Clock Synchronization Over IEEE 802.11—A Survey of Methodologies and Protocols

Aneeq Mahmood, *Member, IEEE*, Reinhard Exel, Henning Trsek, *Member, IEEE*, and Thilo Sauter, *Fellow, IEEE*

***Abstract*—Just like Ethernet before, IEEE 802.11 is now transcending the borders of its usage from the office environment toward real-time communication on the factory floor. However, similar to Ethernet, the availability of synchronized clocks to coordinate and control communication and distributed real-time services is not a built-in feature in WLAN. Over the years, this has led to the design and use of a wide variety of customized protocols with varying complexity and precision, both for wired and wireless networks, in accordance with the increasingly demanding requirements from real-time applications. This survey looks into the details of synchronization over IEEE 802.11 with a particular focus on the infrastructure mode which is most relevant for industrial use cases. It highlights the different parameters which affect the performance of clock synchronization over WLAN and compares the performance of existing synchronization methods to analyze their shortcomings. Finally, it identifies new trends and directions for future research as well as features for wireless clock synchronization which will be required by the applications in the near future.**

***Index Terms*—Accuracy, clock synchronization (CS), IEEE 802.11, precision, timestamping, wireless communication, WLAN.**

I. Introduction and Motivation

THE requirement for precision clock synchronization (CS) has become more and more pertinent as communication technologies have moved from inherently synchronous networks, such as time-division multiplexing networks [1] and traditional bus-driven and fieldbus architectures [2], toward packet-based networks. This has eventually led to the use of Ethernet as the basis for communication between distributed devices in automation networks. One limitation of Ethernet has been its inability to support real-time communication. To introduce a sense of timeliness in communication and control over Ethernet, CS support by means of dedicated network messages and protocols has been a prerequisite. At first, existing CS protocols for computer networks such as the network time protocol (NTP) [3] have been used for this purpose, but their limitation in accuracy and precision has led to the design of customized protocols better suiting the needs for distributed real-time systems. Nowadays, IEEE 1588 Precision Time Protocol (PTP) [4] has become a de facto standard for CS in various applications domains such as industrial automation, telecommunication, and entertainment [5]–[7], owing to its simple structure and ability to achieve high CS performance in nanosecond range using hardware timestamping and dedicated networking elements [8].

The transition of network-based communication from wired toward wireless has presented not only cost reductions by avoiding cabling costs, but also offered flexible network topologies and mobility. One such technology, which extends beyond its normal usage in offices, is IEEE 802.11 wireless local area network (WLAN) [9]. The infrastructure mode of WLAN, which allows communication between stations (STAs) via a centralized entity called access point (AP), fits perfectly for usage in centralized architectures employed in applications as industrial automation [10] and smart grids [11], [12]. Moreover, it has opened the possibility for niche usages, such as in wireless localization on the factory floor [13]. In addition to WLAN, other solutions for wireless communications also exist on the factory floor, such as IEEE 802.15.4 [14] as physical layer, e.g., ISA100.11a [15] and WirelessHART. However, the possibility to cover larger areas along with high data rates makes WLAN a more attractive proposition.

Like Ethernet, WLAN also has its roots in the IT world and thus, offers limited only real-time capabilities [16]. This can largely be attributed to its contention-based carrier sense multiple access with collision avoidance scheme in the medium access control (MAC) layer. The deficiencies in the channel access scheme in the MAC layer are an important research topic to make WLAN a completely viable candidate for industries [17]. Especially, time division multiple access (TDMA)-based MAC schemes are of key importance, because they avoid random delays when accessing the shared wireless medium. An example of such a TDMA-based communication protocol for IEEE 802.11 above MAC layer is presented in [18]. In [19] and [20], Trsek *et al.* have proposed TDMA-based access with modifications directly in the MAC layer. The major prerequisite

Manuscript received June 1, 2016; revised September 25, 2016; accepted November 8, 2016. Date of publication December 13, 2016; date of current version April 18, 2017. Paper no. TII-16-0500. *(Corresponding author: Aneeq Mahmood.)*

A. Mahmood is with the Centre for Integrated Sensor Systems, Danube University Krems, 3500 Krems an der Donau, Austria (e-mail: aneeq.mahmood@donau-uni.ac.at).

R. Exel is in Vienna, Austria (e-mail: reinhard.exel@gmail.com).

H. Trsek is with the rt-solutions.de GmbH, 50968 Köln, Germany (e-mail: trsek@rt-solutions.de).

T. Sauter is with the Centre for Integrated Sensor Systems, Danube University Krems, 3500 Krems an der Donau, Austria, and also with the Institute of Computer Technology, Vienna University of Technology, 1040 Wien, Austria (e-mail: thilo.sauter@donau-uni.ac.at).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2016.2629669

1551-3203 © 2016 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

TABLE I
REAL-TIME CLASSES AND THEIR REQUIREMENTS [22], [23]

Types of application	Real-time requirements		Real-time
	Latency	Jitter	class
Monitoring and diagnostics (PLC to PLC)	10–100 ms	–	1
Process control (PLC to distributed IOs)	1–10 ms	$\leq$1 ms	2
Motion control	$<$1 ms	$\leq$1 μs	3

for these TDMA-based approaches is accurate CS in the range of $\leq$1 μs.

In this context, the achievable accuracy of the CS has a direct influence on the industrial application. Hence, the temporal requirements of the application must also be considered. According to [21], different types of industrial applications must be supported by industrial communication networks, such as control, or monitoring and diagnostics. The generic real-time classes for both types of applications are shown in Table I listing their specific temporal requirements as well as the specified real-time class as defined in [22].

For communication beyond infrastructure mode of WLAN, pure ad-hoc [24] or mesh-based industrial networks over multiple hops [25] can also be considered. However, hybrid networks, with WLAN segments in infrastructure mode attached to a wired network [26], are more commonly used in industries, and act as an intermediate evolutionary step toward a network with a fully wireless backbone [10], [27]. For such networks as well, there is a need to provide temporal consistency to enable a synchronous communication between the wired and the wireless entities of the system. Hence, a global time base is required, which ensures integration of wireless subsystems with the base wired network to enable a seamless real-time performance [19]. This ultimately requires availability of synchronized schemes for not only wired, but also wireless communication.

To date, there are several approaches to provide CS over WLAN in the infrastructure mode. Some of them are described by IEEE 802.11, while others have been designed specifically for a particular application. The goal in this study is to investigate the general issues of providing accurate and precise CS for IEEE 802.11 and to identify the factors which contribute to the performance of wireless CS, along with their impact. Then, this work will use these factors to elaborate existing CS methods and protocols, and discuss their advantages and shortcomings when deployed in a WLAN. To this end, this work will also analyze the use of protocols designed for wired networks over wireless as this addresses the potential of achieving CS homogeneity, in terms of protocol and performance, for the entire system.

This paper is structured as follows: Section II describes the basics of packet-based CS; Section III describes the various parameters affecting the CS performance; Section IV analyzes the CS methodologies which are already part of IEEE 802.11; non-IEEE 802.11 CS protocols for WLAN are described in Section V; Section VI summarizes and compares the different CS methods; Section VII provides an outlook on the future research activities for CS in IEEE 802.11.

II. BASIC PRINCIPLES AND DEFINITIONS FOR CS

A clock can be termed as a device to keep time and measure time intervals. A simple clock consists of an oscillator, whose frequency establishes a measure of time together with a counting entity, called timer. In an ideal case, this oscillation frequency is constant over an infinite time, so that for an ideal clock C, $dC(t)/dt = 1\,\forall t$, and we define the reference clock C_r as $C_r(t) = t$. In reality, the reference clock is highly stable, yet not an arbitrary fine-grained time piece with exactly unity rate. On the other hand, modern devices use quartz-crystal oscillators as their frequency source, which are low cost, but show time-variant behavior [28]. Thus, if a jth clock C_j in the network increments at the rate $a_j \approx 1$ and has an offset b_j with respect to the reference clock, its clock reading can be expressed by

$$C_j(t) = a_j t + b_j. \tag{1}$$

The goal of CS is to compare the time between C_j and the reference clock so that the clock error ϵ is minimized, where

$$\epsilon = C_j(t) - t. \tag{2}$$

To achieve this goal in packet-based networks, periodic packet exchanges need to take place between devices D_j, containing C_j, and D_r containing the reference clock. The device D_r containing the reference clock takes a snapshot of its time (termed timestamp), and stores it in a synchronization packet, just before transmitting the packet. The device D_j containing C_j also draws a timestamp upon receiving this packet. The two methods of drawing timestamps for CS are as follows:

1) *Hardware Timestamping:* The timestamps are drawn at the physical (PHY) or MAC layer by the hardware itself.
2) *Software Timestamping:* The timestamps are generated at a higher protocol layer by software, for instance, within the device's operating system (OS).

Based on the timestamps from C_r and C_j, following methods are used to perform clock corrections:

1) *Offset Correction:* Only the offset b_j at time t between C_r and C_j is corrected to perform CS.
2) *Rate Correction:* Only rate a_j of C_j is corrected periodically so that it follows C_r.
3) *Offset and Rate Correction:* Periodically, both a_j and b_j are corrected together to synchronize C_j to C_r.

Once clocks are corrected, the CS performance is measured using certain metrics. The two commonly used metrics for comparing performance of CS protocols are as follows:

1) *Accuracy:* It is determined by calculating the mean value of ϵ over time.
2) *Precision:* It is measured as the standard deviation of error ϵ from its mean value. The lack of precision in CS can also be referred to as CS jitter.

It should be noted that other metrics to analyze CS performance do exist, for example, maximum time interval error in telecommunication networks [5]. These metrics are often application specific, i. e., metrics used to analyze one application may not be used for other applications. However, accuracy and precision are deemed sufficient to compare CS protocols.

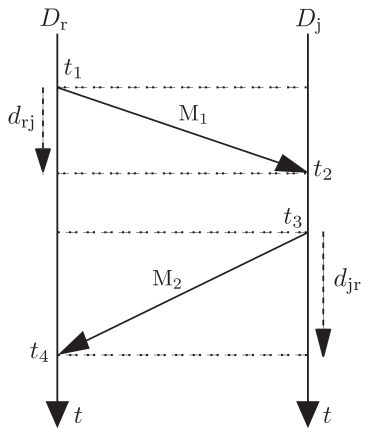


Fig. 1. Two-way packet exchange for synchronization.

The determination of the rate a_j is straightforward. It involves determining the ratio of a duration captured by the local and reference clock, such as two messages sent from D_r to D_j. The measure of clock offset b_j, based on difference of their timestamps, also includes the propagation time of the packet between them. The propagation delay is commonly removed by two-way packet exchanges as shown in Fig. 1. At time t_1, a timing packet M$_1$ exits D_r; this packet is received by D_j at t_2; the propagation delay between D_r and D_j is d_rj. Later at time t_3, D_j sends a packet M$_2$ which is received by D_r at t_4; the propagation delay between D_r and D_j is d_jr. Based on this information,

$$t_2 = t_1 + d_\mathrm{rj} + b_j \tag{3}$$

$$t_4 = t_3 + d_\mathrm{jr} - b_j. \tag{4}$$

The offset b_j can be determined as

$$b_j = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} - \frac{d_\mathrm{rj} - d_\mathrm{jr}}{2}. \tag{5}$$

Thus, if propagation delays are symmetric, i. e., $d_\mathrm{jr} = d_\mathrm{jr}$, then offset calculations will lead to bias-free synchronization. However, if the delays are not symmetric (as shown in Fig. 1), a bias is present due to the asymmetry given as $(d_\mathrm{rj} - d_\mathrm{jr})/2$. Unfortunately, the asymmetry cannot be resolved, no matter how many synchronization rounds are performed. In general, for wireless networks asymmetry can occur, if the channel reciprocity does not apply (e.g., different carrier frequencies, Doppler shifts between the devices). However, in case of quasi-static wireless devices with the same baseband modulations and carrier frequencies in both directions, channel reciprocity is given and so the asymmetry vanishes [29]. Apart from this intrinsic type of asymmetry, large asymmetries can be introduced by lacking or improper calibration of delays, as will be outlined in Section III-C.

III. Performance Aspects for Wireless CS

Fig. 2 highlights different elements impairing the CS performance over the wireless channel. When a WLAN reference device D_r (i.e., the AP) sends a packet for synchronization, it is timestamped by the timestamper block being driven by the oscillator, to provide a snapshot of the clock. The timestamper can either be a software (in the OS), or a hardware one built inside the PHY layer hardware. As the processing delays of hardware timestamping solutions can be engineered to be constant, the timestamps can have much lower uncertainty (jitter) than software solutions. In contrast, generating timestamps by software means creates indeterministic delays due to scheduling, caches, concurrency. The egress timestamps is put inside the packet "on the fly" as it passes through the PHY. If this is not supported, another "follow_up" packet will be required to transmit the egress timestamp.

Upon exiting D_r, the packet traverses the medium, experiences a propagation delay, and reaches the receiving device D_j, i. e., an STA. There, the packet is again timestamped through hardware or software means. This ingress timestamp and the egress timestamps are eventually used by a clock adjustment block to correct the rate and offset of the clock. Based on this discussion, CS performance depends upon

1) the oscillator steering the clock;
2) the quality of timestamps;
3) determination of propagation delays; and
4) how and how often clock adjustments is performed.

Additionally, the packet drops on the wireless channel are also a parameter which affects the quality of wireless CS. The impact of these aspects on CS is discussed below.

A. Oscillator Impact

As already stated, modern day devices use quartz crystal oscillators (XOs) as clock source. Yet, XOs diverge from an ideal clock behavior because of factors such as voltage, temperature, mechanical stress, and structure affecting oscillation frequency [30], [31]. Based on [32], the frequency ν of an XO at time t can be written as

$$\nu(t) = \nu_o + \Delta\nu + Q\nu_o t + \Phi(t) \tag{6}$$

where ν_o is the nominal frequency of the oscillator, $\Delta\nu$ is the frequency skew of the oscillator, Q is the drift rate of the oscillator owing to ageing and environmental effects, and Φ is the random noise component. A statistical measure for the stability of an oscillator is the Allan variance [33]. It is a function of sampling interval τ, and over τ-spaced N samples, it is given by

$$\sigma_y^2(\tau) = \frac{1}{2(N-1)} \sum_{i=1}^{N-1} (y_{i+1} - y_i)^2 \tag{7}$$

where $y_i = (\nu - \nu_o)/\nu_o$ is the instantaneous frequency error at the sampling instant i. From the Allan variance of an XO with advertised ± 50 parts per million (ppm) frequency skew [34], it is shown that for small τ (up to about 1 s) the long-term noise components are small, and the frequency changes are typically in the range of parts per billion (ppb), i.e., around three orders of magnitude smaller than the advertised mean frequency skew. This implies that the role of oscillator instability (in the ppb range) is negligible with respect to the mean frequency skew when CS is performed only via offset correction. As CS by rate correction inherently compensates for the skew, it performs on average three orders of magnitude better than offset

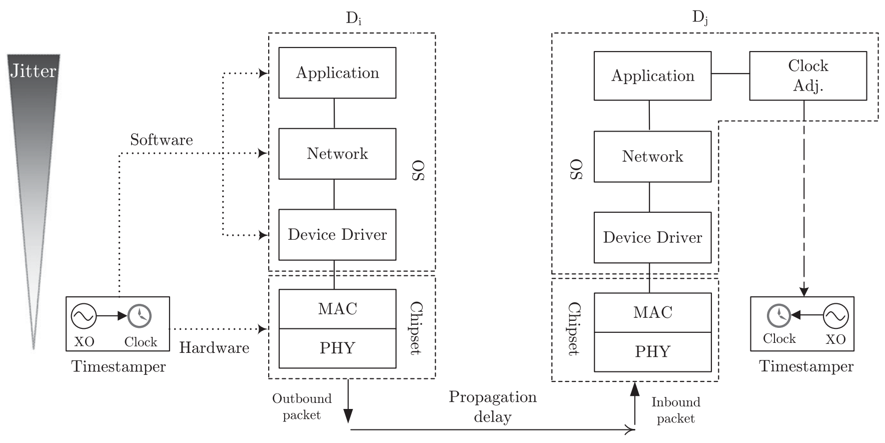


Fig. 2. Various factors affecting packet-based synchronization.

correction. A combination of both rate and offset correction will compensate for both offset and skew, and will lead only to minor improvements in precision, yet fast synchronization settling times.

It should also be noted that when analyzing the impact of oscillators from CS's perspective, the interesting aspect is not the short-term variance of the frequency error from (7) itself, but the timing error Δt which has been accumulated during every τ since the beginning of synchronization. Mathematically, the error ϵ_τ from the beginning of synchronization till time t can be written as

$$\epsilon_\tau(t) = \int_0^t \Delta t(\tau)\, d\tau. \tag{8}$$

Using the clock model of (1), the slave time at t will be

$$C_j(t) = a_j t + b_j + \epsilon_\tau(t). \tag{9}$$

In reality, the skew of the oscillator is also not constant. Nevertheless, if the clock adjustment mechanism takes care of the initial clock offset and compensates for the skew, then the remaining error source, to be minimized for CS, is the accumulated phase noise ϵ_τ. The characteristics of ϵ_τ can be analyzed through its variance σ^2, which is obtained by integrating Allan variance twice, and is equal to $\tau^2\sigma_y^2(\tau)/3$ as shown in [35].

B. Timestamping Techniques

For timestamping in CS, a timer with high resolution should be chosen. The variance of uniform quantization noise with a timer resolution of T_n seconds is $T_n^2/12$. Hence, a higher resolution will offer less quantization noise. Moreover, the timestamp at both the sender and the receiver should be drawn with respect to a common reference plane (RP) to avoid artificial asymmetry in offset calculation. For instance, timestamps can be taken when the start-of-frame delimiter, which follows the preamble of wireless frame, passes the RP. For wireless communication, the RP can be chosen to be the antenna connector. However, as the demodulation and decoding of a wireless frame requires some processing delay, timestamps are taken at a later point in time and then corrected for the incurred delay. This is simpler for hardware (HW) timestamping, as compared to software (SW) timestamping, as timestamps are drawn in the PHY close to the RP.

1) Hardware Timestamping: The hardware timestamping performance of current devices for Ethernet is mainly dictated by the sampling frequency of the clock that detects the asynchronous arrival of a frame. With the typical clock frequencies of IEEE 1588 implementations, timestamping resolutions in the nanosecond range are typical [36]. However, this simplification to timestamp quantization is only partially valid for wireless timestamping as additional factors impair the timestamps. These include the signal-to-noise ratio, carrier- and baseband frequency shifts, multipath reflections, nonlinearities, the temporal rate of change of all these factors, and mainly the selected timestamping method. A simple low-level timestamping method could be, for example, a correlator unit for the SFD as every frame starts with the SFD. Then, a timestamp could be taken whenever the magnitude of the correlator exceeds a certain threshold level, a method often used for Ultra-Wideband receivers. However, noise and the other impairments have an influence on the correlator output magnitude and with that on the timestamping instant. Even worse, depending on the selected threshold level, timestamps for noise-only signals may be drawn (false positives), or timestamps for actual frames might be missed (false negatives). Instead of this simple but potentially erroneous approach, a good wireless timestamping approach should minimize the errors, e.g., by taking all the known properties of the signal to be timestamped into account. Hence, if the modulation parameters are taken into consideration, and when averaging over the entire frame, much better timestamping performance can be achieved than with a simple threshold detector. The WLAN receiver architecture, presented in [37] and [38], follows this approach and can deliver timestamps quantized with 88.7 ps and a timestamping standard deviation of only 50 ps. Due to multipath propagation on the wireless channel, the timestamps can still be biased. Yet, for synchronization applications the bias is almost symmetric (due to channel reciprocity) and therefore absorbed in the round-trip delay compensation.

2) Software Timestamping: For SW timestamping, it is significant to choose a stable clock source which has less variation in access time for drawing timestamps. For Unix-based OSs, several clock sources along with their accessibility and stability have been studied in [39]. Broomhead *et al.* have mentioned that the time stamp counter (TSC), the advanced configuration and power interface (ACPI) timer, and the high precision event timer (HPET) are available in modern computers for keeping time. From the analysis in [39], it can be seen that all timers show similar stability behavior under varying temperature and CPU loads, and are equally suitable for CS. However, the 32-bit variants of both HPET and ACPI timers, present in certain hardware configurations, make them prone to overflow several times per second. The 64-bit TSC in modern computers does not have this issue and is found to offer the smallest access time due its location inside the CPU. On a 3 -GHz CPU, the median access time for the TSC is 240 ns with a latency spread of only 80 ns.

It should be noted that the analysis for the TSC in [39] is carried out for a single-core CPU without frequency scaling. While this setup has shown that the TSC is stable (over a week long testing phase), frequency scaling and the use of multicore CPU architectures pose the threat of varying increment rate for the TSC because of varying frequency and potential frequency mismatch between different cores, respectively. Some strategies for using the TSC in multicore architecture have been mentioned in [40], [41], which involve maintaining TSC offsets or frequency skew between the different cores, but this will make the use of the TSC very complex. Nevertheless, in a single-core CPU with fixed nominal frequency, the TSC offers a stable and fast time source for software timestamping.

Another important choice for SW timestamping is to decide where to timestamp the packet. From Fig. 2, the device driver is the earliest point in the OS where SW timestamping can be performed to minimize timestamping uncertainty. Within the driver, a timestamp can be drawn inside the interrupt service routine (ISR), which is the earliest time the OS is notified of an egress or ingress packet. In this case, egress timestamps are drawn after transmission, and would require an additional packet to be sent from the sender to the receiver. However, such egress timestamps do not include the random channel access delay which occurs before the transmission on the channel. This significantly reduces the timestamping jitter. Nevertheless, with timestamping in the ISR, a new source of randomness is introduced, namely the interrupt handling delay (IH) delay, which varies with the CPU and the OS. Ferrari *et al.* [42] and Mahmood *et al.* [43] have provided experimental setups for calibrating IH delays for wired and wireless networks, respectively. Nevertheless, it is difficult to put an upper bound on this delay as it changes with CPU load and behavior of the OS. Still, OSs with real-time extensions [44], can put a maximum limit on interrupt handling delays and thus, can improve the quality of SW timestamps drawn in the ISR at both the transmitter and the receiver.

C. Propagation Delay Compensation

To provide accurate CS, any bias originating from asymmetry in (5) should be reduced to zero. This can be achieved when the propagation delays on the wireless channel are correctly

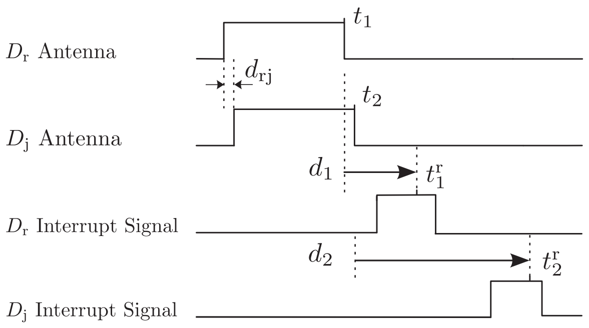


Fig. 3. Delays inside the WLAN chipset and the operating system [43].

calculated, and egress and ingress timestamps are compensated to reflect the time at the RP. This is a particular issue for software timestamping, where the delay from the RP varies for different frame sizes and different modulation and coding schemes (MCS). Thus, these differences for M_1 and M_2 in Fig. 1 should be considered for determining the correction value to the RP.

Thus, if S is the packet size, and R is the data rate over the channel, then the propagation delay d could be calculated as $d = S/R$ as cited in [45]. However, IEEE 802.11 packets do not have a fixed data rate throughput; the packet preamble and headers are sent with the lowest data rate, whereas the payload can be sent with a higher rate. Moreover for packets using OFDM-based modulation schemes, frame lengths have to be an integer multiple of 4 μs which leads to addition of extra bytes to the payload. As a result, the duration of the physical layer frame is not a linear function of the payload divided by the transmission rate as indicated in [46]. Hence, the simplest way to avoid propagation delay asymmetry is, if M_1 and M_2 are of same size and use the same data rate. Otherwise, the effect of different size and data rate on the channel must be taken into account.

Regarding processing delays in the chipset, compensating them with respect to the RP is not trivial with SW timestamping inside the ISR in the driver. Fig. 3 shows these processing delays inside the device when a packet is sent from D_r to D_j and is timestamped in the ISR. The timestamps (t_i for $i = 1, \ldots, 4$) from (5) need to be adjusted with respect to the RP. Based on Fig. 3, $t_1 = t_1^r - d_1$ and $t_2 = t_2^r - d_2$ where superscript "r" on t_1 and t_2 indicates the timestamps drawn away from the RP, and d_1 and d_2 indicate the IH and timestamping delay at the transmitter D_r and the receiver D_j, respectively. Similar equations can be written for t_3 and t_4, using delays d_3 and d_4, so that (5) can be modified to

$$\tilde{b_j} = b_j + \frac{(d_3 + d_4) - (d_1 + d_2)}{2}. \tag{10}$$

The delays (d_q for $q = 1, \ldots, 4$) contain both the processing time in the chipsets and the IH delays, the latter being not constant. However the processing delays are deterministic and can be used in (10). The processing delays in d_1 and d_3 exist during transmission and have shown to be constant for a particular WLAN chipset [47]. The delays d_2 and d_4 exist during packet reception, and depend mainly upon packet size, and direct memory access (DMA) buffer size, and DMA interface speed between the chipset and the host. Hence, for a particular

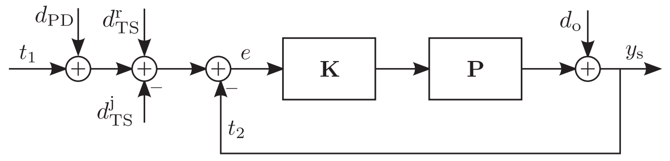


Fig. 4. Block diagram of the PI controller-based clock adjustment procedure.

chipset and specific interface between the chipset and the host device, these delays become deterministic and can be estimated (cf., [43, p. 4]) to remove asymmetries coming from processing delays inside the chipsets. An alternative to estimate these delays is to ensure that same chipsets are being used at the AP and STAs, and the CS traffic employs the same data rate. In this way, the delays inside the chipsets are identical and cancel out in the round trip delay calculation and do not contribute to propagation delay asymmetry.

D. Clock Adjustment Procedure

The clock adjustment procedure involves steering the local clock to the reference clock to minimize clock offset between them. In general, linear least-square regression approach can be used for clock adjustments, both when the propagation delay between the two devices is ignored [48] and when the propagation delay is calculated using the two-way packet exchange as shown in [49]. In addition, there exists other statistical or control-theory-based approaches to perform synchronization. A summary of statistical signal processing-based techniques in wireless sensor networks including the Kalman Filter (KF) or Particle Filter is provided in [50]. Use of Wiener Filters for synchronization has been carried out in [51]. A feedback control mechanism based on a proportional integral (PI) controller is shown in [5]. All these approaches use statistical or measurement-based modeling of noise sources in synchronization and try to minimize them. This is shown in Fig. 4 for a PI-controller-based feedback mechanism where clock D_j synchronizes to D_r.

The timestamps from D_r and D_j are disturbed by timestamping noise d^r_{TS} and d^j_{TS} from both clocks, respectively, and any variations in propagation delay estimation are exhibited by d_{PD}. The timestamping noise contain overall jitter accumulated during the drawing of timestamps including the quantization error. The error e, obtained from the difference of this disturbed t_1 and t_2 is fed into the PI controller K. Its output is fed into the plant P which integrates the rate output coming from K. The output of P is superimposed by noise from the D_j's oscillator d_o, which gives the output clock y_s. In the absence of dead-time in the feedback loop, Giorgi and Narduzzi in [52] have shown that the output time is affected by aforementioned disturbances such that in the z-domain

$$Y_s(z) \approx \frac{P(z)K(z)}{1+P(z)K(z)}(D_{PD} + D^i_{TS} - D^j_{TS}) + \frac{D_o}{1+P(z)K(z)} \quad (11)$$

where the capitalization of terms in (11) represents the z-transform to the equivalent time domain signals and symbols shown in Fig. 4, e.g., $Z\{d_o\} \rightarrow D_o$. The goal is to choose the proportional and integral coefficients (k_p and k_i) of the PI controller K to minimize the errors shown in (11). A simulation-based analysis from [53] shows that adaptive k_p and k_i values are required to adjust and adapt to varying timestamping jitter (including varying PDs), oscillator noise, and synchronization rate. In general, smaller k_p and k_i will give optimum performance for a stable oscillator, as it will reduce the noise bandwidth of the control loop. Larger PI coefficients will provide optimum performance for accurate timestamps and unstable oscillators. Moreover, choosing a smaller value for k_p and k_i results in poles with large magnitudes in (11) which lead to larger loop settling times. Therefore, depending upon the type of application and required precision versus settling time, the right values for k_p and k_i should be chosen.

One example of using statistical methods for synchronization is via a KF, which requires knowledge about the error covariances matrices for all the disturbances shown in Fig. 4. The Kalman Gain matrix is recursively updated to find a tradeoff between oscillator noise, and timestamping and propagation delay errors in the bid to reduce minimum mean square offset and skew. In theory, error covariance matrices can be adapted during operation to take varying conditions and environments into account, hence providing inherent adaptive control for CS. However, often in Kalman-filter-based implementations such as in [54], oscillator noise variance is derived from oscillator stability data-based on the Allan variance [28], while other errors can be updated "on the fly." This makes the KF adaptive against all errors in Fig. 4 except the oscillator error. A method to address varying oscillator noise due to temperature variations is shown in [55]. This approach, however, relies on neural network-based training to provide clock offset estimate for synchronization. In general, regardless of the method used to perform clock adjustment, an adaptive methodology is needed which can take variations of the error sources, involved in synchronization, into account.

E. Impact of Synchronization Rate and Packet Loss

Another important parameter for wireless CS is how often synchronization packets need to be exchanged between the AP and STAs. The synchronization rate is chosen to minimize the combined effect of the oscillator error and quality of timestamps, along with propagation delay fluctuations. In general, a higher synchronization rate in conjunction with HW timestamps will improve CS precision. However, with SW timestamping, this would imply frequently bringing noisy timestamps into the controller. Thus, having a higher synchronization rate will not significantly improve the performance in the presence of large jitter coming from timestamps and/or from varying propagation delays [56].

On the other hand, having a lower synchronization rate suits the application as (to a certain extend) more bandwidth is available for real-time communication. However, a lower rate results in larger deviations form the reference clock. In that case, the

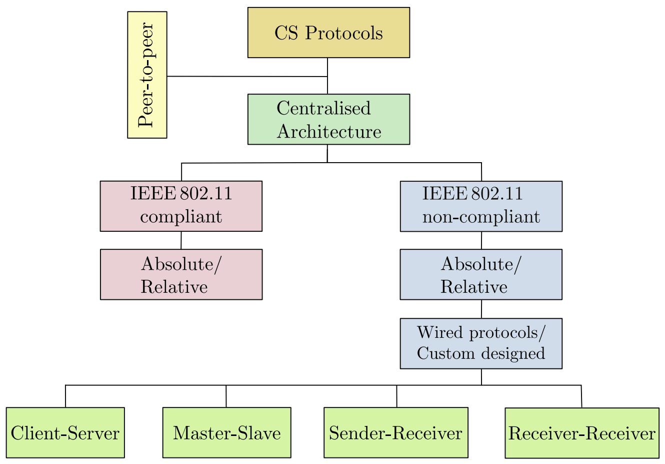


Fig. 5. Classification of CS protocols over IEEE 802.11.

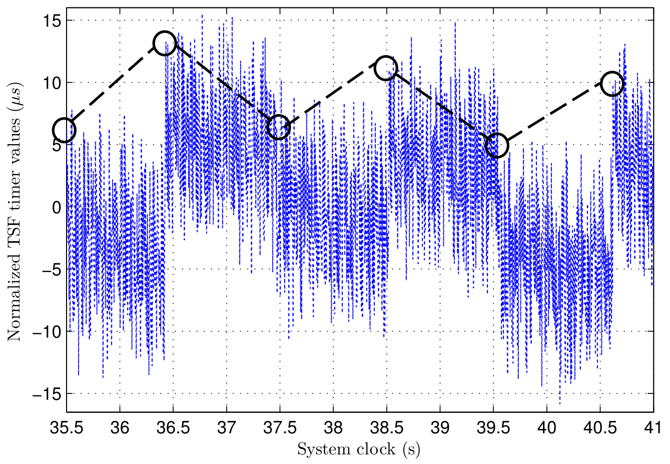


Fig. 6. Jumps in TSF timer due to periodic offset corrections [56].

overall error can be improved, e.g., by increasing the loop bandwidth by choosing a higher value of k_p and k_i for a PI controller. This will inevitably lead to lower overall precision than in the case of higher synchronization rate [53]. At very low synchronization rate (e.g., once every 10 s or more) for an XO, oscillator wander will become the dominant noise source in the control loop and use of HW or SW timestamping will become a moot point.

Packet losses on the channel naturally result in a reduced clock adjustment rate than desired, whose immediate impact is a loss in CS precision. However, a more adverse effect comes from the non-zero dead-time of the controller. Frequent packet losses on the channel lead to higher dead-times inside the control loop (regardless of using PI controller or the KF) which, in turn, can result in control loop instability and/or complete loss of synchronization. Hence, the control loop should be carefully engineered not only to minimize the impact of timestamping jitter and the oscillator phase noise in the best case, but also to be robust against missing synchronization packets. For example, the impact of successive packet losses can be minimized by tuning the controller to rely more on the oscillator than on incoming timestamps for clock correction. However, in the case of packet losses over a larger time period, oscillator wander will take place, as already discussed. In this case, both general data communication and CS will be affected.

IV. Synchronization Over IEEE 802.11

This section provides an overview and comparison of existing synchronization schemes and protocols over IEEE 802.11. A taxonomy of such schemes, which is relevant to forthcoming discussion, is described at first (see Fig. 5).

A. Classification of CS Approaches

1) *Centralized versus Peer-to-Peer (P2P) Synchronization:* In the centralized synchronization approach, a single clock in the network acts as the main time source. The clients exchange packets only with this reference clock for CS. For P2P synchronization, all clients can communicate directly with each other and there is no device acting as the reference. Thus, each device can request or share timing information with foreign devices. Within the scope of infrastructure mode of IEEE 802.11, which fits under the centralized CS approach, the protocols can be either IEEE 802.11 compliant or are custom-designed to suit underlying industrial applications. Such protocols can be further classified as follows:

2) *Absolute or Relative Synchronization:* Absolute synchronization aims at synchronizing devices in a network to an absolute reference such as the International Atomic Time (TAI) or Universal Coordinated Time (UTC), and is also called external synchronization. It demands access to an absolute time source (e.g., a receiver locked to global positioning system (GPS) [57]). Relative (or internal) synchronization is only used to ensure that synchronized devices in the network share a common timebase. It is used mostly for coordinating activities among network devices, and for making distributed measurements in a network.

3) *Wired Protocol or Custom Designed:* This classification considers whether the protocol has been originally designed for wired communication and is imported over IEEE 802.11, or it was designed specifically to be used over WLAN.

4) *Client–Server or Master–Slave Synchronization:* In the client–server approach, the client requests to be synchronized to the server which is the timing reference. In the master–slave design, the master or the reference clock is chiefly responsible for synchronizing the slaves in the network via unicast or multicast messages.

5) *Sender–Receiver (SR) or Receiver–Receiver (RR) Synchronization:* An SR synchronization approach is a simple centralized synchronization approach where the receiver synchronizes to the sender (reference). In the RR paradigm, the receivers do not synchronize to the sender time, but use a broadcast message from the sender as a

timing signal. The receivers timestamp this timing signal and then exchange the timestamp information with other receivers that have timestamped the same message. The difference between other receivers' timestamps and its own timestamp is used by a receiver to steer the local clock. Hence, all the receivers synchronize internally, and not with respect to the sender.

Based on this taxonomy, different methods to synchronize clock in IEEE 802.11 for the infrastructure mode are presented in this section.

B. Relative Synchronization in IEEE 802.11 and Its Limitations

In the infrastructure mode, the AP is the timing reference for all STAs in the BSS; the AP synchronizes the STAs using the basic timing synchronization function (TSF) method. The AP periodically broadcasts the time from its TSF timer ($\mathrm{TSF}_{\mathrm{AP}}$) in dedicated Beacon frames. The timestamp inside Beacons contains the value of the TSF timer when the first bit of the timestamp is handed to the physical layer (PHY) from the MAC-PHY interface, plus the transmission delay from the PHY to the antenna. The receiving STA shall set the value of its TSF timer ($\mathrm{TSF}_{\mathrm{STA}}$) to the timestamp from the AP present inside the Beacon. In addition to Beacons, Probe Response frames are also used to perform TSF timer synchronization. However, these frames are exchanged only when a STA joins the BSS; afterward periodic Beacons are used by the STA to remain synchronized to the AP.

It should be noted that the TSF-based synchronization does not include any propagation delay estimation for offset estimation. Therefore, synchronization in all these methods will be biased. Nevertheless, performance requirements for TSF in IEEE 802.11 are not particularly high; accuracy and precision in the range of a few microseconds are sufficient, which is still achievable with hardware timestamps from TSF timers even if no propagation delay compensation is performed. From the application usage perspective, it has not been possible to use this scheme because TSF timers have traditionally been inaccessible by the applications, and are used solely for IEEE 802.11 physical and MAC layer operations. This, however, has changed in IEEE 802.11-2012, where the standard provides necessary primitives to access the TSF timer from higher protocol layers, namely `GETTSFTIME.request` and `GETTSFTIME.confirm`.

It should also be highlighted that as the TSF method performs only offset correction and no rate corrections, the TSF timers in STAs are nonmonotonically increasing clocks due to possible time wraps into the past and future. The measurement results shown in [56, Fig. 6] show the instant of offset correction for $\mathrm{TSF}_{\mathrm{STA}}$ with O symbol, and the dashed lines highlight the nonmonotonically increasing time which is the result of offset only clock correction. Moreover, as the time error for rate imperfections increases linearly with the delay from the last offset adjustment, the rate-induced error is not constant and is nonzero mean. This error cannot be removed without rate corrections, but can be minimized by choosing a small Beacon interval.

C. Absolute Synchronization Methods in IEEE 802.11 and Their Limitations

In addition to relative synchronization, IEEE 802.11 has also defined two external synchronization mechanisms, which are described below.

1) Timing Measurement (TM) Method: The TM method is a synchronization scheme between STAs which uses two-way packet exchange to achieve end-to-end synchronization. This method is a part of IEEE 802.1AS-2011 [7], which is the standard to provide CS guidelines for time-sensitive applications in wired-wireless audio–video bridging (AVB) networks. In the TM methods, the timer used for timestamping has a resolution of 10 ns. This implies that the TM mechanism does not employ the TSF timer whose resolution is limited to 1 μs. IEEE 802.11 does not specify any additional information about the timer used for timestamping in TM frames. It is assumed that this timer is a built-in vendor-specific timer, can carry TAI or UTC time, and complies with requirements of IEEE 802.1AS-2011 compliant AVB networks. However, it is difficult to analyze the performance of the TM method because of lack of available hard- and software support. Nevertheless, with HW timestamping support and with delay compensation via two-way packet exchange, it is assumed that TM's CS accuracy will be in the range of few nanoseconds.

2) Timing Advertisement (TA) Method: This method allows an AP to share timing information from a specific timing standard such as UTC or TAI using its TSF timer. The AP broadcasts a timestamp from $\mathrm{TSF}_{\mathrm{AP}}$ and the offset between $\mathrm{TSF}_{\mathrm{AP}}$ and a local clock ($\mathrm{LC}_{\mathrm{AP}}$), which is part of the device (e.g., the system clock), in the WLAN. Upon receiving a frame containing TA-related information, a STA also draws a timestamp from its own $\mathrm{TSF}_{\mathrm{STA}}$. This receiver STA hands this timestamp, the timestamp from $\mathrm{TSF}_{\mathrm{AP}}$ in the frame, and the ($\mathrm{LC}_{\mathrm{AP}}$-$\mathrm{TSF}_{\mathrm{AP}}$) offset from the AP to its higher layers for synchronizing its own local clock ($\mathrm{LC}_{\mathrm{STA}}$) to $\mathrm{LC}_{\mathrm{AP}}$. Just like simple TSF-based synchronization, the TA method does not account for the propagation delay from the sender to the receiver, and will suffer from synchronization bias.

There are two obstacles in implementing the TA scheme, which are shown in a simulation-based analysis in [58]. First, $\mathrm{LC}_{\mathrm{AP}}$-$\mathrm{TSF}_{\mathrm{AP}}$ offset and $\mathrm{LC}_{\mathrm{AP}}$ are running at different rates, which will affect the $\mathrm{LC}_{\mathrm{AP}}$-$\mathrm{TSF}_{\mathrm{AP}}$ offset. The standard requires to synchronize $\mathrm{TSF}_{\mathrm{AP}}$ and $\mathrm{LC}_{\mathrm{AP}}$, but leaves this task to the end user. This can be achieved by periodically obtaining skew estimates between $\mathrm{TSF}_{\mathrm{AP}}$ and $\mathrm{LC}_{\mathrm{AP}}$ using, for example, a KF, and then updating the $\mathrm{LC}_{\mathrm{AP}}$-$\mathrm{TSF}_{\mathrm{AP}}$ offset. Moreover, this $\mathrm{LC}_{\mathrm{AP}}$-$\mathrm{TSF}_{\mathrm{AP}}$ offset should be put into the frame by the PHY "on the fly," just like the $\mathrm{TSF}_{\mathrm{AP}}$ timestamp during transmission. Otherwise, jitter from higher layers will be present in the offset which will severely degrade synchronization performance. However, such TA-compatible WLAN chipsets with "on the fly" offset updates are not available in the market.

Another obstacle for the TA scheme originates from the question when to draw the timestamp from $\mathrm{TSF}_{\mathrm{STA}}$ for $\mathrm{LC}_{\mathrm{STA}}$ to $\mathrm{LC}_{\mathrm{AP}}$ synchronization, as $\mathrm{TSF}_{\mathrm{STA}}$ and $\mathrm{TSF}_{\mathrm{AP}}$ also perform offset correction when, e.g., a Beacon frame carrying TA-related

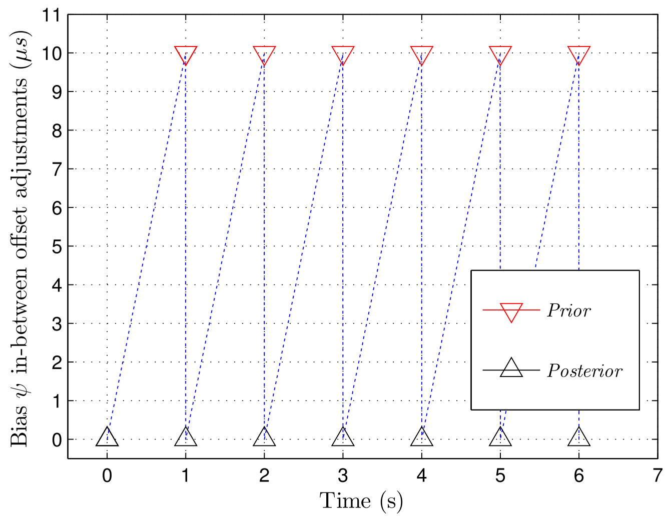


Fig. 7. Bias accumulated during offset corrections at the wireless station [58].

information is received at the STA. If ϕ is the difference between the skews of TSF_{STA} and TSF_{AP}, then a bias $\psi(t) = \phi \times (t \mod T)$ is accumulated over the entire synchronization period T. This bias is plotted for $\phi = 10$ ppm and $T = 1\,s$ in Fig. 7. If the timestamp from TSF_{STA} is drawn *post* offset correction, then ψ affecting LC_{STA}-LC_{AP} synchronization will be zero (shown by Δ in Fig. 7). If the timestamp is drawn *prior* to offset correction (shown with $\triangledown$ in Fig. 7), then $\psi = \phi T$. However, according to IEEE 802.11 ([9, cf. 10.21.1]), the timestamp from TSF_{STA} is drawn upon receiving the first byte of the timestamp from TSF_{AP} in the incoming frame and not after offset correction. Hence, the bias ϕT will be fed into LC_{STA} - LC_{AP} synchronization.

V. Non-IEEE 802.11 Protocols for CS

This section discusses different non-802.11 compliant mechanisms for CS. Based on the taxonomy in Fig. 5, such schemes which provide strictly relative CS will be discussed first. The rest of the CS methods can employ either relative or absolute synchronization.

A. Schemes Providing Only Relative Synchronization

The idea of only providing relative synchronization in a WLAN is based upon using the TSF timers at the application, and exploiting the built-in TSF method for microsecond-level accuracy. This has been made possible by the availability of open source device drivers for WLAN chipsets from several vendors [59], which provide an API for reading the TSF timer. It must be noted that the TSF method does not compensate for the propagation delay, and TSF timers in STAs are nonmonotonic, as already shown, which is undesirable for applications.

To avoid this, Mahmood *et al.* [58] proposed a new protocol called *Sync*TSF, which synchronizes LC_{AP} to its TSF_{AP} within the device, and also synchronizes LC_{STA} to its TSF_{STA}. This leads to end-to-end CS between LC_{STA} and LC_{AP} as the TSF timers are inherently synchronized through TSF. The use of the PI controller for synchronization between a TSF timer and a LC leads to smooth control and provides continuous time for the LC. A limitation of this scheme is that the accumulated bias ψ due to skew difference ϕ will be present at the STA as already shown in Fig. 7. However, as the timestamps are being continuously drawn from the TSF_{STA} for synchronizing LC_{STA}, the overall bias will be approximately $\phi T/2$. This bias in *Sync*TSF can be minimized by using a smaller T (i.e., the beacon interval in this case), but this can lead to higher packet rate on the channel. For a commonly used beacon interval of 102.4 ms, the CS accuracy of *Sync*TSF is determined to be 1.8 μs; the CS precision turns out to be 341 ns. To achieve maximum accuracy without increasing the packet rate, rate reconnection between the TSF timers and bias correction at the STA will be required.

A similar scheme is developed in [60] for CS in multimedia applications, where virtual timers are drawn from TSF_{AP} and TSF_{STA}. This method also uses open source drivers for accessing the timers and uses software timestamping. This method is affected by the noncontinuous nature of TSF timing correction which is reflected in its performance. The minimum CS offset is shown to be 10 μs but it can go up to 180 μs, which results in low overall precision.

B. Existing Wired Protocols Over WLAN

The two commonly known CS protocols which are designed for wired media, but have also been used for wireless synchronization, are NTP and IEEE 1588 PTP. Both these protocols are capable of distributing absolute or relative time in the network.

1) IEEE 1588 PTP Over WLAN: PTP presents a master–slave approach, where a single master clock continuously sends multicast timing information to synchronize clocks on different nodes. The master shares its time with the slaves using the two-way messaging similar to Fig. 1. If the PHY of the master clock supports drawing HW timestamps and putting it inside the packet "on the fly," then the M1 packet (called `Sync` in PTP) multicasts the time to all slaves in the PTP domain. However, if this "on the fly" timestamping is not possible, another packet, called `Follow_up`, is transmitted which contains the actual egress timestamp t_1. The slave sends its egress timestamp t_3 in M2 packet (called `Delay_Req` in PTP) to the master who, in turn, sends its ingress timestamp t_4 in another packet (called `Delay_Resp`) in PTP) to the slave. With the slave now possessing t_1, t_2, t_3, and t_4 timestamps, and assuming symmetric propagation delays, the offset between the master and the slave can be computed using (5). It should be noted that the `Delay_Req` packet is a multicast packet as per PTP, but should be sent as a unicast in WLANs to directly calculate the offset and delay between the AP and a wireless station. The master–slave hierarchy in PTP is established using the default best master clock algorithm (BMCA). However, for IEEE 802.11, the AP can be considered the master which synchronizes all STAs in the WLAN. Thus, as specified in [4], the default BMCA can be replaced with a custom BMCA which always chooses the AP as the master clock.

The PTP implementation over WLAN has been first carried out in [61] using SW and HW timestamping. The SW-based implementation was carried out for both Windows and Linux OS using built-in timers and interrupt-based timestamping. The HW-timestamping platform was FPGA-based, and was attached directly to IEEE 802.11b transceivers. The timestamps were drawn upon receiving transmission and reception signals from the transceivers. With HW timestamping, a mean CS error of 1.1 ns was achieved with a standard deviation of 3.1 ns. For SW-timestamping, the mean clock error for both Windows and Linux platforms was less than of 10 μs; the Windows platform, however, had a lower overall jitter (2.2 μs) as compared to Linux (6.3 μs). A similar analysis has been carried out in [62], which showed that even submicrosecond accuracy and precision are possible with SW timestamping over Linux. Nevertheless, these studies are limited in the sense that they only discuss overall CS performance and does not investigate other parameters affecting CS performance.

To overcome these limitations for SW-based PTP implementations over WLAN, Mahmood *et al.* in [43] have carried out an analysis which measured interrupt handling and timestamping delays and jitters for a Linux-based system, and have also provided a method to measure and calibrate fixed delays occurring inside the WLAN chipsets. Using optimized PI controllers, the overall CS error has been 59 ns with a jitter of 460 ns with only SW timestamping.

As part of improving the performance of HW-based solutions, Cooklev *et al.* in [63] analyzed the timestamping precision for the now defunct Intersil chipsets. Their proposed design takes 39.44 μs to draw a timestamp with a standard deviation of 145 ns for IEEE 802.11b devices. Another method which proposes to use HW timestamping is described in [64], where TSF timers are used for HW timestamping but both rate and offset correction are being performed at the STA. Exel have also only provided timestamping jitter, but have failed to provide end-to-end CS performance. This was achieved in [38] for IEEE 802.11b devices. Using a custom-designed FPGA board, HW timestamping delay of 89 ps and standard deviation of only 50 ps were achieved in this work. The overall CS accuracy was shown to be 240 ps with a standard deviation of 531 ps, which shows the potential of HW timestamping-based CS for wireless communication.

2) NTP Over WLAN: NTP primarily is a client–server protocol. Devices in NTP act as either primary or secondary servers or clients. The primary server derives its time directly from a UTC reference clock like GPS. On the contrary, the client NTP system can be synchronized to one or more upstream servers, but it is not able to synchronize other clients. Thus, for an implementation over WLAN, the AP can act as server, while the STAs become clients. To calculate clock offsets, NTP also relies on two-way packet exchange and its equation to calculate clock offset matches (5).

As NTP has been designed to be used over the Internet, there are many studies which highlight its performance and which show how the propagation delay on the Internet affects its precision. Inside a WLAN however, the performance is limited by where the SW timestamping is done. Thus, if it is done at the application, the packet will face a random channel access delay, which will increase the timestamping jitter. For a WLAN having 100 STAs, average propagation delay for a semiconductor factory using NTP over IEEE 802.11 [65] is shown to be 2.7 ms with a standard deviation of 2.39 ms. The final CS precision turns out to be 2.8 ms which varies with the number of STAs as more STAs will result in higher channel access delays.

Mahmood *et al.* in [66] have tried to improve the CS performance by doing SW timestamping at device driver level inside the ISR, which can avoid channel access delays. This has resulted in improving the precision by three orders of magnitude to 5.27 μs. Further performance improvements can be obtained by using NTP with HW timestamping as shown in [67], whose approach results in a CS precision of 100 ns over Ethernet. This indicates that HW timestamping solutions with NTP may also lead to better CS performance over the wireless channel.

C. Customized WLAN Protocols

A custom-designed protocol for WLAN providing only relative synchronization was already discussed in [58]. Another protocol providing absolute or relative CS, for real-time applications has been proposed in [69], which synchronizes all STAs with SW timestamping inside the device drivers. The novelty of this solution is that it is the first protocol of its kind to propose storing timestamps inside Beacons for application synchronization. In this way, there is no protocol overhead in establishing CS using this scheme. Mock *et al.* have also proposed to include several older timestamps inside the Beacon instead of just the latest one. In this way, if one Beacon is dropped, the next Beacon will contain the lost timestamp for the dropped Beacon. The clock adjustment itself is done with a rate correction-based approach. A final CS precision of 150 μs has been achieved using a proprietary driver on a Windows platform.

An algorithm called local selection (LS) to synchronize clocks in a WLAN was designed in [70]. The LS algorithm involves two way packet exchange for propagation delay calculations by the STAs. The delay value, along with a timestamp from the AP, is used to correct the STA clock only if the STA time is lagging behind the AP time. In this case, the STA time is corrected to represent the AP time. To make sure that the STA clock never goes faster than the AP clock, rate estimation is performed on the STA clock to slow it down. The idea of the algorithm has been to do simple delay measurements and one-time rate correction to achieve tight synchronization. However, the results on a Linux platform with SW timestamping in the device driver have shown that CS precision under 15 μs is achieved only when periodic rate corrections are performed at the STA. Thus, the performance of the LS algorithm coincides with that of a rate and offset correction procedure, for example, a PI controller.

The protocols and schemes mentioned so far follow the sender–receiver topology. However, a major source of jitter lies at the sender in its channel access mechanism, which discourages the use of SW timestamping at higher layers. The reference broadcast synchronization (RBS) is a relative CS scheme [72] which has been developed to synchronize the receivers not to the sender but with each other. The receivers share the reception time of a broadcast frame from the sender with each other, and then adjust their clocks based on the differences in the respective reception times.

To fit this scheme in the scope of infrastructure mode of IEEE 802.11, where one device remains the reference, reference broadcast infrastructure synchronization (RBIS) has been proposed in [71]. In RBIS, the timing master is one particular STA and not the AP. The reception of the Beacon frame from the AP is timestamped by all STAs including the master STA. The master STA then transmits its own timestamp to other STAs in a separate frame which is used by STAs for synchronization. The advantage of using another STA, and not the AP, as a reference is that commercially available APs can be used without any modifications in RBIS. In schemes like PTP, the AP's firmware needs to be modified to support timestamping in the device driver. In RBIS, a simple PC with Linux and open source driver can act as master STA, and the AP remains unmodified. This STA can be connected to a GPS to provide absolute time in the network. It should be noted that RBIS does not compensate for propagation delay, which will lead to systematic bias. However, this bias will depend upon the difference of propagation delay between AP and STAs, and the propagation delay between AP and the master STA. Thus, this bias will be smaller than the bias coming from a typical sender–receiver protocol without propagation delay compensation.

Finally, Cena *et al.* [73] propose a new CS scheme which exploits the advantages of different already mentioned protocols. In essence, this scheme proposes to store most, if not all, of timing related information inside beacons to avoid any synchronization overhead. If absolute CS is required, the AP can be modified to support either the TA scheme or PTP with connection to external reference such as GPS. If an unmodified AP is a system requirement, RBIS can provide relative or internal synchronization. Based on these and other minor modifications, synchronization using SW timestamping support over commercially available WLAN devices can be ensured. However, no actual implementation of such a CS method is carried out.

VI. Summary of CS Solutions Over IEEE 802.11

Table II summarizes various solutions for synchronization over IEEE 802.11 in light of the classification made in Fig. 5. It can be observed from this table that most of the non-IEEE 802.11 CS schemes can support absolute synchronization. This results directly from the fact that before IEEE 802.11-2012, there has been no support for application synchronization in the standard. Thus, new CS solutions have to be developed, which are capable of providing absolute or relative synchronization. Another observable trend is that the performance of CS is directly affected by the timestamping mechanism being used. The SW timestamping at the application is the worst in this regard as it is affected by the jitter from the entire protocol stack; interrupt-driven SW timestamping inside the device driver has resulted in CS accuracy of less than 50 μs. With HW timestamping, CS performance drops in the nanosecond range or below. To this end, the TM scheme of IEEE 802.11 is significant as it offers a standardized HW timestamping-based CS solution which can compensate for the propagation delays and can also provide absolute or relative application synchronization. The overall performance of this scheme is yet to be established due to lack of available HW and SW support to the scientific community. Nevertheless, like other HW-based solutions, it can be assumed to perform no worse than a few tens of nanoseconds. Thus, if Table II is compared with generic application requirements from Table I, HW timestamping-based solutions can easily provide synchronization for all application classes. SW timestamping-based solutions, in general, can fulfil the timing requirements for classes 1 and 2, and will require optimized settings to satisfy the requirement of class 1 applications.

Regarding protocol overhead, schemes (802.11-based or otherwise) which exploit beacon frames for carrying timestamps over the wireless channel do not require extra bandwidth. This is one disadvantage of using existing wired protocols over wireless which use dedicated packets for timing support. However, modified implementations of these protocols are discussed in the literature; they also propose carrying timing information in the beacons by the AP. This, in turn, leads to the issue of modifying the AP inside the device driver to draw reliable SW timestamps and fit them in the beacons as cited in [62] and [69]. In this context, Cena *et al.* [71] is significant as it proposes a SW timestamping-based solution which uses completely unmodified APs, and proposes changes only in STAs.

Finally, although HW timestamping support undoubtedly improves CS precision, optimized CS performance is obtained only when along with timestamping capabilities, propagation delays on the channel and within the devices are compensated, and clock adjustment is done efficiently. This would lead to CS performance in the nanosecond range with SW timestamping, and in the picosecond range with HW timestamping as shown in [43] and [38], respectively.

It should be noted that some studies for CS in WLAN based only on simulation are also available in literature. Publications such as [74] and [75] provide simulations of PTP over IEEE 802.11, but do not focus on correct modeling of timestamping errors, oscillator imperfections, and clock adjustment procedures. They only provide an environment for performing simulation, and hence such simulation-only studies are excluded from this study.

VII. Future Trends and Open Issues

While previous discussion has focused on various aspects of CS performance and existing synchronization solutions for IEEE 802.11, the research on this topic is ongoing. There is still a lot of room for improvement in terms of protocol design and CS performance as discussed below.

A. High performance CS and Standardization Activities

While for most industrial applications, CS accuracy and precision provided by protocols in Table II is sufficient, scenarios such as in wireless LXI [76] and time-based localization require CS accuracy below 1 ns. This will require advanced HW timestamping techniques, as existing work for high performance CS over WLAN focuses only on IEEE 802.11b

TABLE II
SUMMARY OF CS SOLUTIONS FOR IEEE 802.11

Description	Timing Reference	Timestamping Mechanism	Propagation Delay Calculation	Protocol Overhead	Accuracy	Precision
IEEE 802.11-based solutions						
TSF Scheme [9]	Relative	HW	None	None	4.0 μs	<1.0 μs
Timing Measurement (TM) method [9]	Absolute/Relative	HW	Two-way	None	NA	NA
Timing Advertisement (TA) method [9], [58]	Absolute/Relative	SW	None	None	0.50 μs	2.50 μs
non-IEEE 802.11 solutions						
TSF with virtual clocks [60]	Relative	SW	None	None	10.00 μs	NA
*Sync*TSF [58]	Relative	SW	None	None	1.80 μs	0.50 μs
PTP using Windows driver [68]	Absolute/Relative	SW	Two-way	Yes	5.50 μs	6.30 μs
PTP using Linux driver (1) [61]	Absolute/Relative	SW	Two-way	Yes	2.30 μs	4.40 μs
PTP using Linux driver (2) [62]	Absolute/Relative	SW	Two-way	Yes	6.60 μs	0.59 μs
PTP with optimized parameters in Linux [43]	Absolute/Relative	SW	Two-way with in-device delay compensation	Yes	59.00 ns	0.46 μs
PTP with HW timestamping (1) [68]	Absolute/Relative	HW	Two-way	Yes	1.10 ns	3.10 ns
PTP with HW timestamping (2) [38]	Absolute/Relative	HW	Two-way with PHY delay compensation	Yes	240.00 ps	531.00 ps
NTP with application timestamping [65]	Absolute/Relative	SW	Two-way	Yes	2.72 μs	2.39 ms
NTP with timestamping in the driver [66]	Absolute/Relative	SW	Two-way	Yes	0.50 ms	5.27 μs
Application CS with Beacons [69]	Absolute/Relative	SW	None	None	NA	150 μs
LS Algorithm for CS [70]	Absolute/Relative	SW	Two-way	Yes	35.00 μs	15 μs
Infrastructure mode-based RBS [71]	Absolute/Relative	SW	None	Yes	0.20 μs	0.18 μs

devices [38], [61]. On the other hand, COTS IEEE 802.11 devices contain OFDM-based PHYs, and HW timestamping for such devices is an ongoing task [77], [78]. The IEEE 802.11 standardization committee is also focusing on supporting high accuracy as part of improving its localization support. The IEEE 802.11az amendment [79] will propose a fine timing measurement (FTM) scheme which will enable absolute and relative localization with better accuracy. The FTM scheme is an extension to the existing TM scheme, and apart from improving certain protocol features, FTM requires the HW timestamps to have a resolution of 100 ps compared to 10 ns in the TM scheme [80]. This, in turn, reduces the standard deviation of the quantization error from 2.9 ns to 29 ps and hence, helps ensuring subnanosecond CS support which is a prerequisite for position determination within ± 1 m range.

Finally, the two-way packet exchange for determining propagation delay is valid only for static devices or devices with limited mobility, e.g., a moving hand of a robot. For mobile devices such as automated guided vehicles on the factory floor, propagation delays will change as a function of speed of the STAs in the WLAN. A simple two-way packet exchange to determine the offset in such scenarios will suffer from asymmetries which will lead to synchronization bias. The magnitude of this bias will be critical for applications which are seeking very high accuracy. Therefore, the effect of the speed of mobile STAs on synchronization accuracy, along with methods to properly compensate these varying propagation delays, needs further investigation.

B. Synchronization Support in Hybrid and Mesh Networks

For a successful deployment of a wireless network in industry, it is required to integrate it into a wired system [17], [26], which leads to a hybrid wired/wireless communication system. From the timing perspective, the seamless integration of a hybrid of wired and wireless systems must be ensured by using an interoperable solution for providing a global time base. However, it is shown in [81] that the concatenation of different communication systems with different asynchronous cycles has a negative impact on the overall temporal behavior of the system as the communication cycles need to wait for each other. This aspect is seldom considered in the literature and must be further investigated. In that respect, an example of hybrid synchronization protocol can be IEEE 802.1AS-2011, which offers synchronization in wired-wireless AVB networks. As already mentioned in Section IV, the TM synchronization method is used for exchanging timestamps between wireless stations to achieve synchronization. From the protocol perspective, the exchange of timestamps is carried out via the IEEE 802.11 MAC layer management entity (MLME) which is operating at the MAC layer. However, the request to start synchronization comes is originated in the media-dependent (MD) layer, which itself is operating at logical link layer. Using primitives defined in the IEEE 802.11, the MD layer communicates with the MLME to perform packet exchange for two-way synchronization. The MD layer also contains state machines for master and slave-based operations; upon deciding whether a wireless device is acting as the master or the slave in the network, the appropriate state machine is used for synchronization. One limitation of wireless synchronization in IEEE 802.1AS is that the synchronization packets between the master and each slave device are unicast, which can lead to higher bandwidth consumption by each slave in the wireless network to become synchronized. Hence, though not completely suited for use in industries, the protocol sets a valid precedent for achieving synchronization in hybrid systems.

The IEEE 802.1 time synchronized networking also specifies extensions to guarantee a very low latency and a high reliability for audio/video streaming, and potentially for future industrial communication systems [82]. The 802.1AS-2011 standard is intended to be used here as well for specifying the timing and synchronization for time-sensitive applications, but research in IEEE 802.1 hybrid systems is ongoing and not much published results are available.

Even though this study focuses only on infrastructure mode of IEEE 802.11, future wireless systems might require even more flexibility and interdevice communication. Wireless mesh networks are a promising solution for industrial communication systems of the future [83], because they are far more flexible in nature and can be considered as a wireless backbone network. They can support features like self-organization, which automatically establishes the wireless backbone without any manual intervention, and self-healing, which can select alternative routes whenever node or link failures occur [84]. Due to the required real-time capabilities of such networks, CS will also become a major topic of research in future mesh networks. One example of such an advanced mesh network is the IEEE 802.15.4e [85] standard amendment of IEEE 802.15.4. It targets industrial applications and its main feature is the time synchronized channel hopping to increase the reliability of transmissions and reduce the energy consumption of nodes. For such a mesh network, an achievable latency of $\geq$100 ms and a CS accuracy in the range of 1 ms are cited in [86]. Similar research for the IEEE 802.11 mesh networks can lead to long-range flexible networks, which opens the doors for further research on protocol design and high performance for CS in this field.

C. Fault Tolerance, Robustness, and Security

Whereas CS accuracy and precision are keenly sought while designing a protocol for WLAN, robustness and fault tolerance are often not given due attention. In general, fault tolerant protocol design for CS have been an active research topic; a survey of such protocols for wired networks has been provided in [87]. However, these protocols are not optimized against faults encountered in wireless networks. For example, packet losses is more common over the wireless channel than in wired networks. This leads to retransmissions and an increase in the synchronization overhead on the channel, but not many CS protocols contain features to mitigate packet losses. Mock *et al.* [69], [71] have offered robustness against loss of synchronization packets on the wireless channel by putting previous timestamps along with the current one inside synchronization packets. However, if the packet losses occur frequently and over a large time interval, the impact of oscillator wander and control loop dead-time is introduced at STAs which will result in loss of precision or even lead to control loop instability. Hence, CS robustness should be sought not only from the protocol perspective, but also from the perspective of clock adjustment. An example of robust design for wireless synchronization is [88], where a special rate-base KF is added in the control loop specifically to avoid loss of stability and precision. Another approach for robustness against packet loss is to send synchronization and all time-critical packets on two different channels [89], [90]. At present, mostly simulation-based studies are available for such an idea but research is being carried out to provide fault-tolerant CS support against packet losses in actual industrial scenarios [91]. The rise of WLAN mesh networks can also offer robustness by, for example, routing synchronization messages over different paths to other mesh STAs. Hence, fault tolerance and robustness is a crucial topic in wireless CS for both standalone networks and hybrid or mesh networks, and requires further investigations.

Finally, as CS is an essential and crucial network service for many applications, it can also be the potential target of attacks [92], [93]. IEEE 802.11 has a built-in mechanism of encryption and four-way key-exchange to ensure security over the wireless channel. However, security threats can still be present at higher protocol and cannot be mitigated only by encryption. Examples of such threats can be inserting messages with false timestamps, modifying timestamps, or delaying synchronization packets in order to maliciously disturb the CS. Especially in precision CS, straightforward application of standard IT security mechanisms such as IPsec is also not easy as it affects synchronization accuracy [94], [95]. Moreover, interfering and jamming devices in WLANs also affect all kinds of communication including synchronization. In this regard, the approaches of [90], [96] to send data on two different channels, can mitigate attacks against interfering and jamming devices which are attacking a particular channel in WLANs. Nevertheless, mitigation techniques against security threats for single-hop or multi-hop IEEE 802.11 networks in general, and CS in particular, are another open research topic.

REFERENCES

[1] J. Aweya, "Technique for differential timing transfer over packet networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 325–336, Feb. 2013.

[2] T. Sauter, "The three generations of field-level networks–evolution and compatibility issues," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3585–3595, Nov. 2010.

[3] D. L. Mills, *Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2010.

[4] IEEE, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., Sep. 2008.

[5] J. C. Eidson, *Measurement, Control, and Communication Using IEEE 1588*. New York, NY, USA: Springer, 2006.

[6] SMPTE, "Profile for use of IEEE-1588 precision time protocol in professional broadcast applications," *SMPTE ST 2059-2:2015*, pp. 1–19, Apr. 2015.

[7] IEEE, *802.1AS-2011 - IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, IEEE Std., 2011.

[8] H. Gerstung, "Synchronizing PTPv1 and PVPv2 clients with one common time source," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, Sep. 2008, pp. 1–6.

[9] IEEE, *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std., 2012.

[10] G. Gaderer, T. Sauter, F. Ring, and A. Nagy, "A novel, wireless sensor/actuator network for the factory floor," in *Proc. IEEE Sensors Conf.*, Nov. 2010, pp. 940–945.

[11] P. Parikh, T. Sidhu, and A. Shami, "A comprehensive investigation of wireless LAN for IEC 61850–Based smart distribution substation applications," *IEEE Trans. Ind. Inform.*, no. 3, pp. 1466–1476, 2013.

[12] P. Castello, P. Ferrari, A. Flammini, C. Muscas, P. Pegoraro, and S. Rinaldi, "A distributed PMU for electrical substations with wireless redundant process bus," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 5, pp. 1149–1157, May 2015.

[13] A. Nagy, R. Exel, P. Loschmidt, and G. Gaderer, "Time-based localisation in unsynchronized wireless LAN for industrial automation systems," in *Proc. Emerg. Technologies Factory Autom.*, Toulouse, France, Sep. 2011, pp. 1–8.

[14] IEEE, *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements -Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE Std., 2006.

[15] ISA, "Isa100 wireless compliance institute." 2015. [Online]. Available: http://www.isa100wci.org

[16] G. Cena, L. Seno, A. Valenzano, and C. Zunino, "On the performance of IEEE 802.11e wireless infrastructures for soft-real-time industrial applications," *IEEE Trans. Ind. Inform.*, vol. 6, no. 3, pp. 425–437, Aug. 2010.

[17] S. Vitturi, F. Tramarin, and L. Seno, "Industrial wireless networks: The significance of timeliness in communication systems," *IEEE Ind. Electron. Mag.*, vol. 7, no. 2, pp. 40–51, Jun. 2013.

[18] P. Djukic and P. Mohapatra, "Soft-TDMAC: A software-based 802.11 overlay TDMA MAC with microsecond synchronization," *IEEE Trans. Mobile Comput.*, vol. 11, no. 3, pp. 478–491, Mar. 2012.

[19] H. Trsek, *Isochronous Wireless Network for Real-time Communication in Industrial Automation*, (ser. Technologies for Intelligent Automation). 1st ed. Berlin, Germany: Springer, 2016.

[20] H. Trsek, T. Tack, O. Givehchi, J. Jasperneite, and E. Nett, "Towards an isochronous wireless communication system for industrial automation," in *Proc. 18th IEEE Int. Conf. Emerg. Technologies Factory Autom.*, Cagliari, Italy, Sep. 2013, pp. 1–4.

[21] J. R. Moyne and D. M. Tilbury, "The emergence of industrial control networks for manufacturing control, diagnostics, and safety data," *Proc. IEEE*, vol. 95, no. 1, pp. 29–47, Jan. 2007.

[22] M. Felser, "Real-time Ethernet - Industry prospective," *Proc. IEEE*, vol. 93, no. 6, pp. 1118–1129, Jun. 2005.

[23] J. Jasperneite and P. Neumann, "How to guarantee realtime behavior using ethernet," in *Proc. 11th IFAC Symp. Inform. Control Problems Manufacturing*, Salvador-Bahia, Brazil, Apr. 2004, pp. 1–6.

[24] G. Patti, G. Alderisi, and L. Lo Bello, "SchedWiFi: An innovative approach to support scheduled traffic in ad-hoc industrial IEEE 802.11 networks," in *Proc. IEEE 20th Conf. Emerg. Technologies Factory Autom.*, Luxemburg, 2015, pp. 1–9.

[25] S. Ivanov and E. Nett, "Localization-Based radio model calibration for fault-tolerant wireless mesh networks," *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 246–253, Feb. 2013.

[26] T. Sauter, J. Jasperneite, and L. L. Bello, "Towards new hybrid networks for industrial automation," in *Proc. 14th Int. Conf. Emerg. Technologies Factory Autom.*, Palma de Mallorca, Spain, 2009, pp. 1–8.

[27] J. Kjellsson, A. Vallestad, R. Steigmann, and D. Dzung, "Integration of a wireless I/O interface for PROFIBUS and PROFINET for factory automation," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4279–4287, Oct. 2009.

[28] J. Barnes *et al.*, "Characterization of frequency stability," *IEEE Trans. Instrum. Meas.*, vol. 20, no. 2, pp. 105–120, May 1971.

[29] A. Mahmood, "Time-based radio localization in IEEE 802.11b wireless local area networks," Ph.D. dissertation, Vienna Univ. Technol., Vienna, Austria, Jun. 2012.

[30] C.-F. Tsai, W.-J. Li, P.-Y. Chen, Y.-Z. Lin, and S.-J. Chang, "On-chip reference oscillators with process, supply voltage and temperature compensation," in *Proc. Int. Symp. Next-Generation Electron.*, Nov. 2010, pp. 108–111.

[31] E. McVey and C. T. Swanson, "Capacitor temperature compensated crystal oscillators," *IEEE Trans. Ind. Electron. Control Instrum.*, vol. IECI-25, no. 2, pp. 164–166, May 1978.

[32] D. Sullivan, D. Allan, D. Howe, and F. Walls, "Characterization of clocks and oscillators," National Institute of Standards and Technology, Technical Note 1337, 1990.

[33] D. Allan, "Time and frequency (time-domain) characterization, estimation, and prediction of precision clocks and oscillators," *IEEE Trans. Ultrasonics, Ferro Electr. Frequency Control*, vol. 34, no. 6, pp. 647–654, Nov. 1987.

[34] G. Gaderer, A. Nagy, P. Loschmidt, and T. Sauter, "Achieving a realistic notion of time in discrete event simulation," *Int. J. Distrib. Sensor Netw.*, vol. 2011, pp. 1–11, Jul. 2011.

[35] P. Loschmidt, R. Exel, and G. Gaderer, "Highly accurate timestamping for ethernet-based clock synchronization," *J. Comput. Netw. Commun.*, vol. 2012, pp. 1–11, 2012.

[36] K. Han and D.-K. Jeong, "Practical considerations in the design and implementation of time synchronization systems using IEEE 1588," *IEEE Commun. Mag.*, vol. 47, no. 11, pp. 164–170, Nov. 2009.

[37] R. Exel, "Receiver design for time-based ranging with IEEE 802.11b signals," *Int. J. Navigation Observation*, vol. 2012, Jul. 2012, Art. no. 743625.

[38] R. Exel, "Clock synchronization in IEEE 802.11 wireless LANs using physical layer timestamps," in *Proc. IEEE Symp. Presicion Clock Synchronizaion*, San Francisco, CA, USA, Sep. 2012.

[39] T. Broomhead, J. Ridoux, and D. Veitch, "Counter availability and characteristics for feed-forward based synchronization," in *Proc. Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, Oct. 2009, pp. 1–6.

[40] M. Dixon, J. Shrall, and R. Parthasarathy, "Controlling time stamp counter (tsc) offsets for multiple cores and threads," Jun. 23 2011, U.S .Patent App. 12/644,989. [Online]. Available: http://www.google.com/patents/US20110154090

[41] S. Salam and B. Philip, "Clock skew measurement for multiprocessor systems," Oct. 28 2010, U.S. Patent App. 12/430,992. [Online]. Available: http://www.google.com/patents/US20100275053

[42] P. Ferrari, A. Flammini, S. Rinaldi, A. Bondavalli, and F. Brancati, "Experimental characterization of uncertainty sources in a software-only synchronization system," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 5, pp. 1512–1521, May 2012.

[43] A. Mahmood, R. Exel, and T. Sauter, "Delay and jitter characterisation for software-based clock synchronization over WLAN using PTP," *IEEE Trans. Ind. Inform.*, vol. 10, no. 2, pp. 1198–1206, May 2014.

[44] G. Rai and S. Kumar, "A comparative study: RTOS and its application," *Int. J. Comput. Trends Technol.*, vol. 20, pp. 41–44, Feb. 2015.

[45] S. Lee, S. Lee, and C. Hong, "An accuracy enhanced IEEE 1588 synchronization protocol for dynamically changing and asymmetric wireless links," *IEEE Commun. Lett.*, vol. 16, no. 2, pp. 190–192, Feb. 2012.

[46] R. Exel, "Mitigation of asymmetric link delays in IEEE 1588 clock synchronization systems," *IEEE Commun. Lett.*, vol. 18, no. 3, pp. 507–510, Mar. 2014.

[47] Intersil, *HFA3861b Direct Sequence Spread Spectrum Baseband Processor Technical Report Data Sheet*, Intersil Corp, Feb. 2002.

[48] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst.*, 2004, pp. 39–49. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031501

[49] M. Mongelli and S. Scanzio, "A neural approach to synchronization in wireless networks with heterogeneous sources of noise," *Ad Hoc Netw.*, vol. 49, pp. 1–16, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870516301366

[50] Y. C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 124–138, Jan. 2011.

[51] M. Peca, V. Michalek, and M. Vacek, "Clock composition by wiener filtering illustrated on two atomic clocks," in *Proc. Eur. Frequency Time Forum Int. Frequency Control Symp.*, Jul. 2013, pp. 641–644.

[52] G. Giorgi and C. Narduzzi, "Modeling and simulation analysis of PTP clock servo," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, Oct. 2007, pp. 155–161.

[53] R. Exel and F. Ring, "Improved clock synchronization accuracy through optimized servo parametrization," in *Proc. Int. IEEE Symp. Precision Clock Synchronization Meas. Control Commun.*, Sep. 2013, pp. 65–70.

[54] G. Giorgi and C. Narduzzi, "Performance analysis of Kalman-Filter-Based clock synchronization in IEEE 1588 networks," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 8, pp. 2902–2909, Aug. 2011.

[55] M. Mongelli and S. Scanzio, "Approximating optimal estimation of time offset synchronization with temperature variations," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 12, pp. 2872–2881, Dec. 2014.

[56] A. Mahmood, R. Exel, and T. Sauter, "Impact of hard-and software timestamping on clock synchronization performance over IEEE 802.11," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, May 2014, pp. 1–8.

[57] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications*. London, U.K.: Artech House, 2005.

[58] A. Mahmood, R. Exel, and T. Sauter, "Performance of IEEE 802.11's timing advertisement against SyncTSF for wireless clock synchronization," *IEEE Trans. Ind. Informat.*, to be published.

[59] D. Dujovne, T. Turletti, and F. Filali, "A taxonomy of IEEE 802.11 wireless parameters and open source measurement tools," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 249–262, Apr.–Jun. 2010.

[60] F. Guo and T. Chiueh, "Comparison of QoS guarantee techniques for VoIP over IEEE802.11 wireless LAN," in *Proc. 15th Annu. Multimedia Comput. Netw. Conf.*, Jan. 2008, pp. 1–12.

[61] J. Kannisto *et al.*, "Software and hardware prototypes of the IEEE 1588 precision time protocol on wireless lan," in *Proc. 14th IEEE Workshop Local Metropolitan Area Netw.*, 2005, pp. 1–6.

[62] A. Mahmood, G. Gaderer, H. Trsek, S. Schwalowsky, and N. Kerö, "Towards high accuracy in IEEE 802.11 based clock synchronization using PTP," in *Proc. IEEE Symp. Presicion Clock Synchronizaion*, Munich, Germany, Sep. 2011, pp. 13–18.

[63] T. Cooklev, J. Eidson, and A. Pakdaman, "An implementation of IEEE 1588 over IEEE 802.11b for synchronization of wireless local area network nodes," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 5, pp. 1632–1639, Oct. 2007.

[64] J. Chen, Y. Li, Y. Song, and H. Chen, "Hardware-assisted clock synchronization in IEEE802.11 wireless real-time application," in *Proc. IET Conf. Wireless, Mobile Sensor Netw.*, Dec. 2007, pp. 363–366.

[65] D. Anand, D. Sharma, Y. Li-Baboud, and J. Moyne, "EDA performance and clock synchronization over a wireless network: Analysis, experimentation and application to semiconductor manufacturing," in *Proc. Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, Oct. 2009, pp. 1–6.

[66] A. Mahmood, G. Gaderer, and P. Loschmidt, "Clock synchronization in wireless LANs without hardware support," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, Nancy, France, May 2010, pp. 75–78.

[67] S. Butner and S. Vahey, "Nanosecond-scale event synchronization over local-area networks," in *Proc. 27th Annu. IEEE Conf. Local Comput. Netw.*, Nov. 2002, pp. 261–269.

[68] J. Kannisto, T. Vanhatupa, M. Hnnikinen, and T. Hmlinen, "Precision time protocol prototype on wireless Lan," in *Proc. Int. Conf. Telecommun.*, 2004, pp. 1236–1245.

[69] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Continuous clock synchronization in wireless real-time applications," in *Proc. 19th IEEE Symp. Reliable Distrib. Syst.*, 2000, pp. 125–132.

[70] D. Sigg and E. Schreiber, "Clock synchronization for wireless lan," Diploma Thesis, Swiss Federal Inst. Technol. Zurich, Switzerland, 2002.

[71] G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "Implementation and evaluation of the reference broadcast infrastructure synchronization protocol," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 801–811, Jun. 2015.

[72] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844143

[73] G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "A unified clock synchronization protocol for infrastructure wireless LANs," in *Proc. IEEE 1st Int. Forum Research Technologies Society Ind. Leveraging Better Tomorrow*, Sep. 2015, pp. 508–515.

[74] Y. Bang *et al.*, "Wireless network synchronization for multichannel multimedia services," in *Proc. 11th Int. Conf. Adv. Commun. Technol.*, 2009, pp. 1073–1077. [Online]. Available: http://portal.acm.org/citation.cfm?id=1701835.1701868

[75] L. Li, B. Li, and H. Wang, "Clock synchronization of wireless distributed system based on IEEE 1588," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Oct. 2010, pp. 205–209.

[76] Q. Honglei, Q. Changquan, and X. Lingfei, "Research on LXI bus based on wireless technology," in *Proc. Int. Symp. Inform. Technol. Convergence*, Nov. 2007, pp. 106–109.

[77] H.-M. Troger, J. Robert, M. Keppeler, M. Hartmann, and A. Heuberger, "Partial OFDM demodulation for frequency syntonization of wireless sensor nodes," in *Proc. 2015 IEEE Int. Symp. Precision Clock Synchronization Meas., Control, Commun.*, Oct. 2015, pp. 1–6.

[78] I.-C. Chao, K. Lee, R. Candell, F. Proctor, C.-C. Shen, and S.-Y. Lin, "Software-defined radio based measurement platform for wireless networks," in *Proc. 2015 IEEE Int. Symp. Precision Clock Synchronization Measurement, Control, Commun.*, Oct. 2015, pp. 7–12.

[79] *IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Enhancements for Positioning*, IEEE Std., 2015.

[80] K. Stanton, "Addition of p802.11-MC fine timing measurement (FTM)to p802.1AS-Rev," Presentation, 2015. [Online]. Available: http://www.ieee802.org/1/files/public/docs2015

[81] S. Höme, S. Palis, and C. Diedrich, "Design of communication systems for networked control system running on PROFINET," in *Proc. 10th IEEE Workshop Factory Commun. Syst.*, 2014, pp. 1–8.

[82] J. Imtiaz, J. Jasperneite, and K. Weber, "Approaches to reduce the latency for high priority traffic in IEEE 802.1 AVB networks," in *Proc. 9th IEEE Int. Workshop Factory Commun. Syst.*, Lemgo, Germany, May 2012, pp. 161–164.

[83] A. Frotzscher *et al.*, "Requirements and current solutions of wireless communication in industrial automation," in *Proc. IEEE Int. Conf. Commun. Workshops*, Jun. 2014, pp. 67–72.

[84] G. Lukas, A. Herms, S. Ivanov, and E. Nett, "Dependable wireless mesh networks: An integrated approach," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 24, no. 2, pp. 151–169, Apr. 2009.

[85] *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer*, IEEE Std. 802.15.4e-2012, 2012.

[86] D. Stanislowski, X. Vilajosana, Q. Wang, T. Watteyne, and K. Pister, "Adaptive synchronization in IEEE802.15.4e networks," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 795–802, Feb. 2014.

[87] P. Ramanathan, K. Shin, and R. Butler, "Fault-tolerant clock synchronization in distributed systems," *Computer*, vol. 23, no. 10, pp. 33–42, Oct. 1990.

[88] H. Abubakari and S. Sastry, "IEEE 1588 style synchronization over wireless link," in *Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun.*, Sep. 2009, pp. 127–130.

[89] M. Hendawy, "Application of parallel redundancy in a Wi-Fi-based WNCS using OPNET," in *Proc. IEEE 27th Canadian Conf. Electr. Comput. Eng.*, May 2014, pp. 1–6.

[90] M. Rentschler *et al.*, "Simulative comparison of parallel redundant wireless systems with OMNet++," in *Proc. IEEE 23rd Int. Symp. Ind. Electron.*, Jun. 2014, pp. 1135–1140.

[91] G. Cena, S. Scanzio, L. Seno, A. Valenzano, and C. Zunino, "Combining reliability and timeliness in industrial wireless networks: An experimental assessment," in *Proc. IEEE World Conf. Factory Commun. Syst.*, May 2016, pp. 1–4.

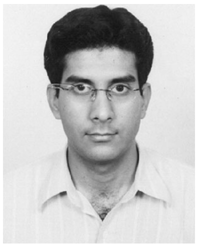
[92] M. Bishop, "A security analysis of the NTP protocol version 2," in *Proc. 6th Annu. Comput. Security Appl. Confe.*, Dec. 1990, pp. 20–29.

[93] G. Gaderer, A. Treytl, and T. Sauter, "Security aspects for IEEE 1588 based clock synchronization protocols," in *Proc. 2006 IEEE Int. Workshop Factory Commun. Syst.*, 2006, pp. 247–250.

[94] A. Treytl, B. Hirschler, and T. Sauter, "Secure tunneling of high-precision clock synchronization protocols and other time-stamped data," in *Proc. 2010 8th IEEE Int. Workshop Factory Commun. Syst.*, May 2010, pp. 303–312.

[95] N. Moreira, J. Lzaro, J. Jimenez, M. Idirin, and A. Astarloa, "Security mechanisms to protect IEEE 1588 synchronization: State of the art and trends," in *Proc. 2015 IEEE Int. Symp. Precision Clock Synchronization Meas., Control, Commun.*, Oct 2015, pp. 115–120.

[96] G. Cena, S. Scanzio, and A. Valenzano, "Seamless link-level redundancy to improve reliability of industrial Wi-Fi networks," *IEEE Trans. Ind. Inform.*, vol. 12, no. 2, pp. 608–620, Apr. 2016.



Aneeq Mahmood (M'13) was born in 1982 in Lahore, Pakistan. He received the Master's degree in electrical engineering from KTH Royal Institute of Technology, Stockholm, Sweden, in 2007, and the Ph.D. degree in electrical engineering from Vienna University of Technology, Austria, in 2013.

From 2007 to 2013, he had been with the Austrian Academy of Sciences where he ~~had~~ worked on projects related to wireless localization and factory automation. He is currently with Centre for Integrated Sensor Systems, Danube University Krems. His ~~current~~ research interests include wireless sensor networks, wired and wireless clock synchronization, and energy optimization in automation.

Reinhard Exel was born in Horn, Austria, in 1980. He received the Master's and Ph.D. degrees in electrical engineering from the Vienna University of Technology, Vienna, Austria in 2007 and 2012 respectively.

He has contributed to several research projects at the Centre for Integrated Sensor Systems (at Austrian Academy of Sciences and Danube University krems) dealing with various aspects of industrial communication, particularly network-based high accuracy clock synchronization and timestamping in wired and wireless networks. His research interests include accuracy improvement techniques of synchronization systems (asymmetry for Ethernet, multipath propagation for wireless) as well as various aspects of time-based real-time locating systems.

Henning Trsek (M'07) received the degree in electrical engineering and information technology from the OWL University of Applied Sciences, Lemgo, Halmstad University, Sweden, and Aalborg University, Denmark, and received the Master's degree in that field in 2005. He received the Ph.D. degree in the area of wireless real-time communication from the Otto-von-Guericke-University Magdeburg, ~~Magdeburg,~~ Germany, in 2016.

Afterward, he was employed at the Institute Industrial IT (inIT), Germany, with the responsibilities of a research group leader. His current research interests include the area of Smart Factories, Industrie 4.0 and Industrial Security. He is currently with rt-solutions.de as a Senior Consultant and responsible for the Industrial Security Department. He is an active member of the IEEE Industrial Electronics Society Technical Committee on Factory Automation Subcommittee on Security. Furthermore, and he is actively contributing to ~~the~~ standardization as member of the VDI working group 5.22 Security in Automation in Germany.

Thilo Sauter (M'93–SM'09–F'14) received the Dipl.-Ing. and Doctorate degrees in electrical engineering from the Vienna University of Technology (VUT), Vienna, Austria, in 1992 and 1999, respectively.

From 1992 to 1996, he was a Research Assistant with the Institute of General Electrical Engineering, VUT, and was involved in research on programmable logic and analog ASIC design. Subsequently, he joined the Institute of Computer Technology, VUT, and led the Factory Communications Group. From 2004 to 2013, he has also been the Director of the Institute for Integrated Sensor Systems, Austrian Academy of Sciences. Since 2013, he has been with the Center for Integrated Sensor Systems, Danube University Krems, and an Associate Professor for Automation Technology at TU Wien since 2014. His current research interests include smart sensors and automation networks with a focus on real-time, security, interconnection, and integration issues. He is author of more than 200 scientific publications and has been involved in the organization of several IEEE conferences. Moreover, he has been involved in the standardization of industrial communication systems for more than 15 years.