Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2011, Article ID 797931, 14 pages
doi:10.1155/2011/797931

Research Article

Secure Precise Clock Synchronization for Interconnected Body Area Networks

David Sanchez Sanchez,[1] Luis Alonso,[2] Pantelis Angelidis,[3] and Christos Verikoukis[4]

[1] *Department of Information and Communication Technologies, Pompeu Fabra University, 08018 Barcelona, Spain*
[2] *Department of Signal Theory and Communications, Polytechnic University of Catalonia, 08034 Barcelona, Spain*
[3] *Department of Engineering Informatics and Telecommunications, University of Western Macedonia, 50100 Kozani, Greece*
[4] *Intelligent Energy Area, Telecommunications Technological Centre of Catalonia, 08860 Barcelona, Spain*

Correspondence should be addressed to David Sanchez Sanchez, david.sanchezs@upf.edu

Received 30 October 2010; Accepted 26 January 2011

Academic Editor: Dries Neirynck

Secure time synchronization is a paramount service for wireless sensor networks (WSNs) constituted by multiple interconnected body area networks (BANs). We propose a novel approach to securely and efficiently synchronize nodes at BAN level and/or WSN level. Each BAN develops its own notion of time. To this effect, the nodes of a BAN synchronize with their BAN controller node. Moreover, controller nodes of different BANs cooperate to agree on a WSN global and/or to transfer UTC time. To reduce the number of exchanged synchronization messages, we use an environmental-aware time prediction algorithm. The performance analysis in this paper shows that our approach exhibits very advanced security, accuracy, precision, and low-energy trade-off. For comparable precision, our proposal outstands related clock synchronization protocols in energy efficiency and risk of attacks. These results are based on computations.

1. Introduction

Body area networks (BANs) are receiving a lot of attention for civilian applications [1]. A BAN consists of wireless connected sensors nodes worn by or implated to a human body. Each BAN includes a controller node. The role of this node can be assigned either to a single sensor node or dynamically to any of the nodes of the BAN.

In this paper, we consider the interconnection of multiple BANs by means of the controller nodes. This setup enables quick, modular, and inexpensive deployment of a long range distributed wireless sensor network (WSN) for key applications, such as patient monitoring, for instance, for quick deployment of a medical WSN in a field hospital after disaster events. Each BAN collects vital parameters of a single patient. The cooperation between the different controllers allows for monitoring of multiple patients from a single central or remote location.

In the rest of the paper, we use WSN to refer to the long range wireless network formed by the interconnection of multiple BANs through the controller nodes.

The WSN can be formed in public or hostile areas, where wireless communications can be easily eavesdropped, deleted, and/or modified. In some applications, sensor nodes are left unattended (when detached from the monitored body), being then prone to capture and manipulation by an attacker. The monitored human itself may also be an intruder and, thus, may manipulate its body-attached nodes.

Time synchronization is a key service in WSNs for a diversity of purposes; including data fusion, power management, positioning, message integrity, coordination of future actions, and timestamping of sensed events. However, sensor node clocks have arbitrary starting offsets and nondeterministic fluctuating skews.

Moreover, the special nature of WSNs imposes challenging and intertwined requirements on secure time synchronization design. Firstly, time synchronization must be highly energy-efficient, since sensor nodes operate with batteries. Secondly, time synchronization must be accurate to the microsecond level as to fulfill time-critical BAN applications. Thirdly, time synchronization must be

secure against passive, active, internal, and external attackers.

Existing secure pairwise time synchronization approaches are based either on *receiver-receiver synchronization* [2, 3] or on *sender-receiver synchronization* [3–6]. Based on pairwise time synchronization, secure global time synchronization is achieved by transferring global time from a source node to all the nodes of the network.

Security and accuracy cannot straightforward be provided in WSNs to the cost of sending a larger number of or more frequent synchronization messages for two reasons. Firstly, these solutions impose a high energy cost. Secondly, they do not guarantee that the synchronization of nodes will remain precise between two successive resynchronizations.

We propose a secure, accurate, precise, and energy-efficient time synchronization system for a WSN. We combine secure pairwise synchronization protocol (SPS) [4], rate adaptive time synchronization (RATS) [7], and μTESLA [8]. SPS is used to achieve highly accurate and pairwise secure synchronization. RATS is used to maintain the accuracy achieved by SPS throughout a long period of time. μTESLA is employed to enable efficient digital signatures for BAN-wide broadcast message synchronization.

The system can be used in WSN with extremely low-duty cycle nodes. The system achieves resiliency against compromised nodes without requiring repeating synchronization messages or continuous media sensing. The energy cost of the system is also very low.

The contributions of this paper are fivefold. Firstly, we derive the requirements for a secure time synchronization service for WSNs. Secondly, we exhaustively evaluate existing secure time synchronization proposals for WSN. Thirdly, we propose the SPS with sample exchange (SPS-SE) protocol, a SPS-based protocol for synchronizing two nodes and exchanging time observations for RATS. Fourthly, we propose a novel system for secure time synchronization in a WSN. Finally, we exhaustively evaluate the time synchronization proposal. These results are based on computations.

Temperature is a key parameter influencing clock skews. Therefore, we analyse our proposal for indoor and outdoor scenarios. A representative indoor scenario is a conventional hospital floor with a WSN. A representative outdoor scenario is a field hospital with a WSN.

The remainder of this paper is organized as follows. Section 2 derives the requirements for a secure time synchronization service and evaluates existing secure time synchronization proposals. In Section 3, we present the model of WSN for our system and we give important definitions and background. We describe our time synchronization system in Section 4. Sections 5 and 6, respectively, evaluate the security and performance level of the system. Finally, Section 7 concludes and discusses our future work.

2. Evaluation of Secure Time Synchronization Approaches

We first derive the requirements for a secure time synchronization service for WSNs. Secondly, we classify and evaluate existing secure time synchronization schemes against these requirements.

2.1. Requirements. A secure time synchronization service for WSNs must comply and trade off the following requirements: *low cost*, *accurate*, *precise*, *secure,* and *periodically-scheduled*.

Firstly, among all sensor node components, the radio consumes the most significant amount of energy [9, 10]. Therefore, the synchronization service must minimize the number of messages exchanged by sensor nodes. Secondly, the time synchronization service must enable applications with time accuracy demands at the tens of μs level. Thirdly, time synchronization among nodes must be precise up to the hundreds of μs for long periods. This requirement is particularly challenging to comply with for low-cost sensor nodes. Fourthly, WSNs are especially vulnerable to security attacks. Since sensor nodes use wireless communications, an external attacker may easily delete, forge, and modify time synchronization messages. Additionally, the attacker may launch pulse-delay [4] and/or wormhole [11] attacks, in which the adversary delays and/or rushes the authenticated synchronization messages, respectively. Since sensor nodes are not tamper-proof, an attacker may also compromise a (or a few) sensor node(s). Then, the attacker can use the sensor node(s) to inject false time synchronization messages. In addition, the attacker may instruct the sensor node(s) not to cooperate in the synchronization protocol. Finally, substantial clock drift during sleep periods requires fine scheduling of the time synchronization protocol.

2.2. Existing Techniques. Ganeriwal et al. [4] proposed several techniques for secure pairwise synchronization (SPS), multihop synchronization, and groupwise synchronization. The SPS adds timestamps and message integrity codes (MICs) to protect the synchronization messages. To remove the time uncertainty introduced by the MAC access waiting time, they propose to timestamp the message below the MAC layer. Their practical measurements show that SPS can synchronize two Mica2 motes with an accuracy of 10 μs. An attacker can delay a time synchronization message only up to 20 μs without being noticed. However, SPS exhibits no resiliency to compromised nodes.

Secure multi-hop synchronization [4] can be used to synchronize sensor nodes not within direct wireless communication range. Ganeriwal et al. propose three similar techniques: secure opportunistic multi-hop (SOM), secure direct multi-hop (SDM), and secure transitive multi-hop (STM). The three techniques extend SPS by using one or a set of intermediate trusted nodes. For five hops, SDM and STM provide a 25 μs time synchronization accuracy and exhibit a pulse-delay attack vulnerability window of 50 μs to 120 μs, respectively. However, they exhibit no resiliency to compromised nodes. SOM can cope with compromised nodes but exhibits very poor accuracy and pulse-delay protection.

Group multi-hop synchronization [4] can be used to synchronize a group of sensor nodes of a wireless neighborhood. They first propose a lightweight secure group synchronization (L-SGS) that exploits multicast authentication to synchronize the neighborhood. This technique is also vulnerable to compromised nodes. To solve this vulnerability, Ganeriwal et al. propose secure group synchronization (SGS). SGS requires nodes to exchange and process messages after the initial multicast exchange to check time consistency. SGS and L-SGS provide 10 μs accuracy and 20 μs pulse-delay vulnerability window. The consistency check is inefficient, since it does not exploit the broadcast nature of the wireless neighborhood. Moreover, the consistency check can only tolerate one compromised node, and no provision is made to cope with a subset of compromised nodes. Moreover, they allow for whatever neighborhood member to anarchically start the (L-)SGS protocol, which can be exploited for battery depletion attacks.

Manzo et al. [2] discuss several internal attacks against and countermeasures for Reference Broadcast Synchronization (RBS) [12], Timing-sync Protocol for Sensor Networks (TPSN) [13], and Flooding Time Synchronization Protocol (FTSP) [14]. The internal attacks can be summarized as different flavors of injecting false information to disrupt the time synchronization protocol operation, for example, introducing false timestamps. To secure time synchronization protocols, they suggest to elect time root nodes probabilistically, to send time synchronization messages through alternate paths, and to use μTESLA [8] to authenticate broadcast synchronization messages. Unfortunately, the strength and performance of these countermeasures is not analyzed. Moreover, they provide no mechanism to authenticate the timeliness of synchronization messages and thus no protection against pulse-delay and wormhole attacks.

Song et al. [3] investigated countermeasures for delay attacks against synchronization messages launched from compromised nodes. They proposed two methods for detecting and tolerating delay attacks: a generalized extreme studentized deviate (GESD) based and a threshold based. The general idea is to identify the malicious time offsets that are under delay attacks after collecting a set of time offsets from multiple involved nodes. The underlying assumption is that a malicious node magnifies its clock offset to accomplish the delay attack. Then, the GESD-based method filters out an outlier by applying the assumption that clock offsets from benign nodes follow the same (or similar) distribution or pattern. Similarly, the threshold-based method filters out outliers by rejecting clock offsets above an upper bound. They also show that these methods can be used to improve the accuracy of RBS in the presence of clock offset outliers. However, Song et al. do not provide arguments and/or proofs validating that benign clocks follow the same (or similar) distribution in practice. Moreover, both methods require each node to receive a sufficiently large number of messages to detect outliers, thus the accuracy improvement comes at a substantial energy cost.

Sun et al. [6] proposed to leverage SPS and μTESLA to provide global time synchronization in multi-hop static WSNs. SPS is periodically and asynchronously employed to pairwise synchronize all the nodes of the WSN. Subsequently, global time is transferred from (set of) source nodes to the rest of sensor nodes. To improve the communication efficiency, authenticated global time synchronization messages are broadcasted locally in a wireless neighborhood (cf. L-SGS and SGS). To be resilient against compromised nodes, nodes already synchronized to global time rebroadcast synchronization messages. To tolerate up to t compromised neighbor nodes, the receiver must select among $2t + 1$ clock differences through different neighbor nodes.

Sun et al. demonstrated experimentally that for a WSN with only 60 nodes and to tolerate up to 4 compromised nodes per neighborhood, their approach reaches an average global time accuracy below 52.08 μs and a minimum accuracy below 121.52 μs *right after* running the protocol. These numbers correspond to global time synchronization intervals of 5 to 10 seconds. Unfortunately, they do not discuss how this accuracy evolves through the 5- or 10-second interval. Since the clock drift of the CC2420 is 40 ppm [9], the time accuracy of two nodes can diverge up to 80 μs per second after the synchronization. Then, in practice in 5 and 10 seconds the synchronization can loose precision up to 521.52 μs and 921.52 μs, respectively.

To avoid such poor accuracy, we can set up a lower global synchronization interval and, accordingly, also a lower pairwise synchronization interval. However, simple analysis shows that the effect is a substantial increase in energy consumption needed to send multiple re-synchronization messages.

The connectivity of the nodes with the rest of the network is not motivated or argued by Sun et al.'s proposal. In cases with cliquish neighborhoods the resiliency improving approach can turn out to be useless, since neighborhood nodes cannot get global time through different wireless paths to the rest of the WSN.

We identify a number of open issues in the previous proposals. Firstly, none of the above presented approaches analyze the period of time required to synchronize nodes with any of their proposed techniques, failing then to prove effective and efficient for low-duty cycle sensor nodes. For instance, if a WSN application requires nodes to sleep 99% of time, *is 1% of time sufficient to synchronize time of sensor nodes? Which fraction of time out of that 1% is to be dedicated for time synchronization? Which is the maximum accuracy the sensor clocks will achieve?* Yet more important, *which accuracy will the sensor clocks maintain during each synchronization interval?* and *how much power needs a sensor to invest to achieve such accuracy level?*

Secondly, none of these proposals discuss the scheduling of the time synchronization protocol. Unless we assume unsustainable 100% duty cycles, time synchronization cannot be completely asynchronous, but nodes need to prearrange well-delimited intervals of time to synchronize.

Finally, protection for time synchronization protocols against wormhole attacks is not analyzed. Sun et al. [6] propose to detect wormholes by detecting that the transmission delay is less than the maximum expected delay. However, this solution is at odds with the nature of a wormhole, since a wormhole attack decreases the latency of messages

exchanged by two nodes at different locations in the WSN [15].

Hoepman et al. [16] consider an adversary that aims at tampering with the clock synchronization by intercepting messages, replaying intercepted messages, and capturing nodes (i.e., revealing their secret keys and impersonating them).

They present a clock sampling algorithm which tolerates attacks by this adversary, collisions, a bounded amount of losses due to ambient noise, and a bounded number of captured nodes that can jam, intercept, and send fake messages. The algorithm is self-stabilizing, so if these bounds are temporarily violated, the system can stabilize back to a correct state.

The core of their clock synchronization algorithm is a mechanism for sampling the clocks of neighboring nodes at reception of broadcasts called beacons. A beacon acts as a shared reference point.

3. Wireless Sensor Network Model

A BAN consists of wireless connected sensors nodes worn by or implated to a human body. A sensor node is a low-cost, low-power, wireless-enabled computing device. New sensor nodes can be incrementally added after the initial deployment. The BAN ranges a few meters around a human body.

Each BAN includes a controller node (CN) with routing, data fusion, and other functions. The role of this node can be assigned either to a single sensor node or dynamically to any of the nodes of the BAN. Let us assume that in average a BAN includes n sensor nodes. Two neighbour CNs are connected by a direct wireless link.

A WSN is the interconnection of multiple BANs by means of the controller nodes. The number of WSN nodes can range up to thousands of nodes. Therefore, the WSN can occupy a huge area.

The WSN can be formed in public or hostile areas, where wireless communications can be easily eavesdropped, deleted, and/or modified. In some applications, sensor nodes are left unattended (when detached from the monitored body), being then prone to capture and manipulation by an attacker. The monitored human itself may also be an intruder and, thus, may manipulate its body-attached nodes.

We also assume that nodes share pairwise keys [17]. Alternatively, each pair of nodes can *directly* derive a pairwise key [18, 19] just by knowing each node ID, without the need to exchange further messages.

Integrity-protected messages are timestamped below the MAC layer using existing techniques. Therefore, the period of uncertainty needed for the host to access the network interface card and to backoff is removed as demonstrated in [6].

Data sensed by sensor nodes is to be sent to a (small number of) base station(s) in a central or remote location.

3.1. Power Management. The WSN is provided of a power management service to save energy of sensor nodes. This service, in turn, guarantees the longest longevity for the WSN. The basic idea of the power management service is to put the radio of sensor nodes to sleep during idle times and wake it up right before message transmission and/or reception.

To allow communication in WSNs formed of low-duty cycle nodes, sensor nodes need to synchronize active and wake periods of time. This synchronization can be achieved synchronizing each sensor node to a common reference time. However, sensor nodes embed low-cost crystal oscillators which drift from the reference time. Consequently, sleep T_{sleep} and wake T_{wake} periods are not equally measured by all the sensor nodes.

A time period of guard T_{guard} is defined to enable active periods from two sensor nodes to overlap despite their respective clock drift errors. The time of guard T_{guard} is a local time measure. During its time of guard, a sensor node can receive but cannot send data.

3.2. Definitions. In the rest of the paper we use the following definitions.

(i) *Pairwise Time*. Pairwise time is the agreed synchronized time between two arbitrary sensor nodes u and v.

(ii) *BAN Time*. BAN time is the agreed synchronized time among the sensor nodes of a BAN.

(iii) *WSN Time*. WSN time is the agreed synchronized time among all the sensor nodes of the WSN.

(iv) *Coordinated Universal Time (UTC)*. This is the global synchronized time used by humans.

(v) *Clock Accuracy*. Clock accuracy is the degree of closeness of a measured time value to that of a reference clock. For instance, let us consider a reference clock at 12:00:00. A clock c_1 measuring 12:05:00 after 2 days of having been perfectly synchronized is considered to be inaccurate.

(vi) *Precision*. Precision is the degree of closeness to which repeated measured time values agree with each other under unchanged conditions. Let us consider again the clock c_1. During 10 consecutive days, we obtain time readings differing 10 seconds from each other and differing an average of 5 minutes from the reference clock. We consider c_1 to be a precise yet inaccurate clock.

3.3. Prediction of Clock Skew. The time difference measured by two different clocks t_u and t_v depends on differences in *phase* and *frequency* of oscillation of each clock. The phase and the frequency oscillation variation of a clock is often referred in the literature to as *clock offset* and *clock skew*, respectively.

Initially, the offset counts the elapsed time from the time of start of t_u in respect to t_v or vice versa. Note that instantaneously correcting the offset between two clocks is relatively simple by running for instance a pairwise synchronization protocol. However, because of the effect

of the clock skew, the two clocks drift after the initial synchronization. Therefore, to keep the clock drift under a required upper bound, all the related schemes in Section 2.2 propose to resynchronize frequently. Instead, to reduce the frequency of re-synchronization and, thus, the energy consumption of sensor nodes, we propose to predict the clock skew of each sensor node clock.

The variation of clock skew depends on different nondeterministic factors: including aging, noise, warmup, variations in temperature, atmospheric pressure, acceleration, voltage, radiation, and magnetic fields [20, 21].

We observe that temperature is the factor most influencing the frequency of the clocks. Temperature can cause variations up to several tens of ppm while the aggregated variation caused by other factors is far below 1 ppm [21].

We also observe that in typical WSN environments temperature changes smoothly. For instance, outdoors the temperature changes smoothly because of weather conditions. The temperature keeps relatively constant in normal circumstances in most indoor scenarios. In BANs, the effect of temperature change rate is even more negligible, since sensor nodes are separated just a few centimeters from each other.

Hereafter, a time period of no substantial temperature change is referred to as *epoch*. We assume that the clock skew for each sensor node and, thus, the relative clock skew for two nodes remain constant during an epoch [21].

Ganeriwal et al. [7] designed a prediction-based algorithm to model long-term clock skew between two sensor node clocks t_u and t_v. Wu et al. [21] and Elson et al. [12] developed similar concepts, demonstrating thus its suitability for WSNs.

After Ganeriwal et al. [7], the following P-degree polynomial represents the relative clock model between two nodes u and v:

$$\hat{t}_v(t_u) = \sum_{p=0}^{P} \left(\beta_p \cdot t_u^p\right) + \epsilon, \tag{1}$$

where $\hat{t}_v(t_u)$ is the prediction of of the actual t_v measured with the clock of node u. The error ϵ includes both measurement errors and environmental factors that influence the clock stability. Over short timescales, there is the general agreement that a linear relative clock model ($P = 1$) is sufficient [7, 12–14].

Given a window of W past observations $(t_{u,i}, t_{v,i})$, $i = 1, 2, \ldots, W$, the parameters β_0 and β_1 are the values which minimize the residual sum of squares (RSS):

$$\text{RSS} = \min_{\beta_p \forall p=0,1} \sum_{i=1}^{W} \left(t_{v,i} - \left[\sum_{p=0}^{1} \left(\beta_p \cdot t_{u,i}^p\right) \right] \right)^2. \tag{2}$$

It is easy to see that finding the values β_0 and β_1 which minimize RSS is an extremely low-complexity problem. Then, the energy consumed by calculating these parameters can be neglected in comparison to the cost of sending a bit.

We define a sampling period S as the interval of time separating two consecutive observations $(t_{u,i}, t_{v,i})$ and $(t_{u,i+1}, t_{v,i+1})$. Naturally, shorter values of S minimize the error of the prediction.

Given S, there exists an optimal window size W which minimizes the error of the prediction. For instance, Ganeriwal et al. [7] experimentally showed that $W = 8$ and $S = 60$ seconds minimize the prediction error E_p for indoor environment at a temperature range from 25 to 26°C.

In practice, each time we obtain new time observations, by using the low-cost rate adaptive time synchronization algorithm (RATS) [7], we can calculate two optimal values S and W which maintain the error of the prediction within a desired error bound for the calculated S.

3.4. Estimation of Prediction Error. Given a time window of W observations, $(t_{u,i}, t_{v,i})$, the time at node v, $\hat{t}_v$, can be predicted using time at node u, t_u, with (1). Following standard regression theory, we can construct a $(1 - \alpha)$ confidence interval for this prediction as

$$\hat{t}_v \pm \hat{E}_p, \tag{3}$$

where

$$\hat{E}_p = t_{u_{(1-\alpha)/2, W-2}} \cdot \text{SE}\left(\hat{t}_u\right). \tag{4}$$

The first term of the product in (4) refers to an upper quantile of the t_v distribution with $W-2$ degrees of freedom. The second term is the standard error (SE) of the predicted value.

3.5. The RATS Algorithm. The objective of RATS is to repeatedly calculate a new sampling period S so that the synchronization error remains bounded within the user specifications. The pseudocode for the RATS algorithm is as follows:

(1) compute $W = \max(P + 1, T/S)$,

(2) calculate (β_0, β_1) using a window of W samples in (2),

(3) compute $\hat{E}_p$ using (4),

(4) compute $E_p = \Delta \cdot \hat{E}_p$,

(5) if $E_p < \epsilon_{\min}$, then $S = S \cdot MIMD_{\text{inc}}$
else if $E_p > \epsilon_{\max}$, then $S = S/MIMD_{\text{dec}}$,

(6) if $S < S_{\min}$, then $S = S_{\min}$
else if $S > S_{\max}$, then $S = S_{\max}$.

RATS starts with calculating the optimal window size W using the optimal time window T for the given sampling period S. The relative clock model is estimated using a linear estimator on the sample history equal to W. The estimation of the prediction error, $\hat{E}_p$, is then computed and scaled using the scaling factor Δ.

If the error of the prediction E_p is below the lower threshold, we multiplicatively increase the sampling period. Conversely, if it is above the higher threshold, the sampling period is decreased multiplicatively. The sampling period remains unchanged if the error is between the two

thresholds. At the end, we make sure that the new sampling period is within $[S_{\min}, S_{\max}]$ to avoid an unbounded increase/decrease of the sampling period.

During a 2–4-hour learning phase, the nodes derive the values of the optimal time window T and scaling factor Δ for a wide range of S values. In [7], it is showed that this initial calibration is consistent with a great number of environments and for long periods of time. In deterministic WSN deployments, the initial calibration can be performed in factory to save postdeployment energy [7]. In random and mobile WSN deployments in factory calibration would not scale, since we ignore which sensor nodes will be neighbors after the deployment. Therefore, in these cases, the calibration must be performed autonomously by the nodes at the deployment site.

4. Secure WSN-Wise Synchronization

Our proposal consists of two periodic phases. In case the WSN needs to also synchronize to UTC time, we propose to add a third phase.

(1) *Secure CN Pairwise (Re-)synchronization*. Each pair of neighbor CNs use the SPS-SE protocol to synchronize, initialize and maintain RATS, and schedule each subsequent time synchronization iteration. In this manner, a common time reference is set up for the WSN.

(2) *Secure BAN (Re-)synchronization*. The CN uses the SPS-SE protocol to synchronize each BAN member. In this manner, a common time reference is set up for the BAN. RATS is accommodated to use it with multiple nodes.

(3) *UTC Synchronization*. WSN time is translated to UTC time.

In the remainder of this paper, let us consider that at each period R a new controller node is elected in each BAN. Note that if the controller node is fixed, then R is the WSN lifetime.

We divide pairwise and BAN time in a number of variable time periods $S^j_{u,v}$ and S^k_{CL}, where $j = 1, 2, \ldots, r_j$ complying $\sum_{j=1}^{r_j}(S^j_{u,v}) = R$ and $k = 1, 2, \ldots, r_k$ complying $\sum_{k=1}^{r_k}(S^k_{\mathrm{CL}}) = R$, respectively. A period $S^j_{u,v}$ or S^k_{CL} can encompass one or more consecutive sleep plus wake intervals. In any case, we define the beginning of each period $S^j_{u,v}$ or S^k_{CL} right to coincide with the beginning of a wake interval.

The duration of a period $S^j_{u,v}$ does not necessarily coincide with any S^k_{CL}, for $i, k > W_d$. The duration of a period $S^i_{u,v}$ does not necessarily coincide with any $S^j_{u,v}$, for $i, j > W_d$. The duration of a period S^i_{CL} does not necessarily coincide with any S^k_{CL}, for $i, k > W_d$.

In the predeployment phase, that is, during manufacture, the sensor nodes are preconfigured with an initial default sampling period S_d and an initial optimal window size W_d. Once deployed, starting from the beginning of BAN existence, the nodes exchange W_d time observations, each sample separated by S_d seconds. The messages used to exchange these observations are protected by secure means.

After $W_d \cdot S_d$ seconds, the sensors derive the first estimate of their clocks using (1). If this estimation error is between the thresholds $[\epsilon_{\min}, \epsilon_{\max}]$, then the sensor nodes use the clock estimations for synchronizing. Otherwise, the sensors keep exchanging time observations each S_d seconds till the estimation error is between the two thresholds. From this moment on, the sensors use the clock estimations for synchronizing.

During the rest of the BAN existence, the quality of the estimation is optimized to the particular conditions of each epoch. The nodes employ RATS to periodically calculate the optimal duration of $S^j_{u,v}$ and S^k_{CL}, $k, j \geq W_d + 1$, to maintain the precision of the clock estimations between the thresholds $[\epsilon_{\min}, \epsilon_{\max}]$. Moreover, a corresponding new optimal window size $W^j_{u,v}$ and S^k_{CL}, $k, j \geq W_d + 1$, is obtained. Additionally, after each interval $S^j_{u,v}$ or S^k_{CL}, $k, j \geq W_d + 1$, the nodes securely exchange a new time sample and recalculate the clock estimations.

These steps are repeated after each controller node reelection.

In the rest of the section, we thoroughly describe the SPS-SE protocol and each of the phases of the synchronization system.

4.1. Secure Pairwise Synchronization with Sample Exchange. We propose a protocol for secure pairwise synchronization and sample exchange that leverages SPS [4]. SPS is based on sender-receiver synchronization. It performs a handshake protocol between two nodes u and v.

The integrity and authenticity of SPS-SE messages are guaranteed using message integrity codes (MICs) and a shared key $K_{u,v}$. Moreover, the MIC provides resistance to pulse-delay attacks and external attackers.

The SPS-SE protocol consists of the following message exchanges (time samples between brackets denote message time of send (tos) or time of arrival (toa)):

(1) $u(\mathrm{tos}_{u_1}) \rightarrow (\mathrm{toa}_{v_1})v : \mathrm{ID}_u, \mathrm{ID}_v, \mathrm{tos}_{u_1}$,

(2) $v(\mathrm{tos}_{v_2}) \rightarrow (\mathrm{toa}_{u_2})u : \mathrm{ID}_v, \mathrm{ID}_u, \mathrm{tos}_{u_1}, \mathrm{toa}_{v_1}, \mathrm{tos}_{v_2}, \mathrm{MIC}_2$

(3) $u(\mathrm{tos}_{u_3}) \rightarrow (\mathrm{toa}_{v_3})v : \mathrm{ID}_u, \mathrm{ID}_v, \mathrm{toa}_{u_2}, \mathrm{tos}_{u_3}, \mathrm{MIC}_3$,

where $\mathrm{MIC}_2 = \mathrm{MIC}_{K_{u,v}}(\mathrm{ID}_v, \mathrm{ID}_u, \mathrm{tos}_{u_1}, \mathrm{toa}_{v_1}, \text{ and } \mathrm{tos}_{v_2})$, $\mathrm{MIC}_3 = \mathrm{MIC}_{K_{u,v}}(\mathrm{ID}_u, \mathrm{ID}_v, \mathrm{toa}_{u_2}, \text{ and } \mathrm{tos}_{u_3})$, and $K_{u,v}$ is the key shared by u and v.

At the end of the protocol, both nodes calculate the SPS message end-to-end delay $d_{u,v}$ as specified in [4]:

$$d_{u,v} = \frac{(\mathrm{toa}_{v_1} - \mathrm{tos}_{u_1}) + (\mathrm{toa}_{u_2} - \mathrm{tos}_{v_2})}{2}. \tag{5}$$

The end-to-end delay is used to detect pulse-delay attacks against SPS-SE.

The clock offset $\delta_{u,v}$ is also calculated as follows:

$$\delta_{u,v} = \frac{(\mathrm{toa}_{v_1} - \mathrm{tos}_{u_1}) - (\mathrm{toa}_{u_2} - \mathrm{tos}_{v_2})}{2}. \tag{6}$$

Subsequently, u and v add the new time sample $(\mathrm{tos}_{u_2}, \mathrm{toa}_{v_2} - d_{u,v})$ to their respective sample repository.

For sensor nodes using crystal oscillators with stability up to 100 ppm, the duration of the protocol is to be bounded to a few hundred milliseconds. In such case, we can assume the clock drift to be negligible and accept the time observations accurate enough for the prediction.

4.2. Synchronization Method. The synchronization method allows two nodes to adapt their respective time measures. We distinguish two methods: short-lasting synchronization and long-lasting synchronization.

4.2.1. Short-Lasting Synchronization. Short-lasting synchronization is used during the initialization phase of RATS for the nodes to establish short-lasting accurate clock synchronization and for exchanging samples for a clock estimation with the required target precision.

Note that because of the low quality of clock crystals, this method cannot be used to maintain a high precision during a relative long time without an expensive energy cost. For instance, the CC2420 can drift up to 80 μs per second, and in 60 seconds the clocks may drift up to 4810 μs. To guarantee a precision below the 100 μs, the nodes would need to synchronize each second.

The method works as follows. Firstly, by using the SPS-SE protocol, two nodes u and v calculate their relative clock offset. Subsequently, to synchronize a node's time measure, with another's clock measure the clock offset is added (or subtracted, as needed). For instance, if sensor u collects and timestamps a data sample at tc_{u_4}, v translates data collection time to $tc_{u_4} - \delta_{u,v}$ to get the time measure relative to its own notion of time.

For subsequent message exchanges between u and v, the message delay d needs also to be taken into account to calculate the synchronized time. For messages timestamped below the MAC layer immediately prior to their transmission, the delay (d) adds the contribution of the transmission time, the propagation time, and the reception time. The transmission time is the time needed for the sender to transmit the message bit by bit at the physical layer. This time can be easily calculated by the receiver by knowing the length in bits of the message and the radio speed. The propagation time is the actual time taken by the message to traverse the wireless link from the sender to the receiver. In WSN, the distance among neighbor sensor nodes is of a few meters. Therefore, because radio waves move at the speed of light and the radio speed is up to a few Mbit/s, the propagation time is neglected compared to the rest of times. The reception time accounts for the time taken by the receiver in receiving the bits and passing them to the MAC layer. This time can also be easily calculated by the receiving node. Thus,

$$d \approx \text{transmission time} + \text{reception time}. \tag{7}$$

For instance, for a timestamped message that u sends at time tos_{u_5} and reaches v at time toa_{v_5}, v will interpret sent time as $\text{toa}_{v_5} - d + \delta_{u,v}$. For instance, v can check the time integrity of the message by verifying that the difference between tos_{u_5} and $\text{toa}_{v_5} - d + \delta_{u,v}$ is below a certain threshold.

4.2.2. Long-Lasting Synchronization. Long-lasting synchronization is used to maintain precise clock synchronization with fine-tuned RATS.

Each new time sample (e.g., $(\text{tos}_{u_2}, \text{toa}_{v_2} - d_{u,v})$) exchanged with SPS-SE includes the offset but not the delay contribution, which is a particular measure of each exchanged message. Therefore, with estimated clocks, for a timestamped message that u sends at time tos_{u_4} and reaches v at time toa_{v_4}, v will interpret sent time as $\widehat{t}_u(\text{toa}_{v_4}) - d$. Furthermore, if sensor u collects and timestamps a data sample at tc_{u_5}, v translates data collection time to $tc_{u_5} - (t_v - \widehat{t}_u(t_v))$ to get the time measure relative to its own notion of time. Here $t_v - \widehat{t}_u(t_v)$ is an estimation of the current offset between t_v and t_u.

4.3. Secure CN Pairwise (Re-)Synchronization. Secure CN pairwise (re-)synchronization is used to periodically synchronize two neighbor CNs. Each and every pair of neighboring CNs of the WSN is to synchronize following this method. In this manner, WSN time is established.

The interval of time $S^1_{\text{CN}_u,\text{CN}_v}$ starts right after two newly elected CNs CN_u and CN_v discover each other by physical and MAC layer means (the description of these means is out of the scope of this paper).

During time periods $S^j_{\text{CN}_u,\text{CN}_v}$, $j = 1, 2, \ldots, W_d$, BAN controller nodes use the short-lasting synchronization method. Additionally, this time is also employed to exchange the first W_d time samples. Right at the beginning of each time period $S^j_{\text{CN}_u,\text{CN}_v}$, $j = 1, 2, \ldots, W_d$, by using the SPS-SE protocol nodes CN_u and CN_v synchronize and exchange a time sample $(t_{\text{CN}_u,j}, t_{\text{CN}_v,j})$, $j = 1, 2, \ldots, W_d$. To detect wormhole and pulse-delay attacks, each CN also measures the maximum SPS-SE expected message delay $d_{\text{CN}_u,\text{CN}_v}$.

At the beginning of period $S^{W_d+1}_{\text{CN}_u,\text{CN}_v}$ both CNs calculate the first clock estimations and initialize RATS for the first time. At the end of $S^{W_d+1}_{\text{CN}_u,\text{CN}_v}$ both CNs estimate their relative clock offset as follows:

$$\widehat{\delta}_{\text{CN}_u,\text{CN}_v} = t_{\text{CN}_v} - \widehat{t}_{\text{CN}_u}(t_{\text{CN}_v}). \tag{8}$$

If $\widehat{\delta}_{\text{CN}_u,\text{CN}_v}$ is below the required accuracy threshold $\epsilon_{\max}$, then RATS is considered to be fine-tuned. Consequently, BAN controller nodes switch to the long-lasting synchronization method for the following BAN periods.

Otherwise, yet the synchronization method to be used is short-lasting synchronization for subsequent BAN periods $S^j_{\text{CN}_u,\text{CN}_v}$, $W_d + 2 \le j \le r_j$, till the condition $\widehat{\delta}_{\text{CN}_u,\text{CN}_v} \le \epsilon_{\max}$ is satisfied. Let us refer to the period when this condition is satisfied as $S^{j_{\text{opt}}}_{\text{CN}_u,\text{CN}_v}$. Typically, $W_d + 2 \le j_{\text{opt}} \ll r_j$.

During BAN periods $S^j_{\text{CN}_u,\text{CN}_v}$, $j_{\text{opt}} + 1 \le j \le r_j$, BAN controller nodes use the long-lasting synchronization methods. Right at the beginning of each of these periods, CN_u and CN_v exchange a new time sample $(t_{\text{CN}_u,j}, t_{\text{CN}_v,j})$ by using the SPS-SE protocol and add it to their respective sample repository. RATS is employed to periodically recalculate $S^j_{\text{CN}_u,\text{CN}_v}$ (see Section 4.3.1). Additionally, the clock estimations are recalculated using (1). Finally, a real measure of the clock offset is calculated using the SPS-SE protocol

to validate the estimation of the clock offset and, thus, to continuously monitor the quality of the clock estimations.

Since the clocks of CN_u and CN_v drift throughout $S^{j-1}_{CN_u,CN_v}$, $j_{opt}+1 \le j \le r_j$, the measure of $S^{j-1}_{CN_u,CN_v}$ at the end of the period will likely be different at t_{CN_u} and t_{CN_v}. To counter this relativistic effect, we define pairwise period-dependent time of guard. Despite the fact that clocks can get desynchronized, the time of guard guarantees that both CNs are ready to concurrently use the radio channel after long sleeping periods. In order to preserve energy of nodes the time of guard needs to be accurately minimized.

The pairwise period-dependent time of guard to be used at the beginning of $S^{j}_{CN_u,CN_v}$, $j_{opt}+1 \le j \le r_j$, is calculated as follows:

$$T_{guard} = \hat{\delta}_{CN_u,CN_v} + \frac{1}{r_b}. \tag{9}$$

During its T_{guard} each CN is only allowed to receive messages. The first CN exhausting its T_{guard} triggers the re-synchronization.

At the very end of each period $S^{j-1}_{CN_u,CN_v}$, $j_{opt}+1 \le j \le r_j$, the offset $\hat{\delta}_{CN_u,CN_v}$ is accurately predicted by using (8).

The uncertainty included by the data rate r_b is just 4 ppm at 250 kbps. Because nodes of a WSN are separated at most a few tens of meters, the contribution by the propagation delay can be neglected.

4.3.1. Calculation of Optimal Sample Period and Window Size. By using RATS, the two CNs calculate the optimal window size $W^{j}_{CN_u,CN_v}$ for the current period $S^{j}_{CN_u,CN_v}$. Additionally, the optimal duration for the the current period $S^{j}_{CN_u,CN_v}$ is recalculated. The pseudocode for the RATS algorithm is as follows:

(1) compute $W^{j}_{CN_u,CN_v} = \max(P+1, T^{j-1}_{CN_u,CN_v}/S^{j-1}_{CN_u,CN_v})$,

(2) calculate (β_0, β_1) using a window of $W^{j}_{CN_u,CN_v}$ samples in (2),

(3) compute $\hat{E}_p$ using (4),

(4) compute $E_p = \Delta \cdot \hat{E}_p$,

(5) if $E_p < \epsilon_{min}$, then $S^{j}_{CN_u,CN_v} = S^{j}_{CN_u,CN_v} \cdot MIMD_{inc}$

else if $E_p > \epsilon_{max}$, then $S^{j}_{CN_u,CN_v} = S^{j}_{CN_u,CN_v}/MIMD_{dec}$,

(6) if $S^{j}_{CN_u,CN_v} < S_{min}$, then $S^{j}_{CN_u,CN_v} = S_{min}$

else if $S^{j}_{CN_u,CN_v} > S_{max}$, then $S^{j}_{CN_u,CN_v} = S_{max}$.

4.3.2. Estimation of Relative Clock Skew. By using the time observations $(t_{CN_u,i}, t_{CN_v,i})$, where $i = j - W^{j}_{CN_u,CN_v}, j+1-W^{j}_{CN_u,CN_v}, \ldots j$ in (1) and (2), nodes CN_u and CN_v estimate $\hat{t}_{CN_v}(t_{CN_u})$ and $\hat{t}_{CN_u}(t_{CN_v})$, respectively.

4.4. Secure BAN (Re-)Synchronization. Secure BAN (re-)synchronization is used to periodically synchronize BAN members with the CN. This, in turn, guarantees that each BAN member is synchronized to the same reference time. This process establishes BAN time without the need for each BAN member to pairwisely synchronize.

BAN wise synchronization can be scheduled in two different manners. First, we let each node to independently schedule its re-synchronization interval. That is, each node has an own measure of the BAN period $S_{CL,u}$. At the beginning of each node-dependent BAN period, the node synchronizes with the CN. This manner requires the CN to be asleep each time a BAN member u is to synchronize. Because of the independency of the length of each $S_{CL,u}$, $u = 1, 2, \ldots, n$, the requirement of low-duty cycling is hard to comply for the CN.

A second manner consists of letting the CN to schedule a unique re-synchronization interval S_{CL} for all the BAN members. At the beginning of each BAN period, a slot of time is reserved for each node to synchronize with the CN. This scheduling can be designed to accommodate for CN duty cycling requirements.

Observe that to comply that clock estimations are below the required accuracy level during the period S_{CL}, then the CN must select as S_{CL} the minimum duration for a BAN period required by all the nodes of the BAN.

To solve this issue, we have again two possible approaches. The first approach consists of letting each node u of the BAN, $u = 1, 2, \ldots, n$, calculate its own measure of S_{CL}, that is, $S_{CL,u}$. Then, each node independently sends $S_{CL,u}$ to the CN. Finally, the CN heads select $S_{CL} = \min(S_{CL,u})$ for all u.

The second approach consists of the CN calculating $S_{CL,u}$ for all u, $u = 1, 2, \ldots, n$. Then, the CN heads select $S_{CL} = \min(S_{CL,u})$ for all u.

We favor the second approach because it does not require the nodes to send $S_{CL,u}$ to the CN. The need to send messages has implications of added energy consumption and delay both for the BAN members and the CN. The second approach requires much more computational effort in the CN than the first approach. However, the implied energy consumption and delay are neglected compared to the overhead of the first approach.

The rest of the section describes the details of this second approach.

The interval of time S^{1}_{CL} starts right after the BAN is formed. Right at its beginning the CN generates a BAN broadcast key chain of length q by repeatedly hashing a random value K_{CL}. The successive keys $h^i(K_{CL})$, $i = 0 \cdots q-1, q$, are to be used with μTESLA to protect broadcast synchronization messages. We assume that the reader is familiar with μTESLA [8].

The duration of the first W_d time periods S^{k}_{CL}, $k = 1, 2, \ldots, W_d$, is fixed by default. The intervals S^{k}_{CL}, $k = 1, 2, \ldots, W_d$, are used to exchange the first W_d time samples that allow for RATS initialization and for the first clock estimations. At the beginning of each of these periods the CN and each node u of the BAN, $u = 1, 2, \ldots, n$, synchronize and exchange a new time sample $(t_{CN,k}, t_{u,k})$, $k = 1, 2, \ldots, W_d$, by using the SPS-SE protocol. In one of these SPE-SE exchanges, the CN sends the last value of the key chain $h^q(K_{CL})$ for each BAN member.

Because the clocks are not yet estimated, during time periods S_{CL}^k, $k = 1, 2, \ldots, W_d$, the CN and each node u of the BAN, $u = 1, 2, \ldots, n$, use the short-lasting synchronization method. Note that because of clock drifts, CN and each node u may need to re-synchronize multiple times during the duration of any period S_{CL}^k, $k = 1, 2, \ldots, W_d$.

At the beginning of period $S_{CL}^{W_d+1}$ the CN calculates the first clock estimations $\hat{t}_u(t_{CN})$, $u = 1, 2, \ldots, n$, and initializes RATS for the first time. At the end of $S_{CL}^{W_d+1}$ the CN and each node u estimate their relative clock offset as follows:

$$\hat{\delta}_{CN,u} = t_{CN} - \hat{t}_u(t_{CN}). \tag{10}$$

If $\hat{\delta}_{CN,u}$ is below the required accuracy threshold ϵ_{max}, then RATS is considered to be fine-tuned for the CN and the corresponding node u. Consequently, the CN and the node u complying $\hat{\delta}_{CN,u} \leq \epsilon_{max}$ switch to the long-lasting synchronization method for the following BAN periods.

The nodes not yet complying $\hat{\delta}_{CN,u} \leq \epsilon_{max}$ are to use the short-lasting synchronization method for subsequent BAN periods S_{CL}^k, $W_d + 2 \leq k \leq r_k$, till the condition $\hat{\delta}_{CN,u} \leq \epsilon_{max}$ is satisfied.

In this moment the BAN can have from 0 to n nodes synchronized with the long-lasting method. The remaining nodes, up to the n nodes still use the short-lasting method. This status exists till all the BAN members switch to long-lasting synchronization. For both groups of nodes the subsequent measure of S_{CL} is different. The first adapts the measure of S_{CL} by using RATS. The second keep using the default value of S_{CL}.

Let us refer to the period when one or more BAN members first switch to long-lasting synchronization as $S_{CL}^{k_{opt}}$. Typically, $W_d + 2 \leq k_{opt} \ll r_k$. Let us use n' to refer to the BAN members using long-lasting synchronization. In the rest of the section we describe the details of long-lasting synchronization.

Secure BAN long-lasting re-synchronization is performed at the beginning of each period S_{CL}^k, $W_d + 2 \leq k_{opt} \ll r_k$. By using the SPS-SE protocol, nodes CN and u, $u = 1, 2, \ldots, n'$, re-synchronize and exchange a new time sample $(t_{CN,W_d+k-1}, t_{u,W_d+k-1})$ and add it to their respective repository. Additionally, each node u calculates $\hat{t}_{CN}(t_{u,W_d+k-1})$. RATS is employed to periodically recalculate S_{CL}^k (see Section 4.4.1).

When each and every pair (CN, u), for $u = 1, 2, \ldots, n'$, of the BAN is synchronized, then BAN time is established.

Since the clocks of CN and u, $u = 1, 2, \ldots, n'$, drift throughout S_{CL}^{k-1}, the measure of S_{CL}^{k-1} at the end of the period will likely be different at t_{CN} and t_u. To counter this relativistic effect, we define BAN period and node-dependent times of guard $T_{guard,u}$ (see Figure 1) to be used at the beginning of S_{CL}^k:

$$T_{guard,u} = \delta_{CN,u} + \frac{1}{r_b} + B, \tag{11}$$

where u, $1 \leq u \leq n'$, is the local identifier of each BAN member and B is the time required to run the SPS-SE protocol in three steps. This method allocates a different time slot for SPS-SE-based synchronization between the CN and a node u.

At the very end of each period S_{CL}^{k-1} the CN calculates $\delta_{CN,u} = t_{CN} - \hat{t}_u(t_{CN})$, for $u = 1, 2, \ldots, n'$. Each node u calculates $\delta_{CN,u} = \hat{t}_{CN}(t_u) - t_u$.

The CN does not need to contend to access the wireless media. After S_{CL}^{k-1} and each subperiod $T_{guard,u}$ are exhausted, the CN is the only node in the BAN allowed to start communication. After receiving an initial message from the CN, just the corresponding node u, $u = 1, 2, \ldots, n'$, is allowed to answer.

4.4.1. Calculation of Optimal Sample Period. By leveraging RATS, the CN calculates the optimal duration for the current period S_{CL}^k. The pseudocode for the RATS algorithm is as follows:

(1) compute $W_{CN,u}^k = \max(P + 1, T_{CL}^{k-1}/S_{CL}^{k-1})$,

(2) calculate (β_0, β_1) using a window of $W_{CN,u}^k$ samples in (2),

(3) compute $\hat{E}_p$ using (4),

(4) compute $E_p = \Delta \cdot \hat{E}_p$,

(5) if $E_p < \epsilon_{min}$, then $S_{CN,u}^k = S_{CN,u}^k \cdot MIMD_{inc}$
else if $E_p > \epsilon_{max}$, then $S_{CN,u}^k = S_{CN,u}^k / MIMD_{dec}$,

(6) if $S_{CN,u}^k < S_{min}$, then $S_{CN,u}^k = S_{min}$
else if $S_{CN,u}^k > S_{max}$, then $S_{CN,u}^k = S_{max}$.

Finally, $S_{CL}^k = \min(S_{CN,u}^k)$ for all u, $u = 1, 2, \ldots, n'$.

Right after $T_{guard,n}^k + B$, the CN broadcasts the current period S_{CL}^k in an integrity-protected message under key $h^{q-k}(K_{CL})$. After $\max(d_{CN,u})$ seconds, for all u, the CN reveals $h^{q-k}(K_{CL})$ (see Figure 2).

In receiving $h^{q-k}(K_{CL})$ each node u first validates the authenticity of the key by hashing it and comparing it with the previous stored authentic value $h^{q-k+1}(K_{CL})$. If the validation is positive, then the node stores $h^{q-k}(K_{CL})$ to be used in the next BAN time period. Subsequently, the integrity of the message containing S_{CL}^k is verified. Finally, the value S_{CL}^k is stored for scheduling the next re-synchronization.

4.4.2. Estimation of CN Time. By leveraging RATS, each node u, for $u = 1, 2, \ldots, n'$, independently calculates $W_{CN,u}^k$ with the same pseudocode the CN used (see Section 4.4.1). The node stores the obtained $W_{CN,u}^k$ but ignores the obtained $S_{CN,u}^k$. Instead, it will use $S_{CN,u}^k = S_{CL}^k$ as next sampling period.

By using the time observations $(t_{CN,i}, t_{u,i})$, where $i = k - W_{CN,u}^k, k + 1 - W_{CN,u}^k, \ldots k$ in (1) and (2), each node u estimates $\hat{t}_{CN}(t_u)$.

4.5. UTC Synchronization. We propose to securely pairwise synchronize the base station(s) with the CNs to which it is wireless connected using secure pairwise CN synchronization. Additionally, the base station is to be securely synchronized to UTC time by other means (the details of this synchronization means is out of the scope of this paper).

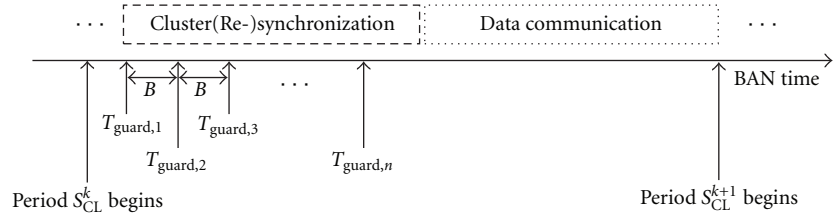

Figure 1: BAN and node-dependent times of guard.

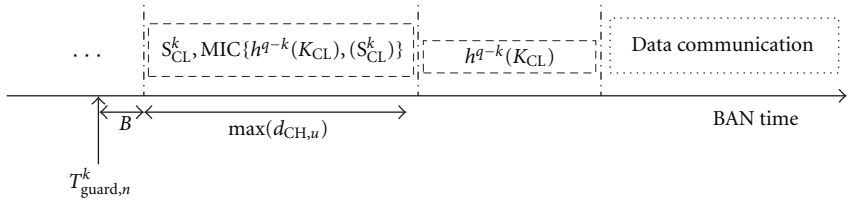

Figure 2: Usage of μTesla.

Then, a correspondence WSN to UTC is then simple at the base station.

5. Security Analysis and Countermeasures

In this section, we identify threats and propose countermeasures to strengthen the security of our synchronization system. Because all the messages are integrity protected, confidentiality protection is provided when needed, and SPS is robust to pulse-delay attacks, the system is robust against external attackers.

In the rest of the section, we present threats and countermeasures for compromised nodes.

5.1. Coping with a Compromised CN. Because of their key mission in the synchronization system, CN*s* are an interesting target for attackers. In any case the effect of a compromised CN is bounded to the interval R.

A compromised CN_c may fake (a subset of) the time samples $\widetilde{t}_{\text{CN}_c,i}$ to be sent to v, $i = 1, 2, \ldots, W_d + k - 1$.

To detect this attack, we use the end-to-end delay. The end-to-end delay is bounded by the maximum and minimum expected delay $d_{\max}$ and $d_{\min}$, respectively. After the SPS-SE protocol is run, v can confidently approximate $d_{\text{CN}_c,v}$. If $d_{\min} \leq d_{\text{CN}_c,v} \leq d_{\max}$, then each faked time sample is rejected.

This method serves us to also detect *wormhole* and *pulse-delay* attacks. Recall that in pulse-delay and wormhole attacks the adversary delays and rushes the authenticated synchronization messages, respectively. To detect a pulse-delay, the sensor node checks if $d_{\text{CN}_c,v} \geq d_{\max}$. To detect a wormhole, the sensor node checks if $d_{\text{CN}_c,v} \leq d_{\min}$.

In secure BAN re-synchronization, a compromised CN_c can fake samples $\widetilde{S}_{\text{CL}}^k$ and $\widetilde{\delta}_{\max}^k$ for a given k. It may assign $\widetilde{S}_{\text{CL}}^k$ a value substantially greater or lower than the actual S_{CL}^k. If $\widetilde{S}_{\text{CL}}^k \gg S_{\text{CL}}^k$, then the value of $T_{\text{guard},u}^k$ for all u becomes expanded. If $\widetilde{S}_{\text{CL}}^k \ll S_{\text{CL}}^k$, then the value of $T_{\text{guard},u}^k$ for all u becomes contracted. Additionally, the nodes need to re-synchronize more frequently than the optimal re-synchronization period. In both situations, the effect is to increase the required duty cycle in nodes and, in turn, to consume more energy than the optimal.

To overcome this threat, when a CN broadcasts S_{CL}^k, it must commit the identity of the node u_x such that $W_{\text{CL}}^k = W_{\text{CN},u_x}^k$. Node u_x verifies that the released S_{CL}^k corresponds to W_{CN,u_x}^k.

Alternatively, especially to cope with scenarios where CN and node u_x are compromised, lower and upper acceptable bounds for S_{CL}^k can be calculated by each node.

5.2. Coping with Colluding CNs. A number of neighbor compromised CNs may collide together to create a delayed path through them.

We discuss this attack by assuming the BAN controller nodes in the path $\text{CN}_{c1} - \text{CN}_{c2} - \text{CN}_{c3}$ collide, that is, in the moment of secure pairwise CN synchronization they introduced an additional delay in the links $\text{CN}_{c1} - \text{CN}_{c2}$ and $\text{CN}_{c2} - \text{CN}_{c3}$.

To solve the attack we exploit a design property of WSNs for increased reliability and power-efficiency. We assume that there exist multiple routes connecting each pair of CNs.

We propose that a fourth legitimate BAN controller node CN_4, which is connected to any of the colliding nodes, detects the delay attack. CN_4 compares the delay introduced by the compromised path with the delay introduced by any or a number t of other paths. The countermeasure consists of adding CN_{c1}, CN_{c2} and CN_{c3} to a blacklist of untrusted nodes and trigger re-election of controller node.

5.3. Coping with Compromised BAN Members. A compromised node u_c can fake (a number of) time samples $\widetilde{t}_{u_c,i}$ to be sent to its CN, $i = 1, 2, \ldots, W_d + k - 1$.

The CN can detect the attack by using the end-to-end delay, as in the case for a CN cheating a node.

Table 1: Performance evaluation parameters.

Data Rate	250 kbps
Clock drift	40 ppm
Confidence interval to estimate E_p	90
$MIMD_{\text{inc}}$ and $MIMD_{\text{dec}}$ factors	2
Maximum $S_{\max}$ and minimum $S_{\min}$ sampling periods	64 s and 30 s
Upper n_{high} and lower n_{low} threshold fractions	0.75 and 0.9

To counter this attack, u_c is added to the blacklist of untrusted nodes.

5.4. Temperature Attacks. In order to de-synchronize some nodes, an attacker may select a location and rapidly vary the temperature in the surroundings of one or group of BANs. For instance, in an indoor scenario, the attacker could increase the heating temperature or decrease the cooling temperature.

We believe this kind of attack to be unpractical in BAN applications since the users would quickly realise and stop the heating or cooling system.

6. Performance Analysis

In this section, we analyse the level of accuracy, precision, energy-efficiency, low-duty cycling, and communication overhead that the system can achieve for MICA2 motes.

The results of accuracy and precision are based on experimental findings borrowed from [4, 7]. We assume MAC layer time stamping and cryptographic computation to achieve a high degree of accuracy.

The rest of properties are derived theoretically.

The parameters on Table 1 are used for the performance evaluation of the proposed system.

For simplicity's sake, in the rest of the section, we consider the case with no compromised nodes.

6.1. Pairwise Accuracy and Precision of SPS-SE. The minimum pairwise clock precision (maximum error) of the SPS-SE protocol is 8.46 μs. Therefore, right after two sensors run the SPS-SE protocol, the accuracy of their synchronized clocks ranges from 0 to 8.46 μs.

This high level of accuracy can only be maintained at a expensive energy cost, since the node clocks can drift up to 80 μs per second. For instance, in 60 seconds the clocks may drift up to 4810 μs.

Therefore, SPS-SE clock synchronization is to be used uniquely to initially exchange the samples for RATS and/or to synchronize the clocks for application requirements.

6.2. Accuracy and Precision of BAN Time. The minimum precision of BAN time combines both the precision of SPS-SE and RATS.

The precision of BAN time depends on the values S_{CL}^k and W_{CL}^k. Figure 3 shows the minimum precision for a range of S_{CL}^k values for three scenarios: Indoor with a temperature range of 25-26°C, Outdoor I with a temperature range of 17–21°C, and Outdoor II with a temperature range of 22–27°C. The value of W_{CL}^k used in the experiments is 2.

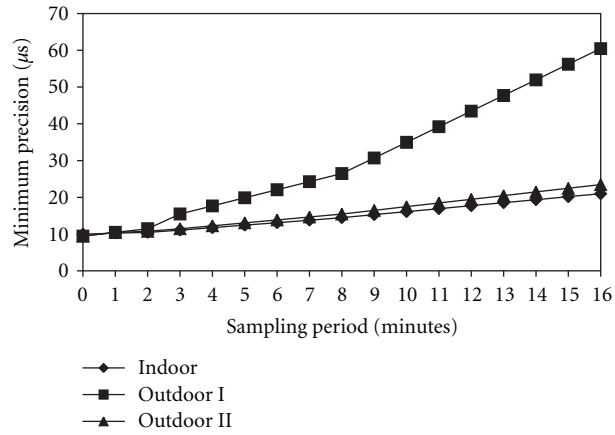

Figure 3: Minimum precision of BAN time versus sampling period.

Table 2: SPS packet length.

Nodes Id	8 bytes
Timestamp	8 bytes
MIC	16 bytes
PHY and MAC layers	17 byte

Figure 3 shows that the sampling period S_{CL}^k does not significantly affect precision of the clock prediction in Indoor and Outdoor II. Indoor achieves a precision of 20.96 μs with S_{CL}^k 16 minutes. This result is a great achievement for BANs in terms of expected accuracy level and energy requirements.

6.3. SPS-SE Duration. The length in time of the SPS-SE protocol can be approximated by the time needed to exchange three messages containing 6 node IDs, 6 timestamps, and 2 MICs. A sufficient number of nodes can be accommodated with 8-byte node IDs (such as in ZigBee networks). A length of 8 bytes is also a fair number for timestamps. A MIC of 16 bytes offers enough security for WSN applications. The physical and MAC layer add a minimum of 17 bytes per message [22]. Then, neglecting other sources of delay, at 250 kbps the SPS-SE protocol exchange lasts approximately 5,72 ms. Table 2 show the parameters used to compute the SPS packet length.

6.4. Minimum Accuracy and Precision of WSN Time. In the initial phase, while RATS is not yet tuned, CNs use SPS-SE for synchronization. Let us define as N_{CN} the maximum number of CNs in the longest synchronization path of the WSN. When WSN time synchronization is triggered, the nodes of the path synchronize sequentially by consecutive pairs. The first pair of nodes synchronizes with minimum accuracy 8.46 μs. While the second pair of nodes synchronize, the clock of the first two synchronized nodes may drift up to 40 ppm times the duration of SPS-SE. The SPS-SE is used $N_{\text{CN}} - 2$

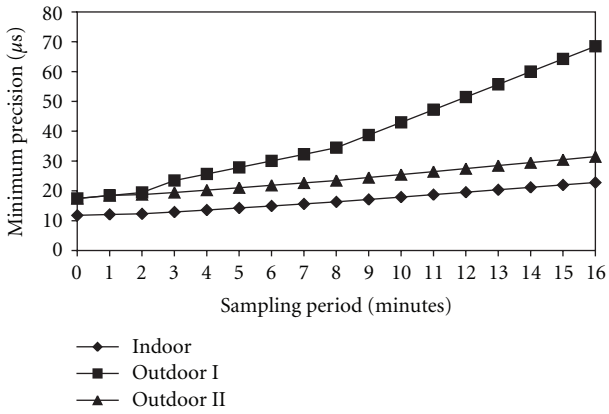

Figure 4: Minimum precision of WSN time versus sampling period, $N_{CN} = 10$.

times after the initial one. Therefore, the minimum accuracy of WSN time can be approximated as $8.46 + 0,2288(N_{CN} - 2)\,\mu$s.

During the RATS phase, the precision of WSN time depends on the values S^k_{CL}, W^k_{CL} and on the number of intermediate CNs. Figure 4 shows the minimum precision for $N_{CN} = 10$ and a range of S^k_{CL} values for three scenarios: Indoor with a temperature range of 25-26°C, Outdoor I with a temperature range of 17–21°C, and Outdoor II with a temperature range of 22–27°C. The value of W^k_{CL} used in the experiments is 2.

Figure 4 (cf. Figure 3) demonstrates that the number of intermediate CNs N_{CN} does not significantly affect precision of the clock prediction.

6.5. Applicability for Low-Duty Cycle Nodes. In this section we demonstrate applicability of the synchronization system for low-duty cycle nodes. Since CNs need to be active longer periods than BAN members and any node may become CN, we only analyze the minimum duty cycle required for a CN.

We use sampling window size $W^k_{CL} = 2$ and period $S^k_{CL} < 16$ minutes, which allow a high level of precision in any scenario, as demonstrated in Figures 3 and 4.

During the RATS phase, a CN needs to be active for each secure pairwise CN re-synchronization and for secure BAN re-synchronization. The initial synchronization is ignored in this analysis as it occurs when nodes need to otherwise increase their duty cycle for BAN formation purposes.

For BAN time synchronization, in each period $S_{C,k}$, a CN needs to be active a minimum of $n \cdot T^k_{guard,u} + \max(d_{CN,u})$ for all u seconds, where $T^k_{guard,u}$ is given by (11). Again, we consider a worst case where the maximum clock skew during the re-synchronization period is the minimum BAN time precision:

(i) Indoor: $\delta^{k-1}_{max} = 20,96\,\mu$s,

(ii) Outdoor I: $\delta^{k-1}_{max} = 60,46\,\mu$s,

(iii) Outdoor II: $\delta^{k-1}_{max} = 23,46\,\mu$s.

Similarly, the period of activity of a CN is the time used for secure pairwise CN re-synchronization. Its value is $ch \cdot T^k_{guard,CN_u} + \max(d_{CN,CN_u})$ for all CN_u seconds. Here ch accounts for the maximum number of neighboring CNs, and $ch \leq n$, so that WSN reliability and scalability is maximized.

Table 3: Worst-case duty cycle.

Scenario	Period of activity (msec.)	Duty cycle
Indoor	94,25936	0,0000981868
Outdoor I	644,01136	0,000670845
Outdoor II	643,41936	0,000670229

The value of $\max(d_{CN,u})$ can be approximated by the time to send the largest message (in step (2)) of SPS-SE. This adds 2.34 ms. The optimal BAN size of a WSN is $5 \leq n \leq 8$ [23]. For this study, we consider the upper limit for a BAN size, that is, $n = 8$. We also assume that a CN may have up to $ch = 8$ neighbouring CNs.

Table 3 shows the period of activity and the duty cycle for the CN. As it can be seen, the duty cycle is much below 1% (considering a sampling period S^k_{CL} of 16 minutes).

6.6. Communication Overhead. The maximum number of bytes a CN needs to send is $(ch+n-1)(\text{length}[\text{SPS}-\text{SE}(1)] + \text{length}[\text{SPS}-\text{SE}(3)] + \text{length}[S_{C,k}] + \text{length}[h^{q-k}(K_{CL})])$. This same CN receives $(ch+n-1) \cdot \text{length}[\text{SPS}-\text{SE}(2)]$ bytes. We use $\text{length}[\text{SPS} - \text{SE}(i)]$ to denote the length of message i of the SPS − SE protocol.

A BAN member sends $\text{length}[\text{SPS} - \text{SE}(2)]$ and receives $\text{length}[\text{SPS} - \text{SE}(1)] + \text{length}[\text{SPS} - \text{SE}(3)] + \text{length}[S_{C,k}] + \text{length}[h^{q-k}(K_{CL})]$ bytes.

We can codify periods S^k_{CN} of 16 minutes at the μs precision with 4 bytes. A key length of 16 bytes is considered to be secure for WSNs. As in the previous section ch is bounded by n, and we consider $n = 8$.

Further considering the values for IDs, timestamps, and MICs of previous sections, the maximum number of bytes sent and received by a CN is 1580 and 1095, respectively.

6.7. Energy Efficiency. The number of Ah consumed by the CC2420 can be calculated according to the formulas in [10]. To send 1580 bytes, the radio module consumes 276.67 nAh, and, to receive 1095 bytes, consumes 169.36 nAh. The total battery consumption for a CN for synchronization is 446.03 nAh over 16 minutes.

Equipped with a 30 mAh cell battery, the node can assume the CN role for over 2 years. If the $n = 8$ nodes of the BAN, rotate the CN role, then the BAN will survive approximately 16 years (assuming that the nodes do nothing else but synchronizing).

7. Conclusions and Future Work

In this paper we have addressed the issue of secure, accurate, and precise synchronization in a WSN formed by the interconnection of multiple BANs.

We have exhaustively analyzed the related work and found a number of open issues for research. Additionally, we have proposed secure, accurate, and precise synchronization

service for this particular kind of WSN. It can be used to provide secure pairwise, BAN-wise, and WSN-wise clock synchronization. We have also discussed a means to synchronize WSN time to UTC time.

We have analyzed both the performance and security of our proposal. We have presented very simple countermeasures to cope with compromised nodes. We have also shown that the synchronization service achieves minimal pairwise accuracy of 8.46 μs.

We have obtained the BAN clock synchronization precision of our system in three scenarios for re-synchronization periods of up to 16 minutes:

(i) indoor with a temperature range of 25-26°C: precision of 20.96 μs,

(ii) outdoor with a temperature range of 17–21°C: precision of 60.46 μs,

(iii) outdoor with a temperature range of 22–27°C: precision of 23.46 μs.

The minimum precision of WSN time is 22.79 μs, 68.46 μs, and 31.34 μs, for each aforementioned scenario, respectively. For these calculations, we considered a number of 10 consecutive BAN controller nodes and a re-synchronization period of 16 minutes.

Yet another interesting result is that sensor node equipped with a 30 mAh (low-resource) cell battery can assume the CN role and help BAN members to synchronize for over 2 years without changing batteries. If the $n = 8$ nodes of the BAN, rotate the CN role, then the BAN will survive approximately 16 years (assuming that the nodes do nothing else but synchronizing).

These results are based on computations.

Acknowledgment

This work has been funded by the Research Project COOLNESS (218163-FP7-PEOPLE-2007-3-1-IAPP).

References

[1] K. Römer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, 2004.

[2] M. Manzo, T. Roosta, and S. Sastry, "Time synchronization in networks," in *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '05)*, pp. 107–116, November 2005.

[3] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," in *Proceedings of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS '05)*, pp. 765–772, November 2005.

[4] S. Ganeriwal, S. Capkun, C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proceedings of the ACM Workshop on Wireless Security (WiSe '05)*, pp. 97–106, September 2005.

[5] K. Sun, P. Ning, and C. Wang, "Secure and resilient clock synchronization in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 395–408, 2006.

[6] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "TinySeRSync: secure and resilient time synchronization in wireless sensor networks," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 264–277, Alexandria, Va, USA, November 2006.

[7] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, and M. B. Srivastava, "Estimating clock uncertainty for efficient duty-cycling in sensor networks," in *Proceedings of the 3rd international Conference on Embedded Networked Sensor Systems*, pp. 130–141, 2005.

[8] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: security protocols for sensor networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (Mobicom '01)*, pp. 189–199, July 2001.

[9] Chipcon, "SmartRF CC2420 Datasheet (rev 1.3), 2005-10-03".

[10] D. Sanchez, *Key management for wireless ad hoc networks*, Ph.D. thesis, IKMZ BTU Cottbus, 2006.

[11] Y. C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03)*, pp. 1976–1986, April 2003.

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

[13] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 138–149, November 2003.

[14] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 39–49, November 2004.

[15] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: a defense against wormhole attacks in wireless ad hoc networks," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, April 2003.

[16] J.-H. Hoepman, A. Larsson, E. M. Schiller, and P. Tsigas, "Secure and self-stabilizing clock synchronization in sensor networks. In the proceedings of the 9th international symposium on self stabilization, safety, and security of distributed systems (SSS 2007)," in *Lecture Notes in Computer Science*, vol. 4838, pp. 340–356, Springer, 2007.

[17] S. A. Çamtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," in *Proceedings of 9th European Symposium on Research in Computer Security (ESORICS '04)*, vol. 3193 of *Lecture Notes in Computer Science*, pp. 293–308, Sophia Antipolis, France, 2004.

[18] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences," *Information and Computation*, vol. 146, no. 1, pp. 1–23, 1998.

[19] D. Sánchez and H. Baldus, "A deterministic pairwise key pre-distribution scheme for mobile sensor networks," in *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm '05)*, pp. 277–288, September 2005.

[20] J. R. Vig, "Introduction to quartz frequency standards," Tech. Rep. SLCET-TR-92-1 (Rev. 1), October 1992.

[21] Y. Wu, S. Fahmy, and N. B. Shroff, "Optimal sleep/wake scheduling for time-synchronized sensor networks with QoS guarantees," in *Proceedings of the 14th IEEE International*

Workshop on Quality of Service (IWQoS '06), pp. 102–111, June 2006.

[22] J. A. Gutierrez, E. Callaway, and R. Barrett, *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4*, Institute of Electrical & Electronics Engineers, 2003.

[23] J. Hwang and Y. Kim, "Revisiting random key pre-distribution schemes for wireless sensor networks," in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 43–52, October 2004.