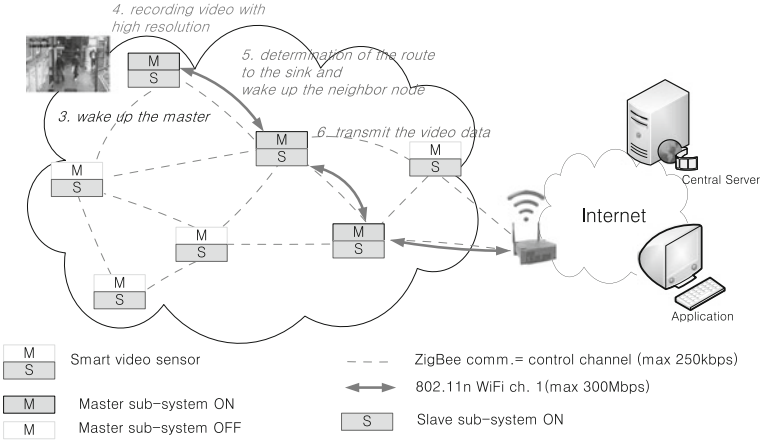
block diagram

Fig. 1 Conceptual overview of the proposed surveillance system

The proposed system transmits recorded video to a remote user through multi-hop communications via the Wi-Fi mesh network. To reduce energy consumption during transmission, we additionally separate the communications channel over dual radios because traditional Wi-Fi mesh networks consume a high amount of energy. Figure 2 shows the conceptual overview of the wireless video sensor network platform. The proposed system uses ZigBee for channel control and a Wi-Fi mesh channel to transmit video data. The system initially operates in its low-power mode by turning the master subsystem off since the master subsystem consumes more energy by several orders of magnitude higher than the slave subsystem does. The network topology and the route to the sink are constructed through the control channel. A number of routing and topology management protocols have been previously developed, including OSLR, AODV, DSR, or



Topology construction and management via the ZigBee control channel



Routing and video data transmission

Fig. 2 Wireless video transmission for low-power communication

BATMAN-advanced [5, 19]. The OSLR and BATMAN protocols are frequently used for Wi-Fi mesh networks.

When the slave subsystem detects an event (whether from the camera, the microphone, the other sensors, or the ZigBee radio), it wakes up the master subsystem to record the ambience and to transmit the recorded data. The master subsystem processes more visual data and extracts much more information than the slave subsystem because the master subsystem has a higher performance processor and a GPU, such as ARM's Mali, with higher data processing capability. The captured video is compressed by system in a manner according to the detected level of importance or the bandwidth available. In addition, the system and the network platform mitigate traffic over the wireless mesh networks by adopting multi-channel, multi-radio, and multi-path approaches.

The system wakes up the neighboring nodes before transmitting the recorded data, and it is important to quickly synchronize time across the nodes to in order to properly organize their schedule in the network. The proposed system would consume a large quantity of energy if Wi-Fi were used to maintain the topology and time synchronization. Therefore, a ZigBee radio is used to perform topology maintenance and to synchronize time, thereby reducing energy consumption. The next section describes the uncertainties introduces by the ZigBee network protocol stack and the methods through which these can be reduced to provide precise time synchronization.

4 Time Synchronization Over ZigBee Networks for Surveillance Camera Systems

Time synchronization is critical for wireless surveillance camera systems as it is for modern computer networks where transmissions must be managed and scheduled while handling contention, among other things. Time synchronization is achieved by sending and receiving time information and frequency over the packet network. A synchronization protocol is used to exchange the time information, such as the *offset* and *propagation delay*, and to synchronize all clocks. This section describes the basic principles for ZigBee-based time synchronization, which is used for the wireless surveillance camera systems.

4.1 Time Synchronization Methods

The global positioning system (GPS) enables precision time-keeping through its satellite clocks because timing is based on a standard atomic clock that uses the oscillations of a particular atom, such as Cesium or Rubidium, as a metronome. Such clocks provide the most stable and accurate time reference. This timing information is obtained by GPS receivers, which require precision timing to compute their distance to each satellite in order to derive their position on Earth.

The Network Time Protocol (NTP) [20] is widely used to synchronize time over computer networks. NTP timestamps are numbered and are exchanged between peers, and messages are then exchanged to calculate the time offset and to synchronize clocks by correcting for the offset. The propagation delay is calculated by using the round trip time.

The IEEE 1588 precision time protocol (PTP) [21] provides a standard method to synchronize devices in a network with submicrosecond precision. The protocol synchronizes slave clocks to a master clock, ensuring that events and timestamps for all nodes use the same timer values. Since a time difference between a master clock and a slave clock is a combination of the clock offset and the message transmission delay, the clock skew is corrected in two phases: offset correction and delay correction. The master node initiates the offset correction by using a sync message and a follow-up message. When the master node sends a sync message, the slave uses its local clock to timestamp the arrival of the sync message. The slave then compares the local timestamp to the actual sync transmission timestamp from the master clock's follow-up message. The difference between the two timestamps represents the offset for the slave, plus the message transmission delay. The second set of messages is necessary to account for variations in the network delay. The slave then timestamps the instant when a delay request message is sent, and the master clock timestamps the arrival of the delay request message. It then sends a delay response message with the delay request arrival of the timestamp. The difference between the timestamps is the slave-to-master delay. The slave averages the two directional delays and then adjusts the clock by the time of the delay to synchronize the two clocks. Since the master and slave clocks drift independently, the offset correction and delay correction are periodically repeated to maintain the clock synchronized (Fig. 3).

In WSNs, sensor nodes synchronize their time according to a reference clock, such as that of the sink node or coordinated universal time (UTC), which is the time standard by which the world regulates clocks and time in time synchronization. For

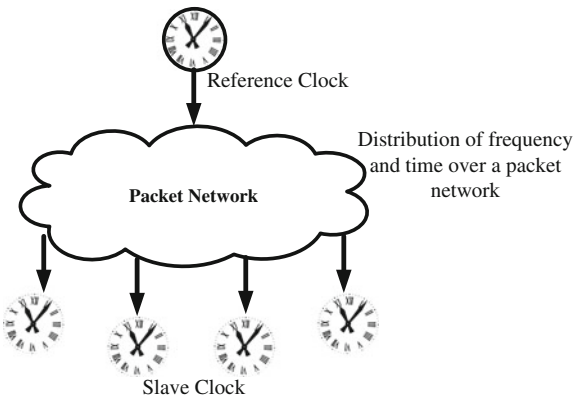


Fig. 3 Time synchronization over a packet network

WSNs, time synchronization requires clocks to be synchronize across a set of sensor nodes connected to one another over single-hop or multi-hop wireless networks. To date, various protocols have been designed to address this problem [22–31]. Time synchronization may be classified into three types: (a) simple unidirectional broadcast, (b) receiver-receiver synchronization, and (c) bidirectional pair-wise synchronization, as shown in Fig. 4.

In the unidirectional reference broadcast method, a reference node simply broadcasts a reference clock signal to other nodes, and these other nodes correct their times to match the reference clock. This method is the oldest and simplest method to synchronize time across a network. The flooding time synchronization protocol (FTSP) [32] is the most well-known approach. FTSP uses a fine-grained clock, media access control (MAC) layer time stamping to reduce jitter and clock drift estimation in order to achieve a relatively high level of precision.

Receiver–receiver synchronization uses an external beacon node that periodically sends beacon messages to the sensor nodes. The sensor nodes that receive the beacon messages exchange the arrival times of the messages among themselves to compare and correct their clocks. Reference broadcast synchronization (RBS) [33] and adaptive clock synchronization (ACS) [34] are receiver–receiver synchronization

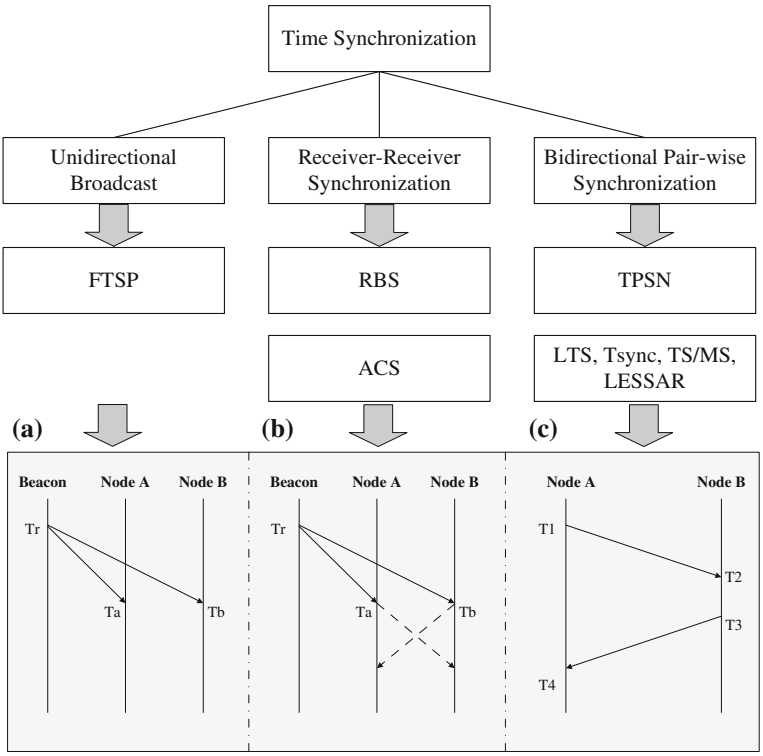


Fig. 4 Classification of time synchronization protocols for WSNs

protocols. RBS does not utilize an explicit timestamp, but rather receivers use the arrival times as points of reference to compare their clocks, as shown in Fig. 4b. This approach directly removes two of the largest sources of non-determinism involved in message transmission: the transmission time and the access time in the network protocol stack. ACS extends RBS and focuses on reducing the number of the messages that are used to exchange the message arrival times. In order to reduce the number of messages, the beacon node is used instead of the sensor node to gather and compare the message arrival times.

Third, bidirectional pairwise synchronization, which can also be referred to as sender–receiver synchronization, uses the round trip time of the message to correct the offset and the propagation delay. This approach uses a handshake protocol between a pair of nodes. That is, sensor nodes achieve clock synchronization with their parent node unlike receiver–receiver synchronization where sensor nodes synchronize their clocks with other sensor nodes on the same level. Figure 4c depicts an example of the basic operation of this method in three sequential phases. First, node A sends its local time at time $T1$, and node B receives the message at time $T2$ and records its local time. Then, time $T2$ is calculated as $T2 = T1 + d + \delta$, where d is the propagation delay between two nodes and δ denotes a clock offset between them. Next, node B responds to node A with an ACK message containing times $T2$ and $T3$. After receiving the ACK message at time $T4$, node A determines time $T4$ as $T4 = T3 + d–\delta$. Finally, node A can calculate the clock offset and the propagation delay between two nodes, as below:

$$\begin{aligned} d &= \frac{[(T2 - T1) + (T4 - T3)]}{2} \\ \delta &= \frac{[(T2 - T1) - (T3)]}{2} \end{aligned} \tag{1}$$

The timing-sync protocol for sensor networks (TPSN) [35], lightweight time synchronization (Tsync) [36], tiny-sync and mini-sync (TS/MS) [37], and level synchronization by sender, adjuster, and receiver (LESSAR) [38] are well-known bidirectional pairwise synchronization protocols for WSNs while NTP is a form of bidirectional pairwise synchronization protocol used over the Internet. TPSN provides synchronization for an entire network. First, a node is elected as a synchronization master, and a spanning tree with the master at the root is constructed by flooding the network. In the second phase, the nodes synchronize to their parent in the tree by means of round-trip synchronization. TSync has a centralized version and a decentralized version. Both protocols use a dedicated radio channel to synchronize messages in order to avoid inaccuracies due to packet collisions. TS/MS uses multiple pairwise round-trip measurements and a line-fitting technique to obtain the offset and drift of two nodes, rather than directly calculating the offset. LESSAR is able to achieve accuracy within a given limitation while also retaining low power consumption, affordable storage, and small computation complexity due to the reduction in packet transmissions.

4.2 Uncertainty in Time Synchronization

Uncertainty is inserted communication within the network protocol stack from the application to the physical layer, including the communications link, as shown in Fig. 5. The time uncertainty in the network protocol stack is dependent on the determination of an instant of time, and such a determination during time synchronization is referred to as time stamping. The time stamping point is critical because it affects the accuracy of the time synchronization procedure. The time stamping point can be determined for any point within the network layers. However, time stamping at an upper layer, such as the application layer, has a disadvantage in that the protocol stack can cause delays that may not be deterministic. The delay between the time stamp and the transmission can vary between a minimal value and a maximal value, depending on the network and protocol states. Transmission can be delayed if it causes a collision, and time stamping by the receiver can be performed at the start of an interrupt, after receiving a frame. The delay in the reception can vary according to the protocol stack and the kernel activity. The delay and jitter can be reduced by performing time stamps as close to the wires as possible [39, 40].

The lowest time stamp point with software is at the MAC layer. However, time stamping at the MAC layer also suffers from delay and jitters. We deal with IEEE 802.15.4 and ZigBee, which is based on carrier sense multiple access (CSMA). Bidirectional pairwise synchronization has an advantage in that uncertainties at the network protocol stack and the propagation delay can be mitigated by using exchange messages. However, this approach requires additional traffic, and the number of messages increases as the scale of the network increases. That is, surveillance cameras contend among themselves to access the channel, as shown in Fig. 6. Thus, a busy channel leads to nondeterministic latency in the MAC layer and finally diminishes the accuracy and precision of the time synchronization. In other words, the MAC verifies whether the channel is clear before it sends a sync or ACK message. If the channel is busy as a result of transmitting other messages, MAC

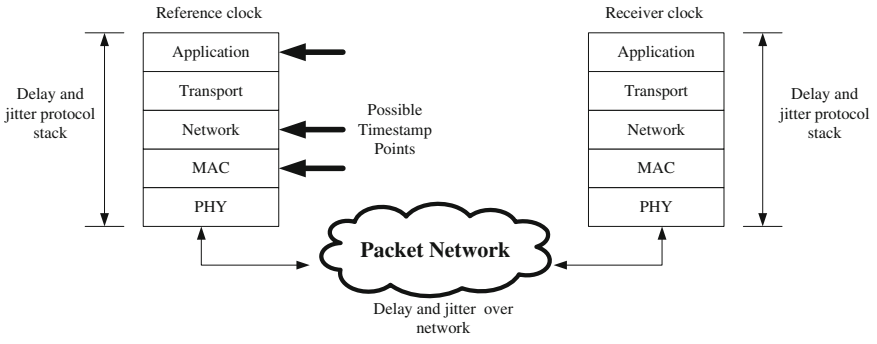


Fig. 5 Uncertainty at the network protocol stack

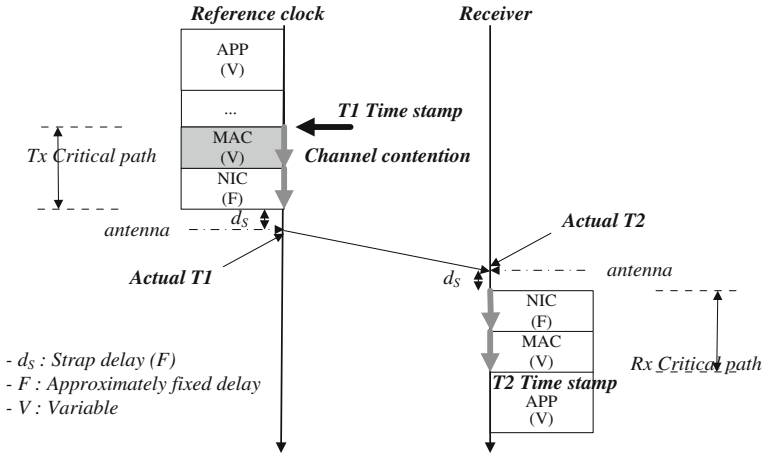


Fig. 6 Uncertainty at the MAC layer

waits for a random back-off period. After waiting for this random back-off period, the node resends the message, including the time stamp value. This delay introduces a serious uncertainty. Thus, the number of messages should be reduced and collisions between the messages must be avoided in order to increase the accuracy and precision of time synchronization.

Tsync [36] and LESSAR [38] are lightweight time synchronization protocols (Fig. 7). These protocols use three message types: *sync*, *delay_req*, and *delay_resp*. A *sync* message is initially sent by the reference clock node, which is defined as level 0 and acts as the root node. The reference node inserts time $T1$ into the *sync* message, and each sensor node receives the packet at time $T2$ and records their local clock. Then, the sensor node determines the clock offset as $\delta = T2–T1$. When calculating the delay between the reference node and other nodes, delay calculations from all of the child nodes can produce a high amount of traffic, which results in inaccurate synchronization.

Thus, the uncertainties in the propagation speeds are assumed to be the same in different nodes, and the uncertainty of the propagation delay is less than that of other uncertainties, such as the send, access, receive time, etc. This assumption underlies the proposed method, where only one child node responds in order to calculate the propagation delay from the reference node or the parent node. The reference node determines which node responds to the sink node in order to measure the delay by consulting its neighbor list. This selection is based on a min-ID selection. The information used for the responding node is inserted into the *sync* message, and the node receiving the *sync* message first checks whether it itself is the target for the message. If so, the node sends a *delay_req* message that includes times $T2$ and $T3$. Otherwise, it will be discarded. Then, the reference node receives the *delay_req* message at time $T4$ and records the arrival time for the message. Next, the reference node determines the propagation delay between its

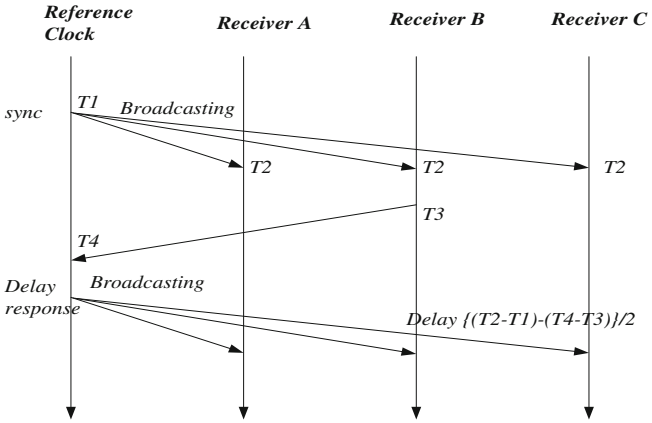


Fig. 7 Lightweight time synchronization

one-hop children nodes and itself, as shown in Eq. (1), and the dealy is broadcast to its one-hop nodes. Finally, the child nodes can correct the propagation delay by receiving the *delay_resp* message from the reference node. As a result, these methods can mitigate random delays at the MAC layer.

However, minimizing the messages is not the optimum solution to remove the random back-off delay. In order to eliminate the delay and the jitter at the MAC layer, it is important to implement hardware-assisted time stamping. Time stamps that use a hardware-assisted stamper can be performed at the media-independent interface between the MAC layer and the Zigbee physical (PHY) layer. When a Zigbee device receives MAC protocol data from the upper layer, it generates a four byte preamble with a one-byte start of frame delimiter (SFD) and a one-byte frame length. Then, the device transfers data to the MAC protocol data unit (MPDU) and performs a cyclic redundancy check (CRC), as shown in Fig. 8. After the last bit of the SFD is transferred at this point, the ZigBee transceiver causes the SFD pin to increase. The time-stamping unit of the sensor node detects the rising edge of the SFD pin, and then the hardware-assisted time stamping unit can detect and store the value of the local clock counter in an internal register.

Figure 9 depicts the time stamping points of the SFD from the time processing unit. Figure 9a illustrates the hardware-assisted time stamping unit and (b) shows that the hardware-assisted time stamping unit eliminates uncertainty at the MAC layer and has the same delay. The time stamping unit is independent of the processor of the main module of the system, and this time stamping unit can be implemented by using an independent processor or FPGA. However, it should be connected to the main processor, which executes the time synchronization protocol.

After hardware-assisted time stamping is performed, the stamped time should be inserted into the message to be transmitted to the receivers. The time captured from the SFD signal is inserted into the MAC protocol data, as shown in Fig. 10a. For the wireless bit-stream, most of the wireless controllers (ZigBee transceiver) provide

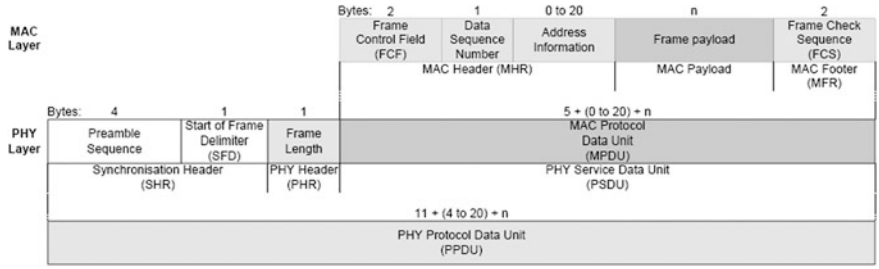


Fig. 8 Frame of IEEE 802.15.4 [41]

FIFOs to transmit and receive data. The processor communicates with the controller (e.g., TI CC2420 and CC2520) by using a synchronous peripheral interface (SPI). Generally, when a message is transmitted, the processor loads up the transmit FIFO with the entire message and then enables transmission.

The timestamp is inserted into the message, and the rest of the message is placed in the FIFO. Assuming that this can all be done quickly enough, the entire message is transmitted properly. If, however, the process is too slow, the FIFO will underrun and the message transmission will abort. This is a real concern since ZigBee specifies a fairly speedy effective bit rate of 250 kbps, where 4 μs is required to transmit 1 bit. In order to insert a time stamp into the payload, the communication speed between the processor and the FIFO of the ZigBee transceiver should be faster than 250 kbps. Generally, the speed of ZigBee transmissions is lower than the actual data rate due to coexistence with Wi-Fi and other radios. Furthermore, since ZigBee-based systems target low-power operation, such systems use low-power processors with a low clock speed. Therefore, these low-speed processors do not achieved the speed required to insert the time stamp into the message.

The message that is used to synchronize time can be separated into two messages, as shown in Fig. 10b. The reference clock sends a sync message, and after passing the *sync* message at $T1$, the time stamping unit reads and stores the local time of the reference clock. The time $T1$ is not inserted into the *sync* message. After receiving the *sync* message, the receiver clock records the value of the local clock counter at $T2$. If the receiver node has information for $T1$ and $T2$, it can calculate the offset between the reference and the receiver clock. However, it does not have information for $T1$, and the reference clock inserts the time stamp for $T1$ into the consequent message. The reference clock may also send the consequent message, which it always associates with a specific *sync* message and contains a more precise estimate for the reference time. The receiver clock uses the information contained in the consequent message to correct its local clock, so as to synchronize time with the reference clock. Such an approach can reduce the uncertainty at the MAC layer during time synchronization.

The propagation delay is also measured, calculated, and corrected according to the round-trip time based obtained using a hardware-assisted time stamping unit, as shown in Eq. (1).

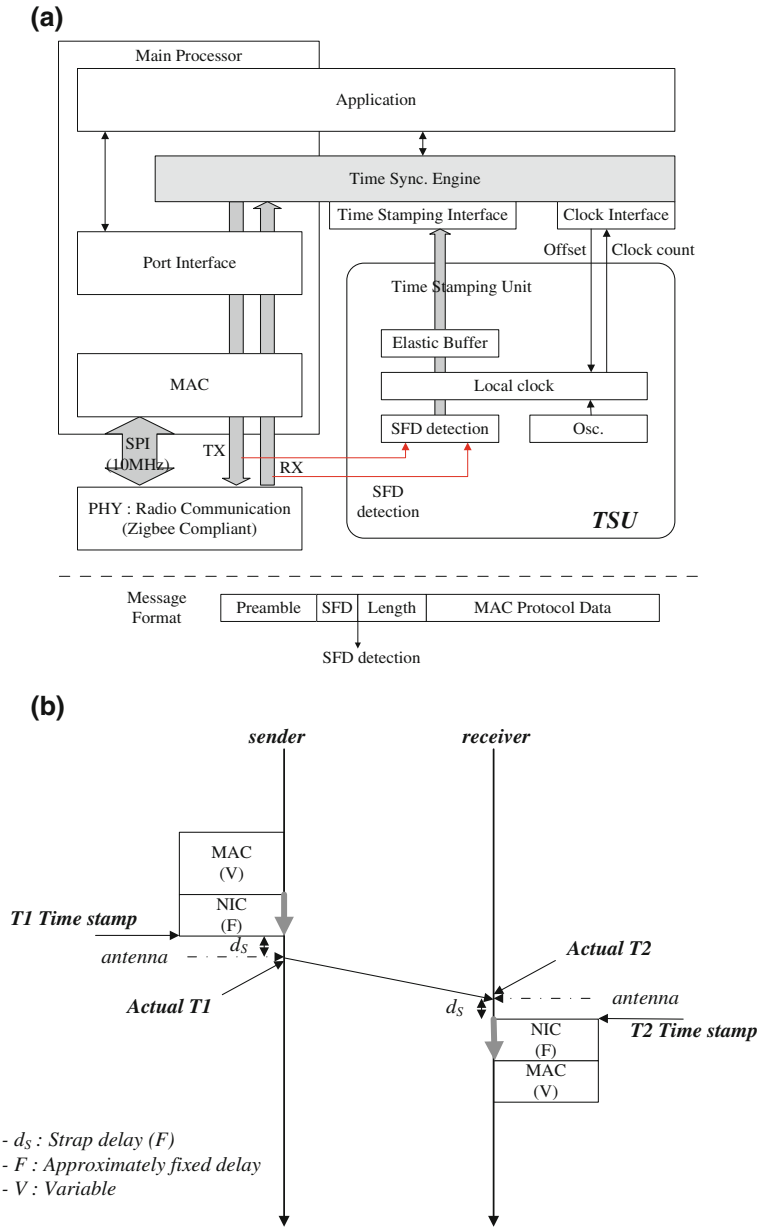


Fig. 9 Hardware-assisted time stamping unit and time stamping points **a** Hardware-assisted time stamping unit, **b** Hardware time stamping point at the sender and the receiver

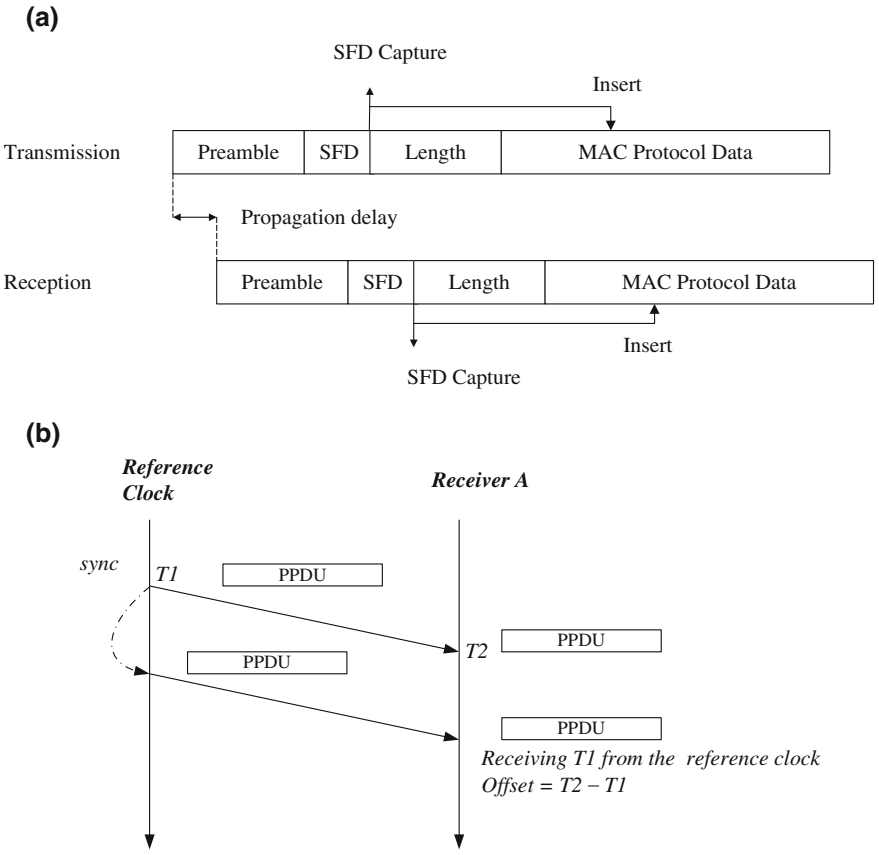


Fig. 10 Precision time stamping and transmission **a** direct insertion of the time stamp and transmission, **b** Consequent transmission of the precision time information

4.3 Drift and Correction

A system clock is controlled by a crystal oscillator that operates in a pre-determined manner. Under ideal circumstances, physical clocks oscillate at a constant frequency, but in the real world, manufacturing variations and exposure to out-of-tolerance conditions (e.g., mechanical shock) result in permanent frequency errors in the crystals. In addition, variations in the temperature, age, humidity, etc., result in short-term errors in the crystals. An oscillator with a 1 parts-per-million (PPM) frequency tolerance has a one microsecond drift every second. In general, cheap oscillators have a frequency tolerance from 20 to 50 PPM, where the maximum drift rate is between 20 and 50 microseconds per second. Such a value is inadequate to provide precise time protocol. Thus, for most cases, a temperature-compensated crystal oscillator (TCXO) with a 1.5 PPM frequency

tolerance reduces the drift rate. Even if two clocks are initially synchronized by correcting the time offset and delay, a difference can accumulate between them as time progresses [42].

Assume that the local clocks at two nodes, i and k, are $c_i(t)$ and $c_k(t)$. If $c_i(t) = c_k(t)$, the two clocks are synchronized at time t. If the algorithm for time synchronization could know the relative offset between $c_i(t)$ and $c_k(t)$ at time t, $c_k(t)$ can be synchronized to $c_i(t)$ at each epoch by correcting for the relative offset. Figure 11 represents the synchronized clock $c_k^o(t)$. Although $c_k(t)$ is exactly synchronized to $c_i(t)$ through a periodic correction, clock $c_k^o(t)$ pursues a line derived from a variation in clock $c_k(t)$ because this synchronization did not consider clock drift. Thus, LESSAR assumes that clock drift quickly changes, and therefore, the synchronization procedure is frequently conducted. However, this eventually reduces the synchronization accuracy because the channel remains busy with excessive synchronization messages [42, 43].

Frequent sync messages can help calculate the drift from the reference clock, as shown in Fig. 12. Equation (2) presents the drift compensation correction for the receiver nodes.

$$\Delta_m = T_{m+1} - T_m$$

$$\Delta_s = T_{s+1} - T_s$$

$$\Delta_{\text{diff}} = \frac{\Delta_s - \Delta_m}{\Delta_m} \tag{2}$$

where Δ_m is the clock drift of the reference clock node that applies to clocks between T_m (the first time stamp) and T_{m+1} (the consequent time stamp). Δ_s is the clock drift of the sensor node, and it applies to clocks between the arrival time of the *sync* message, T_s, and the arrival time of the consequent sync message, $T2$. Δ_{diff}

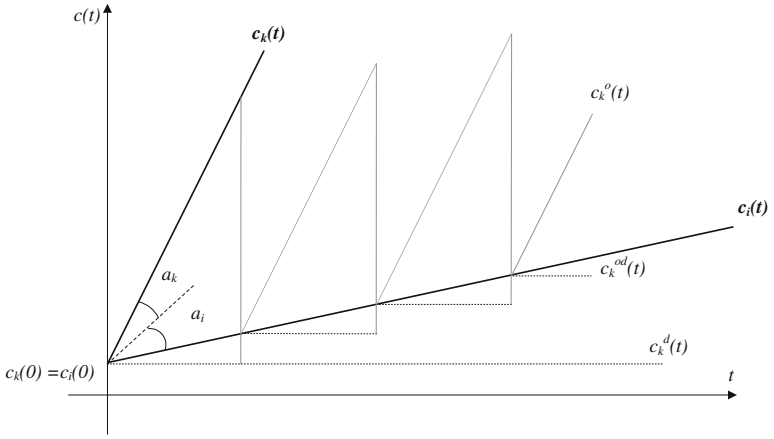


Fig. 11 Clock difference by the local clock drift

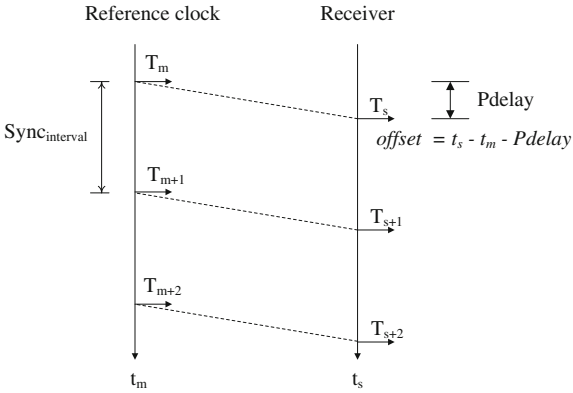


Fig. 12 Frequent time synchronization

indicates the difference between the two nodes that are to be corrected. The approach makes it possible to calculate the drift rate by using only one synchronization procedure, which dramatically reduces the number of messages that are needed for synchronization.

The period of time it takes to correct the clock drift between the two nodes is defined according to Eq. (3).

$$\mathrm{Sync}_{\mathrm{interval}} = \frac{\mathrm{err}_{\mathrm{tolerance}} \times 10^6}{f_{\mathrm{drift}}} \tag{3}$$

where f_{drift} is the drift rate of the crystal oscillator including its stability, and $\mathrm{err}_{\mathrm{tolerance}}$ is the tolerance of the time error between the two nodes.

4.4 Time Representation Error

Most of the uncertainty introduced in the network protocol stack can be reduced by using a precise time stamping unit. This means that the accuracy during time synchronization is determined by the time stamping point and the time stamping unit. However, this time stamping unit also contains a time representation error comprised of the delay and jitter in the signal. The time representation error is the difference between the actual time of an event and the nearest time value that can be represented. Figure 13 shows an example of the time representation error. Assume that the time processing unit operates at 1 MHz. The interval between the clocks is of one microsecond, and the time stamping unit detects an event at either the rising edge or at the falling edge of the clock. When the system uses the rising edge, the timer for the stamping unit is also determined at the rising edge. When an event, such as an SFD occurs at T_E, the timer of the time processing unit does not record the time at which the event occurs, but counts it at the consequent rising edge at T_{DE}. The time representation error is a maximum of one microsecond at a 1 MHz clock speed, but it is difficult to remove the time representation error unless a higher

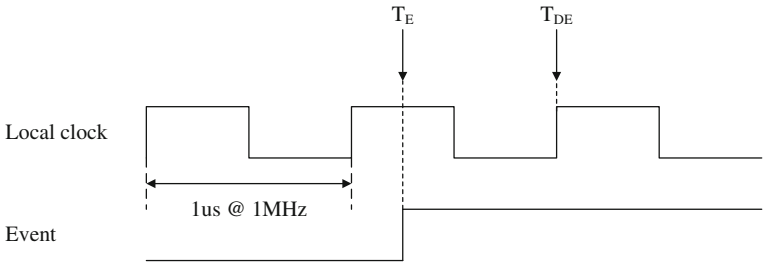


Fig. 13 Time representation error at the time stamping unit

clock speed can be used. For example, a 37.5 MHz oscillator reduces the time representation error to 26.7 ns, which is adequate for submicrosecond accuracy time synchronization. For applications that require a higher time resolution, a higher frequency clock can be used to reduce the time representation error.

The second time representation error occurs at the RF transceiver, as shown in Fig. 14. The RF transceiver requires a certain amount of time to encode and decode the message into electromagnetic waves and vice versa. The encoding time is the time required for the radio chip to encode and transform a part of the message to electromagnetic waves, and this time starts when the radio chip initiates the transfer at an idealized point. The decoding time is the time that is required for the radio chip at the receiver side to transform and decode the message from electromagnetic waves to binary data, and this time ends when the radio chip raises an interrupt indicating reception at the idealized point.

For example, the TI CC2420 radio supports the IEEE 802.15.4/ZigBee standard and has no jitter uncertainty at the transmitter side and a ± 0.125 μs uncertainty at the receiver side because it has an 8 M chip(s). It is impossible to remove the

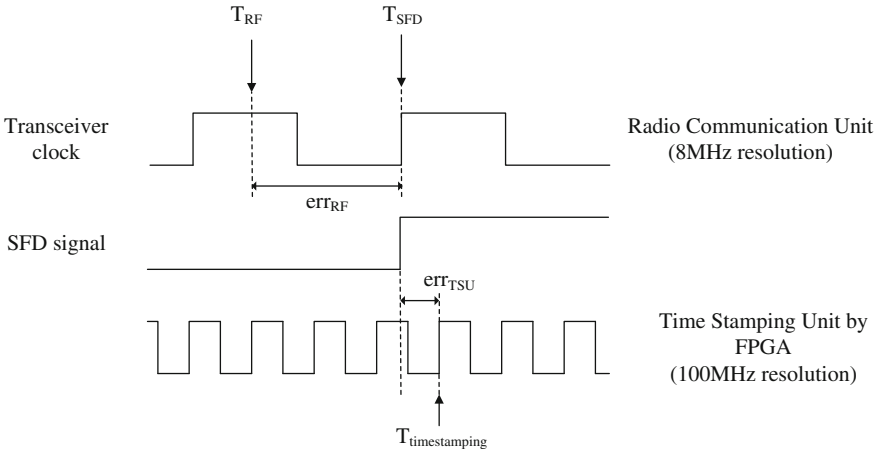


Fig. 14 Time representation error at the ZigBee transceiver

uncertainty at the receiver side. However, a many-to-many message handshake can achieve a reasonable value. Jitter can occur during encoding and decoding, and it can be reduced by using filtering methods, such as a mean, a median, and a learned function of multiple measurements.

The time representation error and the clock skew between the transmitter and the receiver cannot be eliminated. The Kalman filter is an algorithm that operates recursively on streams of noisy input data to produce a statistically optimal estimate of the underlying system state [44]. Although the Kalman filter can produce a better time estimate, we do not describe how to use a Kalman filter for time synchronization in detail. Figure 15 illustrates a general example where a Kalman filter is

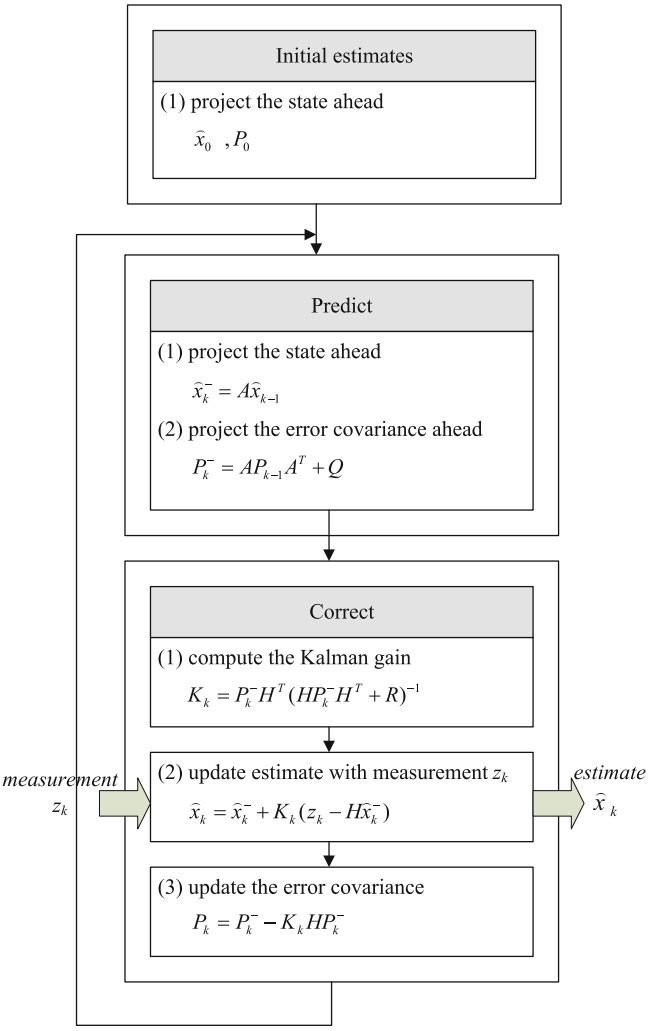


Fig. 15 The Kalman filter architecture

used for precise time synchronization. The Kalman filter is also used in NTP for accurate compensation.

We evaluate the performance of using a Kalman filter during time synchronization (Fig. 16). We minimize most of the uncertainties in the ZigBee network protocol stack and obtain precision time synchronization with sub-microsecond accuracy, as shown in Fig. 16a, where the standard deviation is of approximately 53 ns. However,

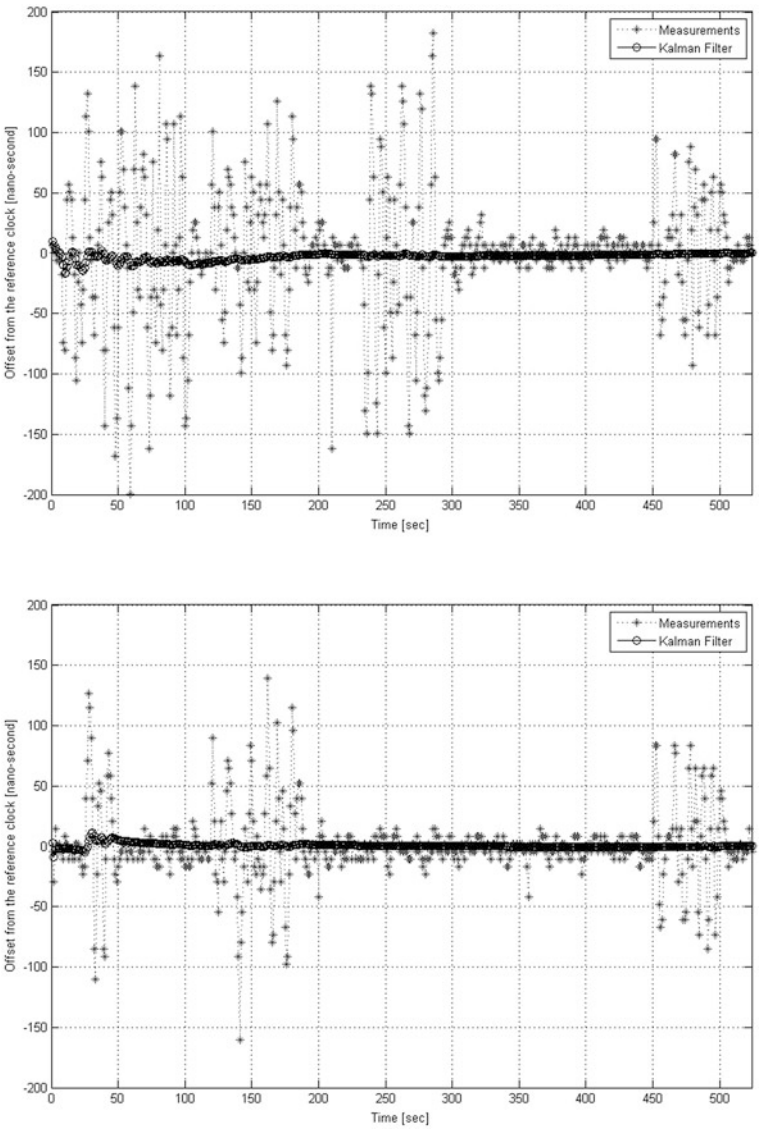


Fig. 16 Performance evaluation: precision time synchronization with and without a Kalman filter

when using a Kalman filter, we obtain a standard deviation of 2.9 ns in single-hop communications. Figure 16b exhibits the case where the time representation error is minimized down to half. The result indicates a value of approximately 29.65 ns when a Kalman filter is not used, and 1.76 ns when the Kalman filter is used. As a result, Kalman filtering is an important procedure that is necessary to reduce uncertainty during time synchronization.

5 Conclusion

This chapter discussed a wireless surveillance camera system and its corresponding time synchronization. The proposed system is decomposed into a multi-sensor environment, video and audio surveillance, and wireless sensor networks. We also described the essential constraints for wireless surveillance cameras in terms of the time synchronization. In particular, we provided an analysis of the uncertainties introduced in the ZigBee network protocol stack, and we also described how to minimize uncertainties during precision time synchronization.

Acknowledgments This work is supported by the Center for Integrated Smart Sensors funded by the Ministry of Science, ICT and Future Planning as the Global Frontier Project.

References

1. Raty TD (2010) Survey on contemporary remote surveillance systems for public safety. IEEE Trans Syst Man Cybern Part C Appl Rev 40(5):493–515
2. Huang G, He J, Ding Z (2008) Wireless video-based sensor networks for surveillance of residential districts. Lect Note Comput Sci 4976:154–165
3. Aruba networks (2011) White paper: using wireless mesh networks for video surveillance version: 1. http://www.arubanetworks.com
4. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54:2787–2805
5. Cho H, Baek Y, Kyung C-M (2014) Wireless video sensor network platform and its application for public safety. In: IEEE international conference on embedded software and systems, Aug 2014
6. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. Comput Netw 38:393–422
7. Akyildiz IF, Melodia T, Chowdhury KR (2006) A survey on wireless multimedia sensor networks. Int J Comput Telecommun Netw 51(4):921–960
8. Akyildiz IF, Melodia T, Chowdhury KR (2007) Wireless multimedia sensor networks: survey. IEEE Wirel Commun 14(6):32–39
9. Karlsson J (2010) Wireless video sensor network and its applications in digital zoo. Doctoral thesis of UMEA University
10. Firetide Inc. (2014) Hot port mesh. http://www.firetide.com
11. Strix Systems (2009) White paper: wireless mesh networks for distributed video surveillance. http://www.strixsystems.com

12. Raniwala A, Chiueh T (2005) Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In: INFOCOM 2005, pp. 2223–2234
13. Yang S-C, Yoon M-K, Kim D-H, Kim J-D (2010) Implementation of a multi-radio, multi-hop wireless mesh network using dynamic WDS based link layer routing. In: International conference on information technology: new generations (ITNG)
14. Freescale (2014) i.MX 6 datasheet. http://www.freescale.com
15. OpenWRT (2014) OpenWRT documentation. http://openwrt.org
16. Ubiquiti Networks (2014) SR71 datasheet. http://www.ubnt.com
17. STMicroelectronics (2014) STM32F407 datasheet. http://www.st.com
18. FreeRTOS (2014) http://freertos.org
19. Open Mesh (2014) B.A.T.M.A.N. advanced documentation overview. http://www.open-mesh.org
20. Mills DL (1992) Network time protocol (version 3) specification, implementation and analysis, RFC 1305
21. IEEE 1588-2008 (2008) IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. IEEE Instrumentation and Measurement Society
22. Yicka J, Mukherjeea B, Ghosal D (2008) Wireless sensor network survey. Comput Netw 52:2292–2330
23. Dong J, Gu L, Zheng C (2011) Research on fault-tolerant strategy of time synchronization for industrial wireless sensor network. In: Proceedings of the 3rd international conference on measuring technology and mechatronics automation, Shanghai, China, pp 1146–1149, 6–7 Jan 2011
24. Rhee IK, Lee J, Kim J, Serpedin E, Wu YC (2009) Clock synchronization in wireless sensor networks: an overview. Sensors 9:56–85
25. Elson J, Romer K (2003) Wireless sensor networks: a new regime for time synchronization. ACM Comput Commun Rev 33:149–154
26. Sundararaman B, Buy U, Kshemkalyani AD (2005) Clock synchronization for wireless sensor networks: a survey. Ad Hoc Netw 3:281–323
27. Sichitiu ML, Veerarittiphan C (2003) Simple, accurate time synchronization for wireless sensor networks. In: Proceedings of the 2003 IEEE wireless communications and networking, New Orleans, LA, USA, 20 March 2003
28. Cox D, Jovanov E, Milenkovic A (2005) Time synchronization for Zigbee networks. In: Proceedings of the 37th annual southeastern symposium on system theory, Tuskegee, AL, USA, pp 135–138, March 2005
29. Noh K, Serpedin E, Qaraqe K (2008) A new approach for time synchronization in wireless sensor networks: pairwise broadcast synchronization. IEEE Trans Wirel Commun 7:3318–3322
30. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E (2002) Wireless sensor networks: a survey. Comput Netw 38:393–422
31. Lim H, Kim C (2001) Flooding in wireless ad hoc networks. Comput Commun 24:353–363
32. Maroti M, Kusy B, Simon G, Ledeczi A (2004) The flooding time synchronization protocol. In: Proceedings of the 2nd international conference on embedded networked sensor systems, SenSys 2004, Baltimore, MD, USA, pp 39–49, 3–5 Nov 2004
33. Elson J, Girod L,. Estrin L (2002) Fine-grained network time synchronization using reference broadcasts. In: Proceedings of the fifth symposium on operating systems design and implementation (OSDI), Boston, MA, USA, pp 147–163, 9–11 Dec 2002
34. Palchaudhuri S, Saha AK, Johnsin DB (2004) Adaptive clock synchronization in sensor networks. In: Proceedings of the international symposium on information processing in sensor networks, Berkeley, CA, USA, 26–27 April 2004
35. Ganeriwal S, Kumar R, Srivastava MB (2003) Timing-sync protocol for sensor networks. In: Proceedings of the 1st international conference on embedded networked sensor systems, SenSys 2003, Los Angeles, CA, USA, pp 138–149, 5–7 Nov 2003
36. Dai H, Han R (2004) Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks. ACM SIGMOBILE Mob Comput Commun Rev 2004(8):125–139

37. Greunen J, Rabaey J (2003) Lightweight time synchronization for sensor networks. In: Proceedings of the second ACM international conference on wireless sensor networks and applications, WSNA 2003, San Diego, CA, USA, pp 11–19, 19 Sept 2003
38. Ye Q, Zhang Y, Cheng L (2005) A study on the optimal time synchronization accuracy in wireless sensor networks. J Comput Netw 48:549–566
39. Weibel H, Bechaz D (2004) Implementation and performance of time stamping techniques. In: Proceedings of the 2004 conference on IEEE 1588, Gaithersburg, MD, USA, 27–29 Sep 2004
40. Cho H, Jung J, Cho B, Jin Y, Lee SW, Baek Y (2009) Precision time synchronization using IEEE 1588 for wireless sensor networks. In: Proceedings of the IEEE international conference on computational science and engineering, Vancouver, BC, Canada, pp 579–586, 29–31 Aug 2009
41. Texas Instrument (2013) CC2420 datasheet. http://www.ti.com
42. Ren F, Lin C, Liu F (2008) Self-correcting time synchronization using reference broadcast in wireless sensor network. IEEE Wirel Commun 15:79–85
43. Song P, Shan X, Li X, Qi G (2009) Highly precise time synchronization protocol for ZigBee networks. In: Proceedings of the IEEE/ASME international conference on advanced intelligent mechatronics 2009 (AIM2009), Singapore, 14–17 July 2009
44. Welch G, Bishop G (2006) An introduction to the Kalman filter, TR 95-041, University of North Carolina